Simfit

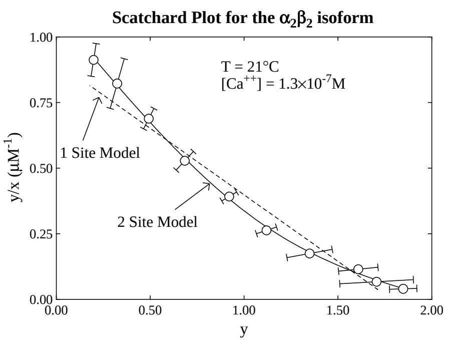
Simulation, fitting, statistics, and plotting.

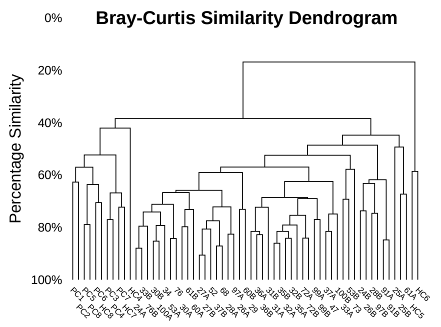
William G. Bardsley

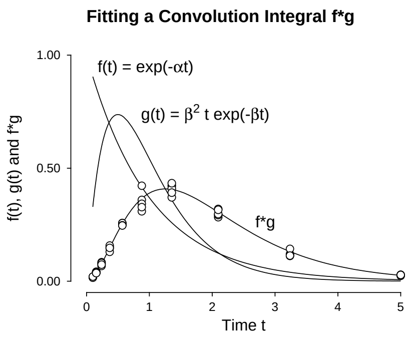
https://simfit.uk

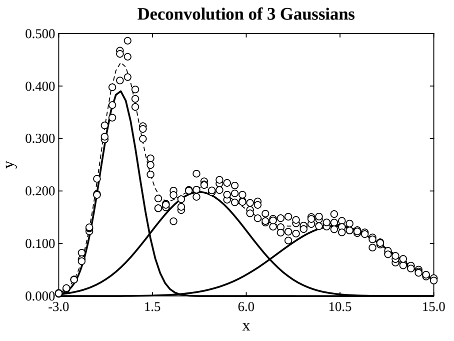
https://simfit.org.uk

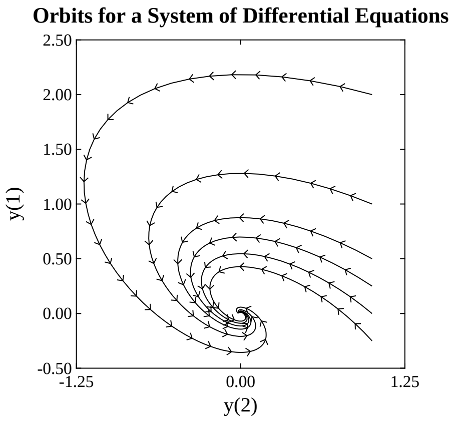
https://simfit.silverfrost.com

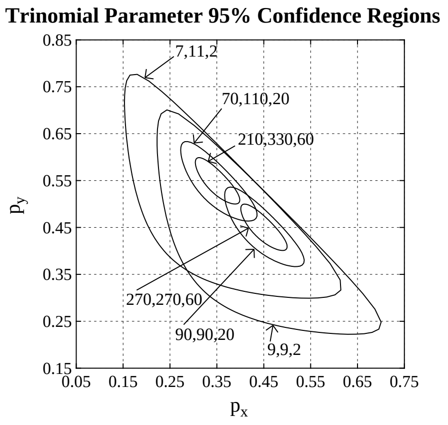












Reference Manual: Version 8.0.0

Contents

List of Tables

List of Figures

Part 1

Overview

SimFIT is a free software Open Source package for simulation, curve fitting, plotting, statistics, and numerical analysis, supplied in compiled form for end-users, and source form for programmers. It runs in Windows, but also Linux (under Wine), and Mac (under Crossover). The academic version is completely free, but there is also a NAG version with additional features which requires the NAG library DLLs.

Applications

analysis	*inverses, eigenvalues, determinants, SVD, zeros, quadrature, optimization,*
biology	*allometry, growth curves, bioassay, flow cytometry,*
biochemistry	*ligand binding studies, cooperativity analysis, metabolic control modelling,*
biophysics	*enzyme kinetics, initial rates, lag times, asymptotes,*
chemistry	*chemical kinetics, complex equilibria,*
ecology	*Bray-Curtis similarity dendrograms, K-means clusters, principal components,*
epidemiology	*population dynamics, parametric and nonparametric survival analysis,*
immunology	*nonlinear calibration with 95% x-prediction confidence limits,*
mathematics	*plotting phase portraits, orbits, 3D curves or surfaces,*
medicine	*power and sample size calculations for clinical trials,*
pharmacology	*dose response curves, estimating LD50 with 95% confidence limits,*
pharmacy	*pharmacokinetics, estimating AUC with 95% confidence limits,*
physics	*simulating and fitting systems of differential equations,*
physiology	*solute transport, estimating diffusion constants, or*
statistics	*data exploration, tests, fitting generalized linear models.*

Summary

SimFIT consists of some forty programs, each dedicated to a special set of functions such as fitting specialized models, plotting, or performing statistical analysis, but the package is driven from a program manager which also provides options for viewing results, editing files, using the calculator, printing files, etc.

SimFIT has on-line tutorials describing available functions, and test data files are provided so all that first time users need to do to demonstrate a program is to select the test file supplied, then observe the analysis. Results are automatically written to log files, which can be saved to disk, or browsed interactively so that selected results can be printed or copied to the clipboard.

SimFIT data sets can be stored as ASCII text files, or transferred by the clipboard into SimFIT from spreadsheets. Macros are provided (e.g., `simfit4.xls` and `simfit6.xls`) to create files from data in MS Excel, and documents are supplied to explain how to incorporate SimFIT graphics into word processors such as MS Word, or how to use PostScript fonts for special graphical effects.

SimFIT has many features such as: wide coverage, great versatility, fast execution speed, maximum likelihood estimation, automatic fitting by user-friendly programs, constrained weighted nonlinear regression using systems of equations in several variables, or the ability to handle large data sets.

Students doing statistics for the first time will find it very easy to get started with data analysis, such as doing t or chi-square tests, and advanced users can supply user defined mathematical models, such as systems of differential equations and Jacobians, or sets of nonlinear equations in several independent variables for simulation and fitting.

SimFIT also supports statistical power calculations and many numerical analysis techniques, such as nonlinear optimization, finding zeros of n functions in n variables, integrating n functions of m variables, calculating determinants, eigenvalues, singular values, matrix arithmetic, etc.

1.1 Installation

Details for installation and configuration will be found in the files `install.pdf` and `configure.pdf` which are distributed with the package. A summary follows.

1. **The installation programs** `simfit_setup8_x_y.exe`, **and** `x64_simfit_setup8_x_y.exe`.
   These can be obtained (in .zip files) from `https://simfit.uk` and contain the whole SimFIT package together with documentation and test files. You should uninstall any existing SimFIT installations before installing if you want, but the installation program will simply overwrite any existing SimFIT files, as long as they do not have the read-only attribute. You should install the package in the default top-level SimFIT folder, say `C:\Program Files\Simfit`, by double clicking on the installation program and accepting the default options, unless you have very good reasons not to. The installation program will create binary, demonstration, documentation, results, and user sub-folders.
2. **The program drivers** `w_simfit.exe`, **and** `x64_simfit.exe`.
   You can make a desktop shortcut to the SimFIT drivers in the SimFIT binary sub-folder if you want to drive the package from a desk-top icon.
3. **The top-level folder.**
   There should be no files at all in the SimFIT top-level folder.
4. **The auxiliary programs.**
   You can specify your own editor, clipboard viewer, and calculator or use the Windows defaults. To read the manual you must install the Adobe Acrobat Reader or similar, and for professional PostScript graphics hardcopy you can also install the GSview/Ghostscript packages.
5. **The configuration options.**
   Run the driver then use the [Configure], [Check] and [Apply] buttons until all paths to auxiliary files are correct. The [Check] option will tell you if any files cannot be located and will search your computer for the correct paths and filenames if necessary.

There are SimFIT mirror sites at `https://simfit.silverfrost.com`, and also at `https://simfit.usal.es` which also distributes a Spanish language sub-set of SimFIT.

1.2 Documentation

There are several sources of documentation and help. Each individual program has a tutorial describing the functions provided by that program, and many of the dedicated controls to the more complex procedures have a menu option to provide help. Also there is a help program which provides immediate access to further information about SimFIT procedures, test files, and the readme files (which have technical information for more advanced users). However, the main source of information is the reference manual which is provided in PostScript (`w_manual.ps`), and portable document format (`w_manual.pdf`). Advice about how to use this manual follows.

The SIMFIT manual is in five main sections.

1. **Introduction**
   A very brief overview of the package is given, with collages to illustrate the graphical possibilities, and advice to help with installation. Then the first time user is guided through some frequently used SIMFIT procedures, such as creating data files, performing curve fitting, plotting graphs, or simulating model systems. Anybody interested in exploiting the functionality provided by the SIMFIT package would be well advised to work through the examples given, which touch upon many standard procedures, but with minimal theoretical details.

2. **Data Analysis**
   This takes each SIMFIT procedure and explains the theory behind the technique, giving worked examples of how to use the test files provided with the package to observe the analysis of correctly formatted data. The document `tutorials.pdf` describes how to profit from the worked examples provided at the SIMFIT website. Before users attempt to analyze their own data, they should read the description of how that particular technique performs with the test data provided. It should be obvious that, if SIMFIT fails to read or analyze a user-supplied data set but succeeds with the test data supplied, then the user-supplied data file is not formatted correctly, so the test files provided should be consulted to understand the formatting required. Suggestions as to how users might employ SIMFIT to analyze their own data are given.

3. **Graphics**
   This explains how to use the more advanced SIMFIT plotting functions and is where users must turn in order to find out how to get SIMFIT to create specialized graphs. Anybody interested in this aspect should browse the example plots displayed in the manual. Also, this section explains how to edit PostScript files and create special Postscript graphical effects,

4. **Appendices**
   This contains several appendices dealing with advanced features and listing all the programs and files that make up the SIMFIT package. There are sections which outline the library of mathematical and statistical models and display the necessary equations. Also, there is a description of the syntax required to develop user-defined models giving numerous examples, a list of all SIMFIT programs and test files, and discussion of interfaces to other software, like GSview/Ghostscript and Microsoft Office.

5. **Index**
   This has hyperlinks to the appropriate manual page.

1.3 Plotting

SIMFIT has a simple interface that enables users to create default plots interactively, but it also has an advanced users interface for the leisurely sculpturing of masterpieces. To get the most out of SIMFIT graphics, users should learn how to save ASCII text coordinate files for selected plots, bundle them up into library files or project archives to facilitate plotting many graphs simultaneously, and create configuration files to act as templates for frequently used plotting styles. Also /simfit/ metafiles can be saved at any point and these can be input retrospectively to resume editing.

SIMFIT can drive any printer and is able to create graphics files in all formats, e.g., .eps, .pdf, .png, .svg, .emf, .jpg, .tif, .pcx, and .bmp. However, Users should save the SIMFIT industry standard encapsulated PostScript files `(*.eps)`, because they are compact, have unlimited resolution, can be collected into collages, or transformed into other formats. The .png format for pictures, and .svg format for scientific graphics are now recommended for the web, and are supported by all modern Windows programs. Windows users who do not want such professional quality graphics can still save enhanced metafiles `(*.emf)`. In order to give some idea of the type of plots supported by SIMFIT, figures 1.1, 1.2, and 1.3 should be consulted. These demonstrate typical plots as collages created by SIMFIT program **editps** from SIMFIT PostScript files.

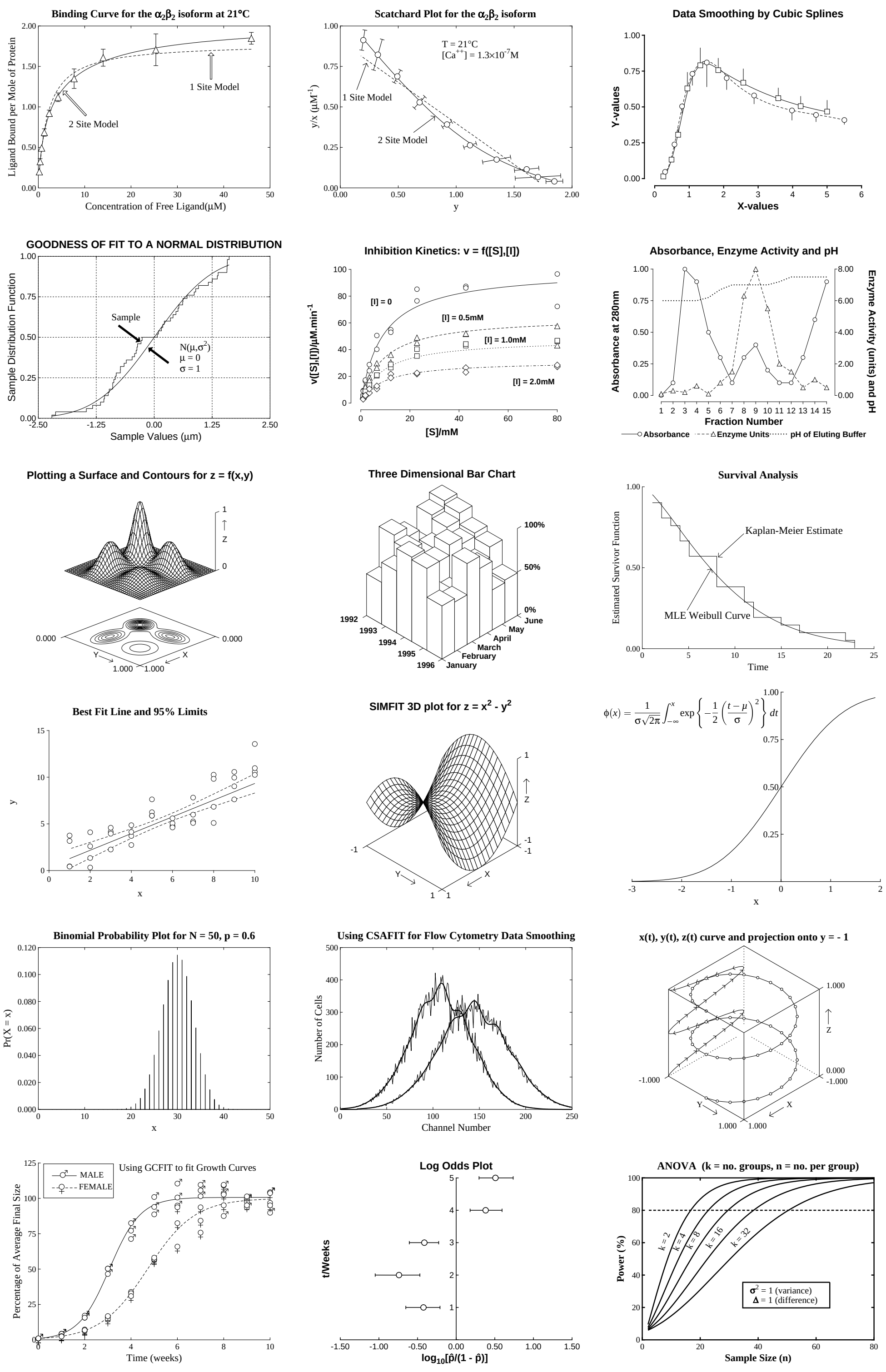


Figure 1.1: Collage 1

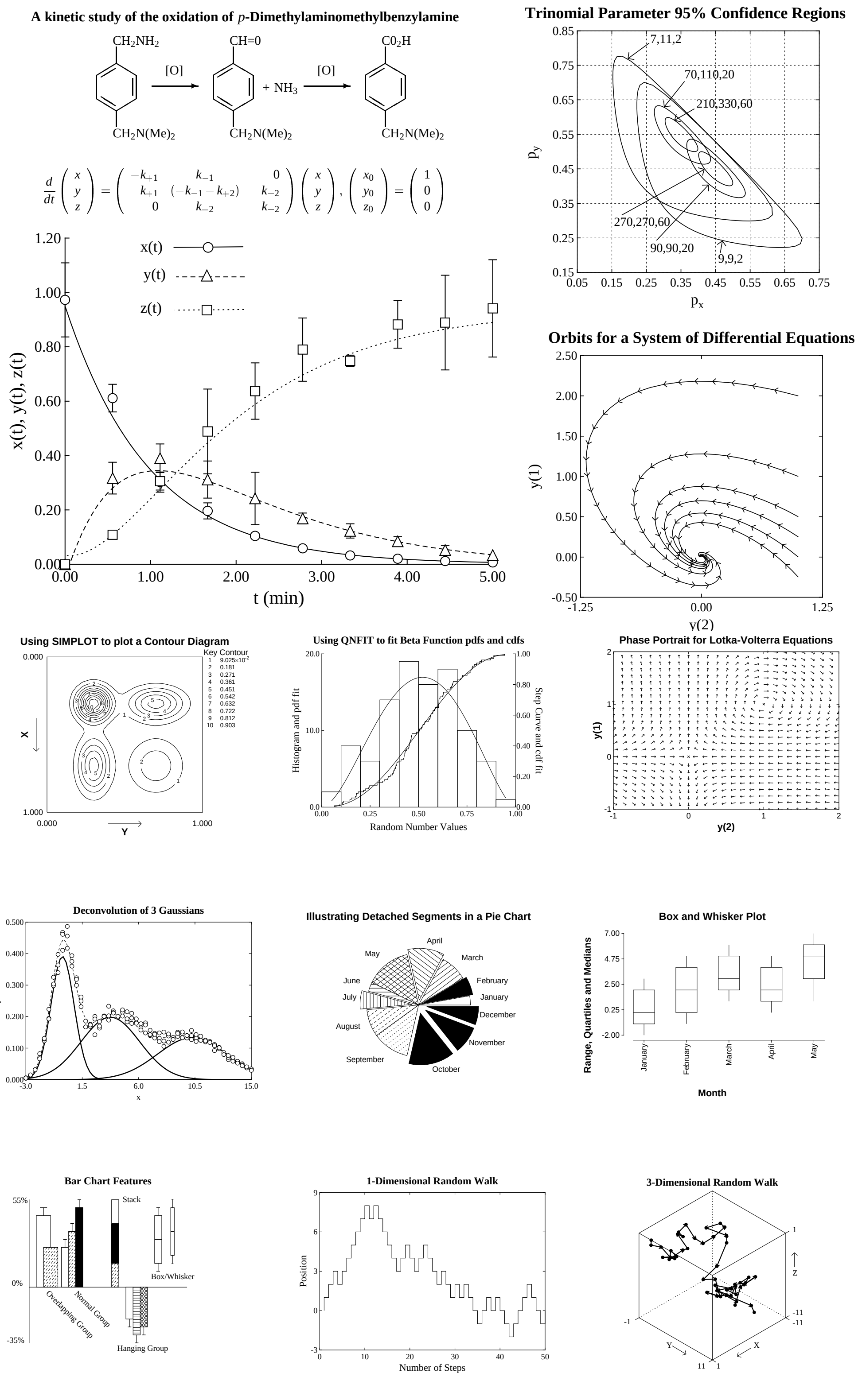
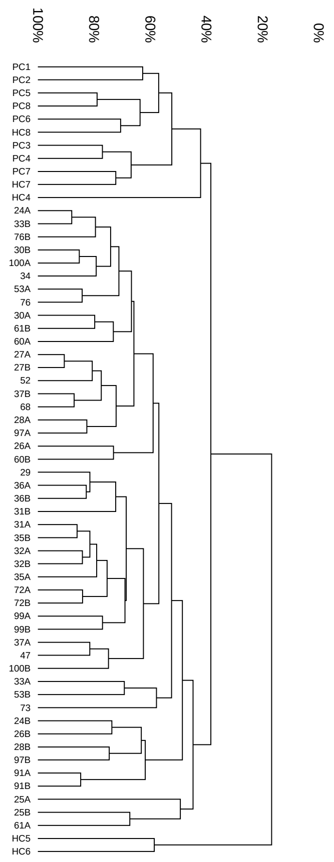
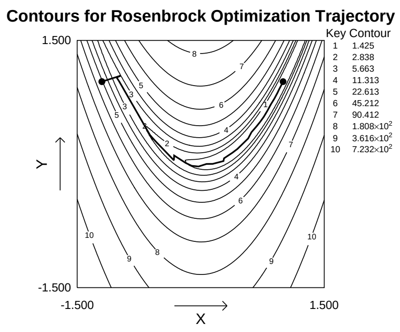
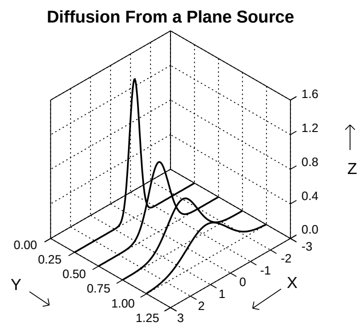


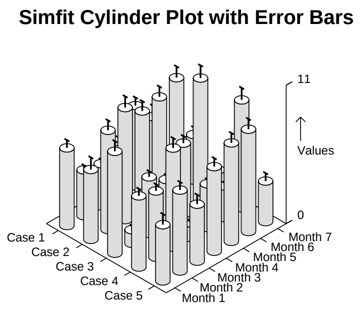
Figure 1.2: Collage 2

K-Means Clusters









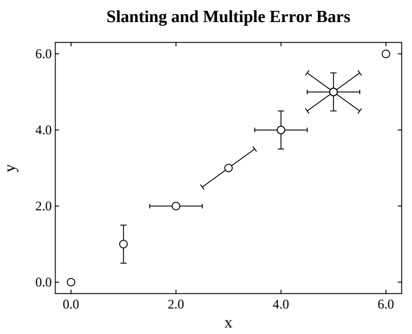


Figure 1.3: Collage 3

Part 2

First time user's guide

2.1 The main menu

The menu displayed in figure 2.1 will be referred to as the main SIMFIT menu.

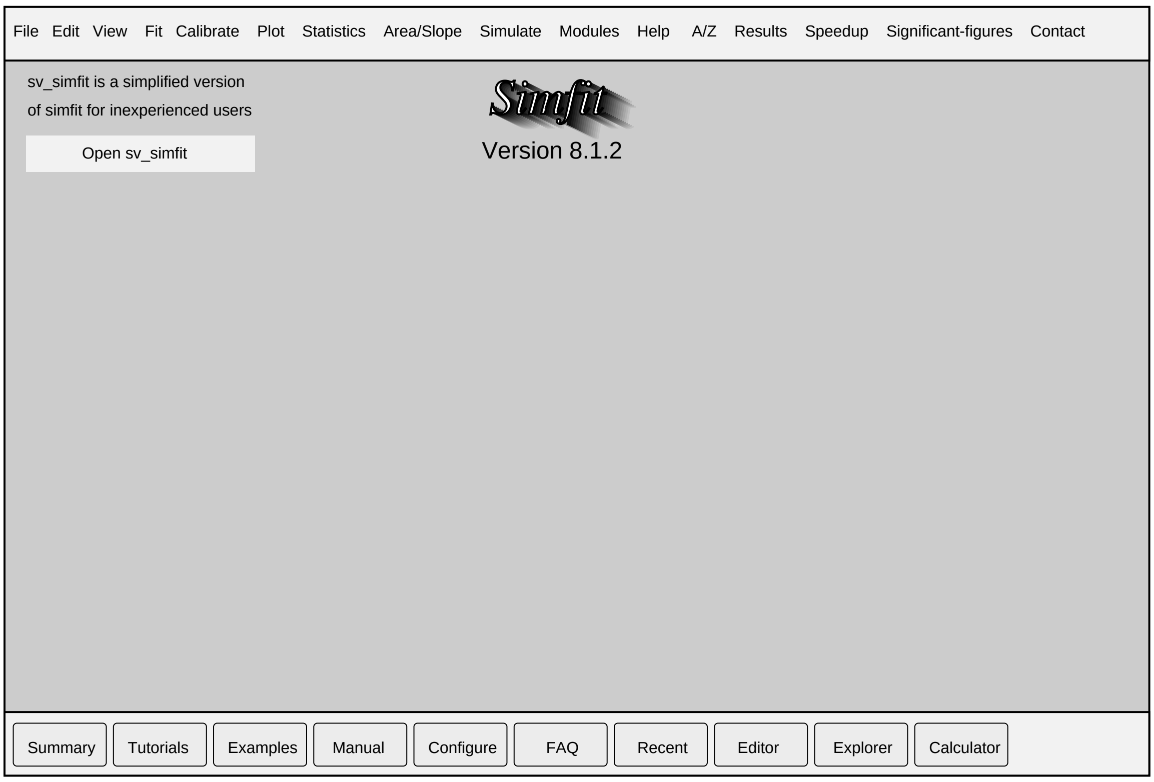

File Edit View Fit Calibrate Plot Statistics Area/Slope Simulate Modules Help A/Z Results Speedup Significant-figures Contact
sv_simfit is a simplified version
of simfit for inexperienced users
Open sv_simfit
Simfit
Version 8.1.2
Summary Tutorials Examples Manual Configure FAQ Recent Editor Explorer Calculator

Figure 2.1: The main SIMFIT menu

From this you can select from pop-up menus according to the functions required, and then choose which of the forty or so SIMFIT programs to use. When you run a program you will be in an isolated environment, dedicated to the chosen program. On exit from the chosen program you return to the main SIMFIT menu. If you get lost in sub-menus and do not know where you are, use the closure cross at the top right corner which appears when SIMFIT programs are active, unless the blank background window has been suppressed.

A brief description of the menus and task bar buttons will now be given.

`File` This option is selected when you want to create a data file by typing in your own data, or transforming clipboard data or text files with data tables from a spreadsheet. You can also define a set of data files for a library file.

`Edit` This option is selected when you want to edit a data file, or create a graphics file from a PostScript file.

`View` This option is selected when you want to view any ASCII text files, such as test files, data files, results files, model files, etc. A particularly useful feature is to be able to view lists of files analyzed and files created in the current session. Also, if program **GSview** or some other PostScript browser has been installed, you can view PostScript files, such as the SimFIT figures and manuals. Adobe Acrobat can also be used to view `*.pdf` files.

`Fit` From this option you can fit things like exponentials, binding models or growth curves, using dedicated user-friendly programs, or you can fit model-free equations, like polynomials or splines. Advanced users can do comprehensive curve fitting from libraries or user supplied equations.

`Calibrate` Choosing this option allows you to perform calibration using lines, polynomials (gentle curves), logistic polynomials (sigmoid curves), cubic splines (complicated curves), or deterministic models (if a precise mathematical form is required). You can also analyze dose response curves for minimum values, half saturation points, half times, IC50, EC50, or LD50 estimates.

`Plot` Some explanation is required concerning this option. All the SimFIT programs that can generate graphical display do so in such a way that a default graph is created, and there are limited options for editing. At this stage you can drive a printer or plotter or make a graphics file, but the output will only be of draft quality. To sculpture a graph to your satisfaction then obtain publication quality hardcopy, here is what to do: either transfer directly to advanced graphics or, for each data set or best-fit curve plotted, save the corresponding coordinates as ASCII text files. When you have such a set of files, you are ready to select the graph plotting option. Read the ASCII coordinate files into program **simplot** and edit until the graph meets your requirements. Then print it or save it as a graphics file. PostScript files are the best graphics files and, to re-size these, rotate, make collages, overlays, insets and so on, you input them into program **editps**.

`Statistics` SimFIT will do all the usual statistical tests, but the organization is very different from any statistics package. That is because SimFIT is designed as a teaching and research tool to investigate statistical problems that arise in curve fitting and mathematical modelling; it is not designed as a tool for routine statistical analysis. Nevertheless, the structure is very logical; there are programs designed around specific distributions, and there is a program to help you find your way around the statistics options and do all the usual tests. So, if you want to know how to do a chi-square or t-test, analyze a contingency table, perform analysis of variance or carry out nonparametric testing, just select program **simstat**. It tells you about tests by name or by properties, and it also does some very handy miscellaneous tasks, such as exhaustive analysis of a sample, multiple correlations and statistical arithmetic. For many users, the program **simstat** is the only statistics program they will ever need.

`Area/Slope` Many experimental procedures call for the estimation of initial rates, lag times, final asymptotes, minimum or maximum slopes or areas under curves (AUC). A selection of programs to do these things, using alternative methods, is available from this option.

`Simulate` If you want to simulate data for a Monte Carlo study, or for graph plotting, program **makdat** creates exact data from a library of models, or from a user-supplied model. Random error to simulate an experiment can then be added by program **adderr**. There is program **deqsol** for simulating and fitting systems of nonlinear differential equations, program **makcsa** for simulating flow cytometry experiments, and program **rannum** for generating pseudo random numbers.

`Modules` From this menu you can use your own specified editor, explorer, SIMFIT modules or, in fact, any chosen Windows program. There are specialized SIMFIT modules which can be accessed using this option.

`Help` From this menu you can run the SIMFIT help program and obtain technical data about the current release.

`A/Z` This provides a shortcut to named programs in alphabetical order.

`Results` This option allows you to view, print or save the current and ten most recent results files. Note that the default is `f\$result.txt`, but you can configure SIMFIT so that, each time you start a program, you can specify if the results should be stored on a named log file. All SIMFIT results files are formatted ready to be printed out with all tables tabbed and justified correctly for a monospaced font, e.g., Courier. However, at all stages during the running of a SIMFIT program, a default log file is created so that you can always copy selected results to the clipboard for pasting into a word processor. The main menu task bar also has buttons to let you view or print any ASCII text file, such as a data or results file.

`Speedup` This provides a shortcut to the speedup options described in the document `speedup.pdf`.

2.2 The task bar

At the bottom of the main SIMFIT menu will be found a task bar, which is provided to facilitate the interface between SIMFIT and other programs.

`Tutorials` This option allows you to read any chosen tutorial document as a small pdf file (usually 4 to 6 pages).

`Examples` This option allows you to read `w_examples.pdf`, the collected volume of tutorials and worked examples.

`Manual` This option allows you to open `w_manual.pdf`, the pdf version of the SIMFIT manual. The pdf manual has book-marks, and extensive hyperlinks between the contents, list of figures, and index, to facilitate on-line use. You should open the SIMFIT manual at the start of a SIMFIT session, then keep the manual open/minimized on the main Windows task bar, so that it is always ready for reference. The manual contains details of all the SIMFIT numerical procedures, statistical test theory, mathematical equations and analytical and plotting facilities, and has more details than the SIMFIT help program.

`Configure` This option starts up the SIMFIT configuration procedure. Use this to configure SIMFIT to your own requirements. Note that, if you select the `[Check]` option and SIMFIT reports missing files, you can specify files exactly or just provide a search path. The `[Apply]` option must be used to change the configuration by creating a new configuration file `w_simfit.cfg`.

`FAQ` This option allows you to run the frequently asked questions section of the SIMFIT help program which gives useful advice about the SIMFIT procedures but is not so comprehensive as the reference manual, which should be consulted for details, and to see worked examples.

`Recent` This option allows you to view, save, print, or copy to the clipboard your recent files analyzed, files created, or SIMFIT results files.

`Editor` This option opens your chosen text editor program, which you can specify using the `[Configure]` option. There are many excellent free text editors, such as **emacs**, which can be specified, and which are far more powerful than Windows Notepad. You should never edit any SIMFIT test file or model file and, to protect against this, experienced users could decide to make all the SIMFIT test files read-only.

`Explorer` This option opens your chosen disk explorer program, which you can specify using the `[Configure]` option.

`Calculator` This option opens your chosen calculator program, which you can specify using the `[Configure]` option.

Some things you can specify from this option are as follows.

❍ Switching on or off the displaying of start-up messages, or saving of results files.

❍ Suppressing or activating warning messages.

❍ Changing the size of fonts in menus, or the percentage of screen area used for plotting.

❍ Checking for consistent paths to auxiliary programs, and locating incorrectly specified files.

❍ Altering the specification of auxiliary programs and modules.

❍ Adjusting the variable colors on the SIMFIT color palette.

❍ Setting the default symbols, line-types, colors, fill-styles and labels for plotting.

❍ Defining mathematical constants that are used frequently for data transformation.

2.3 The file selection control

The SIMFIT file dialogue control is displayed in figure 2.2.

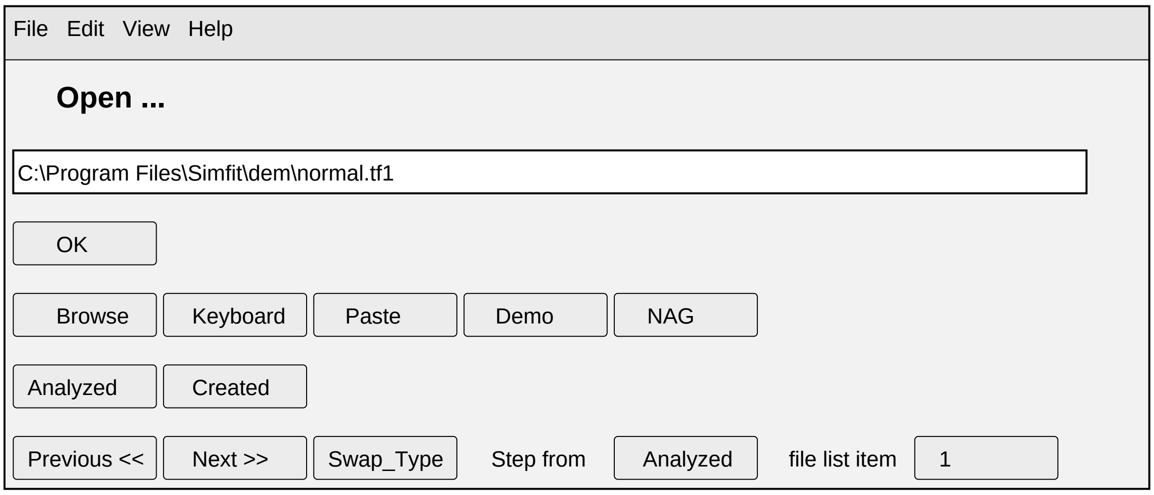


Figure 2.2: The SIMFIT file selection control

This control helps you to create new files (i.e., in the Save As . . . mode) or analyze existing files (i.e., in the Open . . . mode). The top level `[File]`, `[Edit]`, `[View]`, and `[Help]` menus allow you to select appropriate test files to use for practise or browse to understand the formatting. Below this is an edit box and a set of buttons which will now be described.

`File Name` You can type the name of a file into the edit box but, if you do this, you must type in the full path. If you just type in a file name you will get an error message, since SimFIT will not let you create files in the SimFIT folder, or in the root, to avoid confusion.

`OK` This option indicates that the name in the edit box is the file name required.

`Browse` This option simply transfers you to the Windows control but, when you know how to use the SimFIT file selection control properly, you will almost never use the Windows control.

`Keyboard` This option allows you to type in a data set from the keyboard. It is only useful if you understand the file format required and have very small samples. It creates a temporary file in your usr folder.

`Paste` This option is only activated when SimFIT detects ASCII text data on the clipboard and, if you choose it, then SimFIT will attempt to analyze the clipboard data. If the clipboard data are correctly formatted, SimFIT will create a temporary file, which you can subsequently save if required. If the data are not properly formatted, however, an error message will be generated. When highlighting data in your spreadsheet to copy to the clipboard, write to a comma delimited ASCII text file, or use a with a macro like `simfit6.xls`, you must be very careful to select the columns for analysis so that they all contain exactly the same number of rows.

`Demo` This option provides you with a set of test files that have been prepared to allow you to see SimFIT in action with correctly formatted data. Obviously not all the files displayed are consistent with all the possible program functions. With programs like **simstat**, where this can happen, you must use the [Help] option to decide which file to select. When you use a SimFIT program for the first time, you should use this option before analyzing your own data.

`NAG` This option provides you with a set of test files that have been prepared to allow you to use SimFIT to see how to use NAG library routines.

`Analyzed` This history option allows you to choose from a list of the last files that SimFIT has analyzed, but the list does not contain files recently saved.

`Created` This history option allows you to choose from a list of the last files that SimFIT has created, but the list does not contain files recently analyzed. Of course, many files will first be created then subsequently analyzed, when they would appear in both Analyzed and Created lists.

`Previous` This option allows you to scroll backwards through recent files and edit the filename, if required, before selecting.

`Next` This option allows you to scroll forwards through recent files and edit the filename, if required, before selecting.

`Swap Type` This option toggles between Created and Analyzed file types.

If you name files sensibly, like results.1, results.2, results.3, and so on, and always give your data short meaningful titles describing the data and including the date, you will find the [Back], [Next], [Created] and [Analyzed] buttons far quicker and more versatile than the [Browse] pipe to Windows.

2.3.1 Multiple file selection

It often happens that users need to select multiple files. Examples could be:

- ❏ collecting a set of graphics files together in order to create a composite graph in **simplot**;
- ❏ selecting a family of vector files for analysis of variance or correlation analysis;
- ❏ building a consistent package of PostScript files to generate a collage using **editps**, or
- ❏ gathering together results files for fitting several model functions to experimental data in **qnfit**.

The problem with the Windows multiple file selection protocol is that it does not offer a convenient mechanism for selecting subsets from a pool of related files, nor does it provide the opportunity to select files of restricted type, vector files only, for instance. The SimFIT library file method is the best technique to submit a selected set of files for repeated analysis, but it is not so versatile if users want to add or subtract files to a basic set interactively, which requires the project archive technique.

2.3.1.1 The project archive technique

The SimFIT project archive technique has been developed to meet these needs. Where the multiple selection of files is called for, a menu is presented offering users the opportunity to input a library file, or a set of individually chosen files, or to initiate a project.

The project archive technique provides these opportunities:

- ❍ choosing individual files by the normal procedure;
- ❍ selecting files by multiple file selection using the shift and control keys;
- ❍ deleting and restoring/moving individual files;
- ❍ suppressing multiple files from the project, or
- ❍ harvesting files from a project list.

2.3.1.2 Checking and archiving project files

Before files are accepted into a project, a quick check is undertaken to ensure that files are consistent with the type required. Further, users are able to add files to project lists interactively after files have been created, e.g., after saving ASCII text coordinate files to replot using **simplot**. The project archives for recent files are as follows:

`a_recent.cfg`:	any type of file
`c_recent.cfg`:	covariance matrix files
`f_recent.cfg`:	curve fitting files
`g_recent.cfg`:	graphics ASCII coordinate files
`m_recent.cfg`:	matrix files for statistics
`p_recent.cfg`:	encapsulated PostScript files
`v_recent.cfg`:	vector files for statistics.

Files added to a project archive are kept in the order of addition, which sometimes permits duplication but keeps files grouped conveniently together for multiple selection. Search paths and file types can be set from the normal file selection control and missing files are deleted from the archives.

2.4 First time user's guide to data handling

Data must be as tables of numerical data (with no missing values) in ASCII text format, as will be clear by using the `[View]` button on the main SimFIT menu to browse the test files. Such files can be created using any text editor but are best made by using the SimFIT editors, or transferring data from a spreadsheet using the clipboard and **maksim**, or a macro such as `simfit6.xls`. First observe the notation used by SimFIT when creating data tables. Scientific/computer notation is used for real numbers, where E+mn means 10 to the power mn. Examples: 1.23E-02 = 0.0123, 4.56E+00 = 4.56 and 7.89E+04 = 78900.0. This notation confuses non-scientists and inexperienced computer users, but it has the advantage that the numbers of significant figures and orders of magnitude can be seen at a glance. However, correlation coefficients and probabilities are output in decimal notation to four decimal places, which is about the limit for meaningful significance tests, while integers are usually displayed as such. Note that formatting of input data is not so strict: you can use any formatting convention to represent numbers in your own data tables, as SimFIT converts all data supplied into double precision numbers for internal calculations. For instance: 1, 1.0, 1.0E+00, 10.0E-01, and 0.1E+01 are all equivalent. Either commas or spaces can be used to separate numbers in input lists, but commas must not be used as decimal points in data files. For instance: 1 2 3 and 1,2,3 are equivalent.

2.4.1 The format for input data files

The SimFIT data file format is as follows.

a) Line 1: Informative title for the data set ($\leq$ 80 characters)
b) Line 2: Number of rows (m) Number of columns (n) (dimensions of the data set)
c) Lines 3 to $m + 2$: Block of data elements (as a m by n matrix)
d) Line $m + 3$: Number of further text lines (k)
e) Lines $m + 4$ to $m + 3 + k$: extra text controlling program operation or describing the data.

2.4.2 File extensions and folders

SimFIT does not add file extensions to file names, nor is it sensitive to the data file extensions. So you should use the extension `.txt` if you want your text editor or word processor to create, or read SimFIT files. However, you will be prompted to use accepted file extensions (`.eps`, `.jpg`, `.bmp`) for graphics files, and SimFIT will refuse to open executable files (`.exe`, `.dll`, `.bat`, `.obj`, `.com`), or create files in the root directory (e.g., `C:`), or the SimFIT folder (e.g., `C:\Program Files\Simfit`). Data files should be given meaningful names, e.g., `data.001`, `data.002`, `data.003`, `data.004`, `first.set`, `second.set`, `third.set`, `fourth.set`, etc., so that the names or extensions are convenient for copying/deleting.

2.4.3 Advice concerning data files

a) Use an informative title for your data and include the date of the experiment.
b) The first extra text line controls some programs, e.g., **calcurve** in expert mode, but most programs ignore the extra text. This is where you enter details of your experiment.
c) If you enter a vector into programs **makmat/editmt**, do not rearrange into increasing or decreasing order if you wish to do run, paired t or any test depending on natural order.
d) With big data sets, make small files (**makfil/makmat**), then join them together (**editfl/editmt**).

2.4.4 Advice concerning curve fitting files

a) Use **makfil** to make a main master file with x and y values and with all $s = 1$. Keep replicates in the natural order in which they were made to avoid bias in the run test.
b) Enter all your data, not means of replicates, so the run and sign tests have maximum power and the correct numbers of degrees of freedom are used in statistics. If you do not have sample variances from replicates, try an appropriate multiple of the measured response (7% ?) as a standard deviation estimate. Nothing is saved by using means and standard errors of means but, if you do this, the parameter estimates will be alright, but statistics will be biased.

c) To change values, units of measurement, delete points, add new data, change weights, etc., input the main master file into **editfl**.
d) If you have single measurements (or < 5 replicates ?), fit the main master file with all $s = 1$ and compare the result with $s = 7\%|y|$ say, obtained using **editfl**.
e) If you have sufficient replicates (≥ 5 ?) at each x, input the main master file into **editfl** and generate a file with s = sample standard deviations. Compare with results from, e.g., smoothed weighting.
f) For files with means and std. dev., or means ± 95% confidence limits for error bars, use **editfl** on the data, or generate interactively from replicates using the graphics `[Advanced]` option.

2.4.5 Example 1: Making a curve fitting file

Select **makfil** and request to create a file, say `fivexy.1st`, containing five pairs of x, y values and choosing to set all $s = 1$. When asked, give the data an informative title such as `...Five x,y values...`, and then proceed to type in the following five x, y values (which contain a deliberate mistake).

x y
1 1
2 2
3 3
4 5
5 4

When finished request a graph, which will clearly show the mistake as in the dotted line in figure 2.3, namely, that the y values are reversed at the last two x values. Of course you could correct the mistake at this stage but, to give you an excuse to use the curve fitting file editor, you should now save the file `fivexy.1st` in its present form.

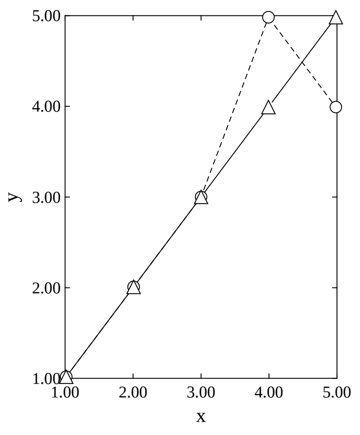


Figure 2.3: Five x,y values

2.4.6 Example 2: Editing a curve fitting file

Read the file, `fivexy.1st`, that you have just created into **editfl**, and ask to create a new file, say `fivexy.2nd.` Then change the values of y at lines 4 and 5 so that the y values are equal to the x values. Now you should see the perfectly straight continuous line as in figure 2.3 instead of the bent dotted line. You are now finished, but before you exit please note some important features about program **editfl**.

1) This editor takes in an old file which is never altered in any way.
2) After editing to change the data and title, a new file is created with the edited data.
3) In this way your original data are never lost, and you can always delete the original file when you are sure that the editing is correct.
4) There are a vast number of powerful editing options, such as fusing files, changing baselines, scaling into new units, weighting, creating means and standard errors or error bar files from groups of replicates.

2.4.7 Example 3: Making a library file

Select **maklib** and make a library file, say `mylib.1st`, with the title `...Two data sets...` and containing the two files you have just made. Browse this file in the SIMFIT file viewer and you will discover that it looks like the following:

```
Two data sets
fivexy.1st
fivexy.2nd
```

You could have made this file yourself with the **edit** or **notepad** Windows tools, as it is just a title and two filenames. Now, to appreciate the power of what you have just done, select program **simplot**, choose a standard x, y plot and read in this library file, `mylib.1st`, to get a plot like figure 2.3.

2.4.8 Example 4: Making a vector/matrix file

Select **makmat** and request to make a vector file called, for instance, `vector.1st`, with a title, say `...Some numbers between 0 and 1...`, then type in ten numbers between 0 and 1. For example:

```
0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5
```

Save this file then make another one called, for example, `vector.2nd` with the numbers

```
0.975, 0.95, 0.925, 0.9, 0.85, 0.8, 0.75, 0.7, 0.6, 0.5
```

We shall use these two vector files later to practise statistical tests.

2.4.9 Example 5: Editing a vector/matrix file

Read in the file called `fivexy.1st` which you made previously and do the same editing that you did with **editfl** to correct the mistakes. Now you will be able to appreciate the similarities and differences between **makfil/editfl** and **makmat/editmt**: **makfil/editfl** have dedicated functions to handle files with column 1 in increasing order and column 3 positive, while **makfil/editfl** can handle arbitrary matrices and vectors.

2.4.10 Example 6: Saving data-base/spread-sheet tables to files

Since spread-sheet and data-base programs can write out tables in ASCII text format, it is easy to transform them into SIMFIT style. For instance, read `fivexy.2nd` into **maksim** and, after discarding the two header and trailer lines, you can create a data file in SIMFIT format. **maksim** is much more than just a utility for re-formatting, it can also do selections of sub-sets of data for statistical analysis according to the following rules.

a) Only hard returns on the input file can act as row separators.
b) Non-printing characters, except hard returns, act as column separators.
c) Spaces or commas are interpreted as column separators and double commas are interpreted as bracketing an empty column.
d) Each row of the input table is regarded as having as many columns as there are words separated by commas or spaces.
e) Commas must not be used as decimal points or as thousands separators in numbers. For example, use 0.5 (not 0,5 or 1/2) for a half, and 1000000 (not 1,000,000) for a million.
f) Single column text strings must be joined and cannot contain spaces or commas. For example, use strings like "Male.over.40" (not "Male over 40") for a label or cell entry.
g) Simple tables of numbers can be entered directly, but titles, row and column counters, trailing text and the like must be deleted until every row has the same number of columns before selection of sub-matrices can commence.
h) Row and column entries can be selected for inclusion in an output file as long as both the row and column Boolean selection criteria are satisfied. To achieve this it is often best to start by globally suppressing all rows and columns and then including as required, e.g., columns 3 and 4, all rows with Smith in column 1, all rows with values between 40 and 60 in column 2.

In order to understand the functionality provided by **maksim** you should create some tables using a text editor such as Windows **notepad** then copy to the clipboard and read into **maksim**. There are also special test files, `maksim.tf1`, `maksim.tf2`, `maksim_1.html`, and `maksim_2.html` that are designed to exploit and illustrate some of the the procedures available in **maksim**.

Note that program **maksim** does not allow editing, for that you use your text editor. It does have a useful visual interface for browsing smallish ASCII text tabular data, so that you can see at any stage what the sub-matrix of selected data looks like. Like all SIMFIT editors, it will never discard or overwrite your primary data file.

2.5 First time user's guide to graph plotting

There are three basic graphical operations as follows.

1) Obtaining a set of coordinates to be plotted.
Every SimFIT program that creates graphs lets you print a default graph, or save ASCII coordinate files, which are tables of coordinates. This phase is easy, but name files systematically.

2) Transferring the graph to a peripheral.
If you have a PostScript printer, use the SimFIT driver not the Windows driver. If not, drive your printer directly or, for more options, use PostScript output with program **GSview**.

3) Including graphics files in documents.
Bitmap files (`*.bmp`) are fine for photographs and histology slides, but are inappropriate for scientific graphs, since they are large and give poor resolution when printed. Vector files (e.g., PostScript files) are better as they are compact, easily edited, and give publication quality hardcopy. Windows word processing packages can import PostScript files but, to view them on screen, you may have to use program **GSview** to add a preview section. The best graphics files for Windows users with no PostScript facilities are either *.png or *.svg

2.5.1 The SimFIT simple graphical interface

The SimFIT simple graphical interface, displayed in figure 2.4 for data in `gcfit.tf2`, best fit logistic curve,

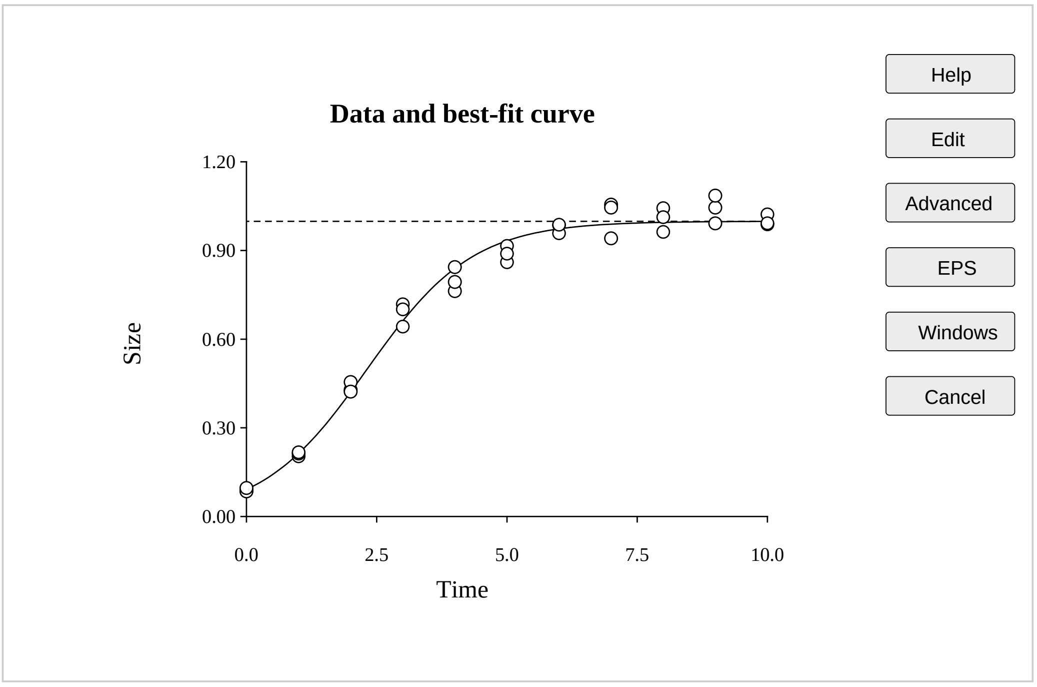


Figure 2.4: The SimFIT simple graphical interface

and asymptote obtained using **gcfit**, provides the options now described.

`Help`	This provides a short tutorial about SimFIT graphics.
`Edit`	This provides only very simple options for editing the graph.

`Advanced` This lets you create ASCII coordinate files, which can be added to your project archive for retrospective use, and is the most powerful option in the hands of experienced users. Alternatively, you can transfer directly into **simplot** for immediate editing.

`EPS` This creates a temporary PostScript file which can be viewed, saved to file, printed using **GSview**, or copied directly to a PostScript printer, for the highest possible quality.

`Windows` This is provided for users who do not have PostScript facilities. Only two of the options should be contemplated; *.png or *.svg.

`Cancel` This returns you to the executing program for further action.

2.5.2 The SIMFIT advanced graphical interface

The SIMFIT advanced graphical interface, displayed in figure 2.5, results from transferring the data from

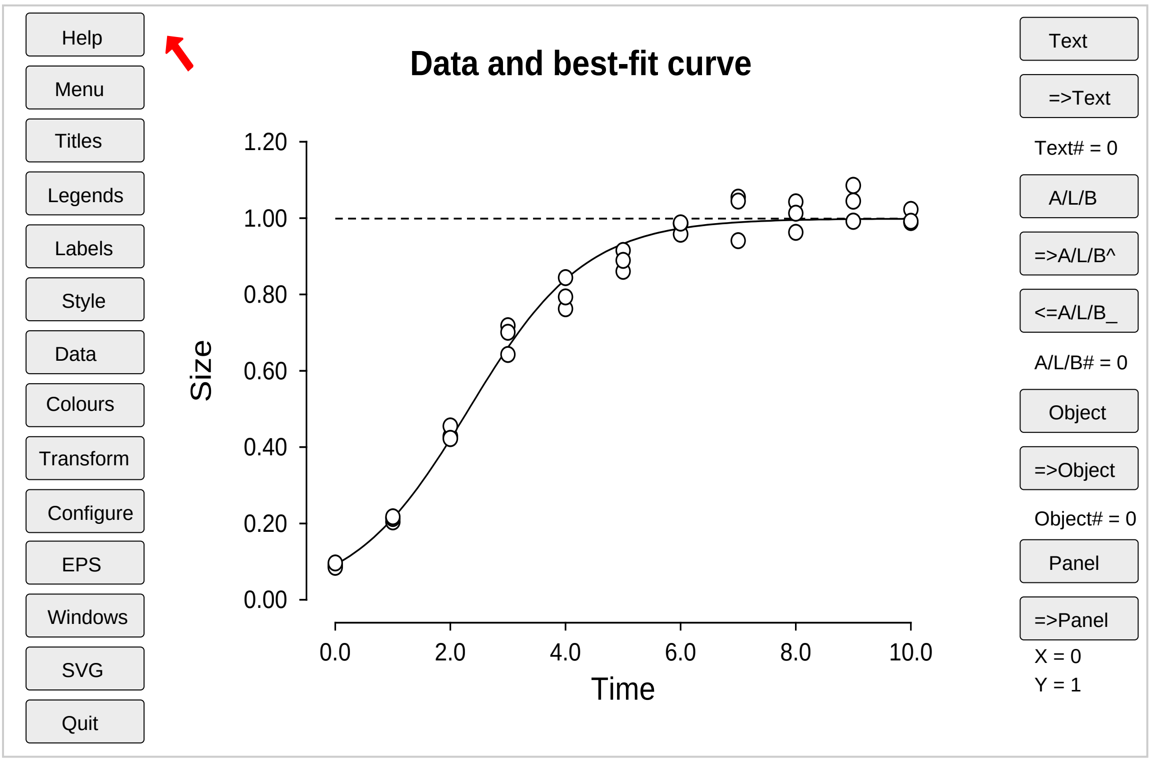


Figure 2.5: The SIMFIT advanced graphical interface

figure 2.4 directly into **simplot**, providing further options, as now described.

The advanced graphics left hand options

`Help` This provides help details that can also be obtained by a right mouse click within the plotting area.

`Menu` This provides for editing from a menu without having to redraw the graph after each edit, and is designed to save time for advanced users plotting many large data sets.

`Titles` This allows you to edit the plot titles.

`Legends` This allows you to edit the plot legends.

`Labels` This allows you to change the range of data plotted, and alter the number or type of tick marks and associated labels.

`Style` This allows you to alter the aspect ratio of the plot and perform clipping. A graph paper effect can be added to aid the placement of graphical objects, and offsets or frames can be specified.

`Data` This allows you to change line or symbol types, add or suppress error bars, edit current data values, add new data sets, or save edited data sets.

`Colors` This allows you to specify colors.

`Transform` This allows you to specify changes of coordinates. Note that there is a specific title and set of plot legends for each transformation, so it makes sense to choose a transformation before editing the titles or legends.

`Configure` This allows you to create configuration files containing all the details from editing the current plot, or you can read in existing configuration files from previous editing to act as templates.

`EPS` PostScript hardcopy, as for the PostScript option with simple graphics.

`Windows` Windows hardcopy, as for the Windows option with simple graphics.

`SVG` Advanced SVG editing including stretching to change aspect ratios.

`Quit` This prompts you to save a configuration file, then closes down the current graph.

The advanced graphics right hand options

`Text` This allows you to select a text string to label features on the graph.

`T = #` This indicates which text string is currently selected. Only one string can be selected at any time.

`A/L/B` This allows you to select an arrow, line, or box to label features on the graph.

`A = #` This indicates which arrow, line, or box is currently selected. Only one arrow, line, or box can be selected at any time.

`Object` This allows you to select a graphical object to label features of the graph.

`O = #` This indicates which graphical object is currently selected. Only one graphical object can be selected at any time.

`Panel` This allows you to specify an information panel linking labels to line types, symbols, fill-styles, etc., to identify the data plotted.

`X = #` This is X the coordinate for the current hot spot.

`Y = #` This is the Y coordinate for the current hot spot.

Moving graphical objects

The right hand buttons in figure 2.5 allow you to move graphical objects about. The way this works is that the red arrow can be dragged anywhere on the graph, and its tip defines a hot spot with the coordinates just discussed. This hot spot is coupled to the current, text, arrow, line, box, graphical object that has been selected and also to the left hand buttons. Help and hardcopy are also controlled by left hand buttons. Note that the appropriate right hand buttons must be used to make a specific text string, arrow, line, box, or graphical object the selected one before it can be coupled to the hot spot. Also, observe that, to drag a horizontal outline box in order to surround a text string, the head and tail moving buttons are coupled to opposite corners of the horizontal rectangle.

2.5.3 PostScript, GSview and SIMFIT

SIMFIT is now distributed with the Ghostscript dlls so it is no longer necessary to install the Ghostscript package. EPS standard PostScript files are created in a special format to facilitate introducing maths, or symbols like pointing hands or scissors (ZapfDingbats) retrospectively. Note the following advice.

a) The default display uses TrueType fonts which are not exactly the same dimensions as PostScript fonts, so text strings on the display and in the PostScript hardcopy will have identical starting coordinates, orientation, and color, but slightly differing sizes. Before making PostScript hardcopy, check the PostScript display to make sure that the text strings are not under- or over-sized.
b) To get the full benefit from PostScript install the **GSview** program which drives all devices from PostScript files.
c) Save ASCII coordinate files from the default plot or transfer directly into **simplot**.
d) Collect your selected ASCII coordinate files into a set and input them individually or collectively (as a library file or project archive) into program **simplot**.
e) Edit the graph on screen until it is in the shape and form you want, using the ordinary alphabet and numbers at this stage where you want Greek, subscripts, superscripts, etc.
f) Now edit to change characters into subscripts, Greek, maths, as required, or add accents like acute, grave, tilde, and finally save or print.
g) With the file, you can make slides, posters, incorporate into documents, create hardcopy of any size or orientation (using program **editps**), transform into another format, view on screen, drive a non-PostScript printer, etc.

The SIMFIT PostScript driver interface

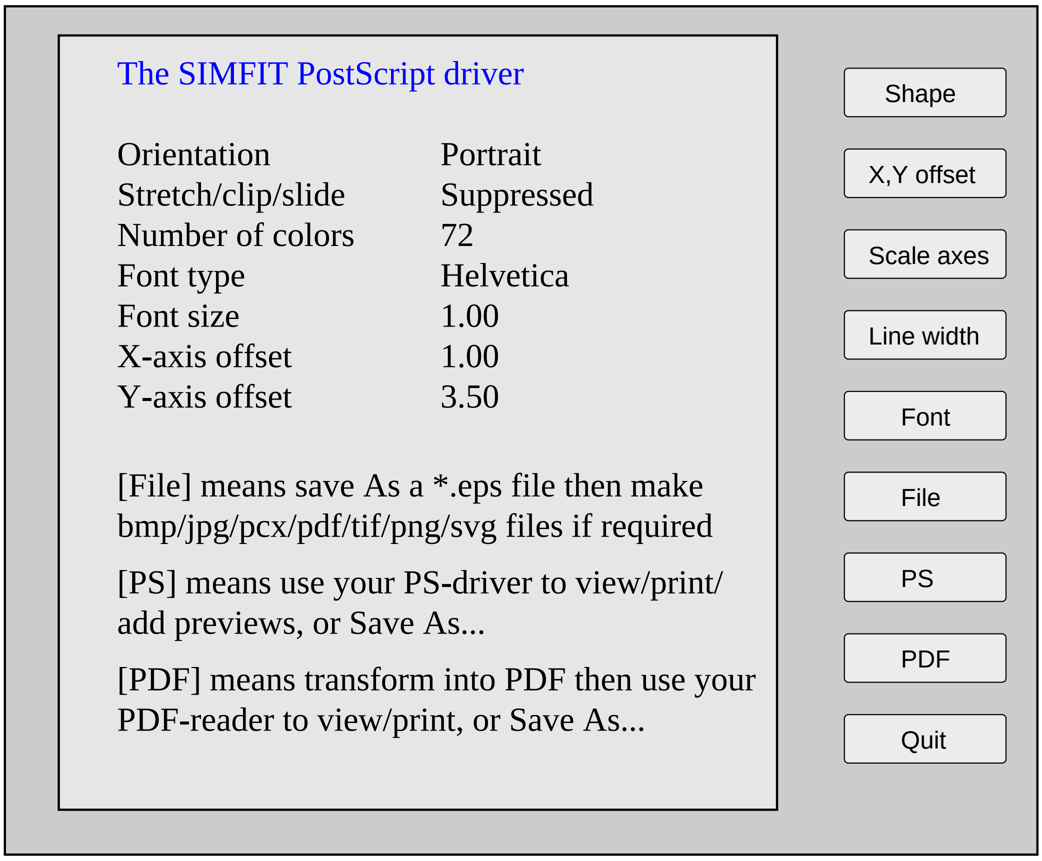
The SIMFIT PostScript driver interface, displayed in figure 2.6, which can be used from either the simple or advanced graphics controls, provides the options now described. x

`Shape` This allows you to switch between portrait or landscape, but it also offers enhanced options where you can stretch, slide or clip the PostScript output without changing the aspect ratio of the fonts or plotting symbols. This is very useful with crowded graphs such as dendrograms, as explained further on page 210, or with map plotting, as discussed on page 218 where aspect ratios have to be altered so as to be geometrically correct.

`X,Y-offset` This allows you to set a fixed offset if the defaults are not satisfactory, but it is better to leave the defaults and edit retrospectively using **editps**.

`Scale axes` This allows you to set a fixed scaling factor if the defaults are not satisfactory, but it is better to leave the defaults and edit retrospectively using **editps**.

`Line width` This allows you to set a fixed line width for all hardcopy, i.e. both Windows and PostScript,


The SIMFIT PostScript driver

Orientation	Portrait
Stretch/clip/slide	Suppressed
Number of colors	72
Font type	Helvetica
Font size	1.00
X-axis offset	1.00
Y-axis offset	3.50

[File] means save As a *.eps file then make bmp/jpg/pcx/pdf/tif/png/svg files if required

[PS] means use your PS-driver to view/print/ add previews, or Save As...

[PDF] means transform into PDF then use your PDF-reader to view/print, or Save As...

Figure 2.6: The SɪᴍFɪT PostScript driver interface

if the default is not satisfactory. However, note that the relative line widths are not affected by this setting, and if extreme line widths are selected, SɪᴍFɪT will re-set line widths to the defaults on start up.

`Font` This allows you to set the default font type, which would normally be Helvetica for clarity, or Helvetica Bold for presentation purposes such as slides.

`File` This allows you to create a PostScript file, but it is up to you to add the extension `.eps` to indicate that the file will be in the encapsulated PostScript format with a BoundingBox.

`EPS` This allows you to visualize the PostScript file using your PostScript viewer, which would normally be **GSview**. If GSview is not configured the pdf equivalent will be displayed.

`PDF` This allows you to copy the PostScript file (as PDF) directly to your PDF-reader.

`Quit` This returns you to the executing program for further action.

Note that all the parameters controlling PostScript output are written to the file `w_ps.cfg`, and a detailed discussion of SɪᴍFɪT PostScript features will be found in the appendix (page 366).

2.5.4 Example 1: Creating a simple graph

From the main SIMFIT menu select [Plot], then **simplot**, and choose to create a graph with standard *x*, *y* axes. Input the library file `w_simfig1.tfl` which identifies `simplot.tf1`, `simplot.tf2`, and `simplot.tf3`, which now display as figure 2.7. Here, **simplot** has used defaults for the title, legends, plotting symbols, line types and axes. This is how **simplot** works. Every graphical object such as a data set, a file with error bars, a best-fit curve, etc. must be contained in a correctly formatted ASCII plotting coordinates text file. Normally these would be created by the SIMFIT programs, and you would make a library file with the objects you want to plot. Then you would choose the shape, style, axes, titles, plotting symbols, line-types, colors, extra text, arrows, accents, math symbols, Greek, subscripts, superscripts and so on and, when the plot is ready, a printer would be driven or a graphics file created. Finally, before quitting the graph, a configuration file would be written for re-use as a template. To see how this works, read in the configuration file `w_simfig1.cfg` to get figure 2.8, while with `w_simfig2.cfg` you will get the corresponding Scatchard plot.

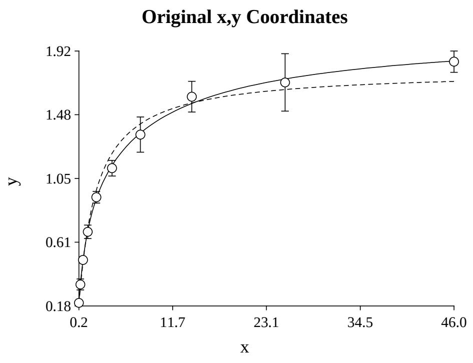


Figure 2.7: The **simplot** default graph

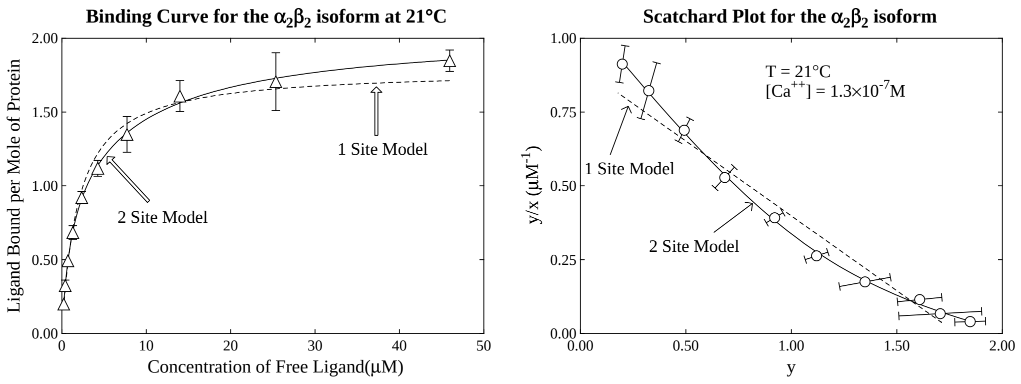


Figure 2.8: The finished plot and Scatchard transform

2.5.5 Example 2: Error bars

Figure 2.8 has three objects; means and error bars in `simplot.tf1`, with best fit curves for two possible models in `simplot.tf2`, and `simplot.tf3`. Actually, you can make a file like `simplot.tf1` yourself. The file `mmfit.tf4` contains curve fitting data, and you make a file with means and 95% confidence limits from this using program **editfl**. The procedure used in SIMFIT is always to do curve fitting using all replicates, not means. Then, when a plot is needed, a file with error bars is generated. To see how this is done, choose the [Edit] option from the main menu, read `mmfit.tf4` into program **editfl**, then select the option to create a file with means and error bars for plotting and create an error bar file like `simplot.tf1` from replicates in `mmfit.tf4`. This illustrates a very important principle in SIMFIT.

You never input data consisting of means from replicates into SIMFIT programs. So, if you calculate means from groups of replicates yourself, you are doing something wrong, as SIMFIT always performs analysis using complete data sets. For instance, means with error bars for plotting can always be calculated on demand from replicates (arranged in nondecreasing order), e.g., using the [Data] option in program **simplot**.

2.5.6 Example 3: Histograms and cumulative distributions

To illustrate histograms we will use `normal.tf1`, with fifty random numbers from a normal distribution ($\mu = 0, \sigma = 1$), generated by program **rannum**. Choose `[Statistics]` from the main menu, then **simstat** and pick the option to test if a sample is from a normal distribution. Read in `normal.tf1`, create a histogram with twelve bins between −3.0 and 3.0, then display the plots as in figure 2.9.

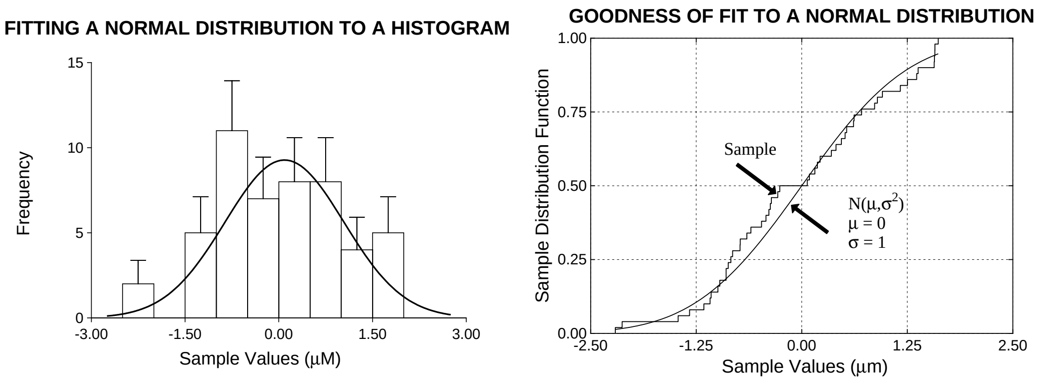


Figure 2.9: A histogram and cumulative distribution

A best fit *pdf* curve can also be created using **qnfit** and a *pdf* file, e.g., with error bars calculated using a binomial distribution as in figure 2.9 Note that, whereas the histogram is easy to interpret but has an ambiguous shape, the cumulative distribution has a fixed shape but is featureless. There is much to be said for showing best fit *pdfs* and *cdfs* side by side as in figure 2.9 since, in general, statistical tests for goodness of fit of a *pdf* to a distribution should be done on the cumulative distribution, which does not suffer from the ambiguity associated with histograms. However trends in data are more easily recognized in histograms from large samples than in cumulative distributions, i.e., stair step plots.

2.5.7 Example 4: Double graphs with two scales

Frequently different scales are required, e.g., in column chromatography, with absorbance at 280*nm* representing protein concentration, at the same time as enzyme activity eluted, and the pH gradient. Table 2.1 is typical where absorbance could require a scale of zero to unity, while enzyme activity uses a scale of zero to

Fraction Number	*Absorbance*	*Enzyme Activity*	*Buffer pH*
1	0.0	0.1	6.0
2	0.1	0.3	6.0
3	1.0	0.2	6.0
4	0.9	0.6	6.0
5	0.5	0.1	6.2
6	0.3	0.8	6.7
7	0.1	1.5	7.0
8	0.3	6.3	7.0
9	0.4	8.0	7.0
10	0.2	5.5	7.0
11	0.1	2.0	7.2
12	0.1	1.5	7.5
13	0.3	0.5	7.5
14	0.6	1.0	7.5
15	0.9	0.5	7.5

Table 2.1: Data for a double graph

eight, and pH could be on a scale of six to eight. If absorbance and activity were plotted on the same scale

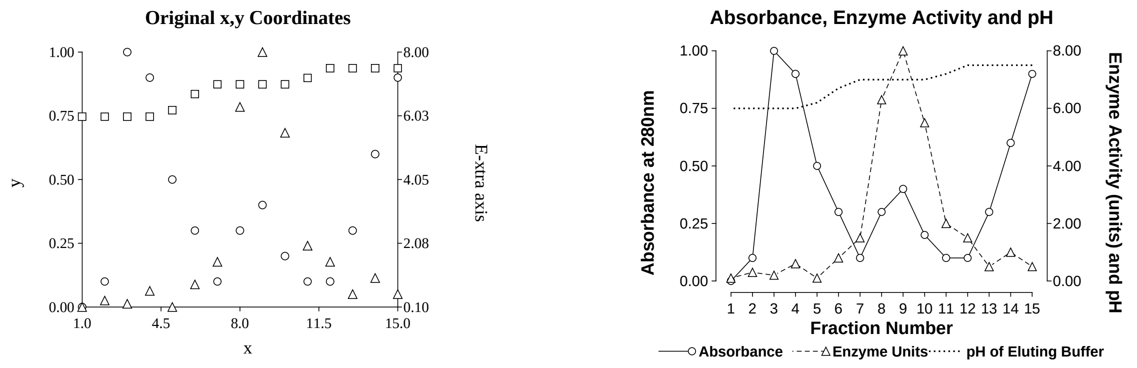


Figure 2.10: Plotting a double graph with two scales

the plot would be dominated by activity, so you could change the units of enzyme activity to be compatible with the absorbance scale. However, to illustrate how to create a double graph, a plot with absorbance on the left hand axis and enzyme activity and pH together on the right hand axis will be constructed. Obviously this requires three separate objects, i.e., files for program **simplot**. You could create the following files using program **makmat**, and the data in table 2.1.
File 1: The first column together with the second column (as in `plot2.tf1`)
File 2: The first column together with the third column (as in `plot2.tf2`)
File 3: The first column together with the fourth column (as in `plot2.tf3`)

Select program **simplot** and choose to make a double graph. Input the first file (absorbance against fraction) scaled to the left hand axis with the other two scaled to the right hand axis to get the left panel of figure 2.10. To transform the left plot into the finished product on the right panel in figure 2.10 proceed as follows:

a) Edit the overall plot title and both plot legends.
b) Edit the data ranges, notation and offset on the axes.
c) Edit the three symbol and line types corresponding to the three files.
d) Include an information panel and edit the corresponding keys.
e) Choose HelveticaBold when creating the final PostScript file.

2.5.8 Example 5: Bar charts

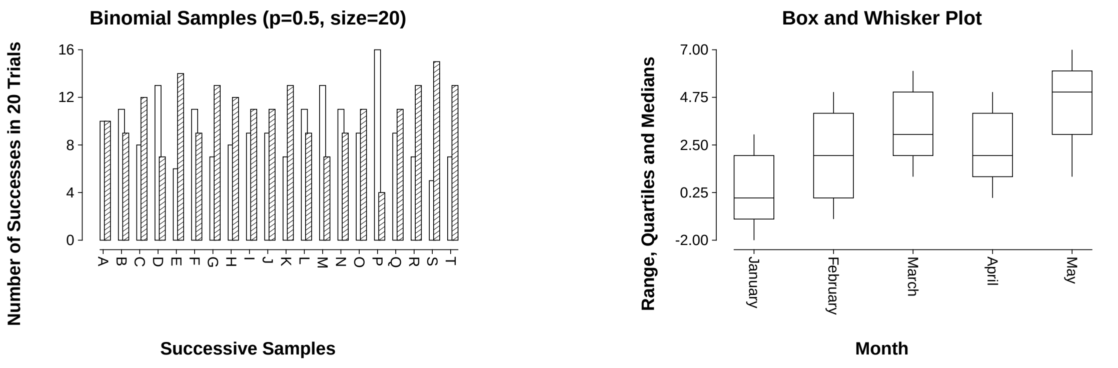
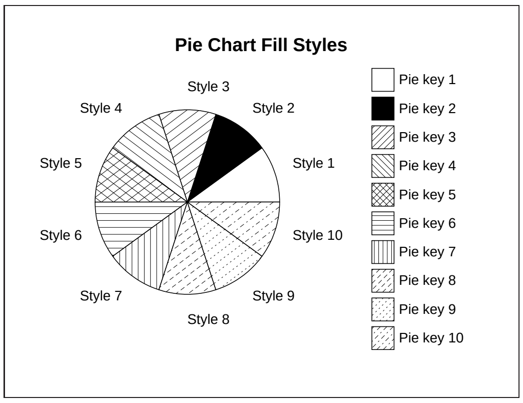


Figure 2.11: Typical bar chart features

Figure 2.11 was made by **simplot** using the *Advanced Bar Chart* option with `barchart.tf1` and `barchart.tf5`. The first plot illustrates groupings, while the second is a box and whisker plot with ranges, quartiles, and median. To see the possibilities, plot and browse the `barchart.tf?` test files. An easy way to prepare advanced bar chart files is to read in a matrix then save an advanced bar chart file. You can supply labels on the file, and change the position of the horizontal axis (the x-axis) to create hanging bar effects.

2.5.9 Example 6: Pie charts



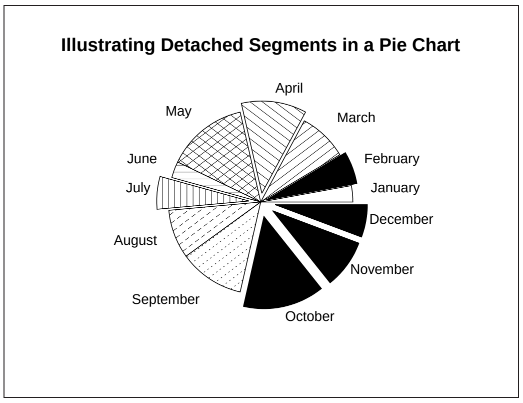
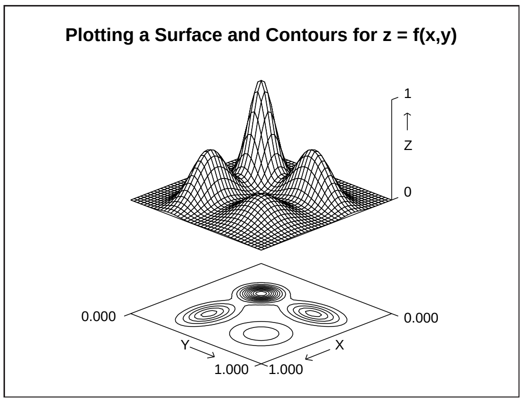


Figure 2.12: Typical pie chart features

Figure 2.12 was produced using the *Advanced Pie Chart* plotting option in **simplot** and `piechart.tf1` and `piechart.tf2`. By consulting the `w_readme.*` files or browsing these test files the convention for fill styles, labels, colors, segment offsets, etc. will be obvious. An easy way to create advanced pie chart files is to read in a vector with positive segment sizes then save an advanced pie chart.

2.5.10 Example 7: Surfaces, contours and 3D bar charts



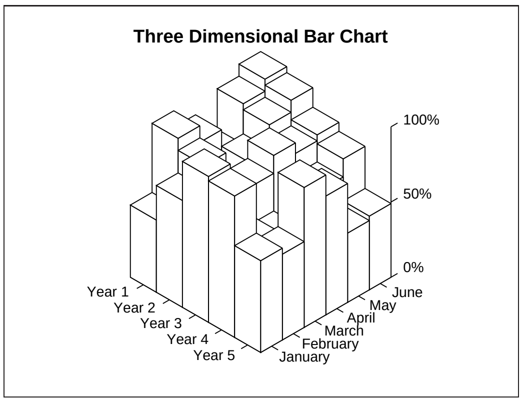


Figure 2.13: Plotting surfaces, contours and 3D-bar charts

Figure 2.13 illustrates a surface plot made using `surface.tf1` with the *Surface/Contour* option in program **simplot**, together with a three dimensional bar chart resulting from `barcht3d.tf1` (after editing legends). Surface plotting requires a mathematical expression for $z = f(x, y)$ and the program **makdat** should be used, since it is too tedious to type in sufficient data to generate a smooth surface. Three dimensional bar chart files do not usually require so much data, so they can easily be typed in, using program **makmat**. The format for surface plotting files will be found in the `w_readme.*` files. You will find that, once a surface file has been input into **simplot**, it is possible to plot just the surface, contours only, surface with contours, or a skyscraper plot. There are also many features to rotate the plot, change the axes, edit the legends, choose colors, add special effects, etc. Run all the `surface.tf?` files to appreciate the possibilities.

2.6 First time user's guide to simulation

Simulation in SimFIT involves creating exact data then adding pseudo random error to simulate experimental error. Exact data can be generated from a library of models or user-defined models.

2.6.1 Why fit simulated data ?

Statistical tests used in nonlinear regression are not exact, and optimization is not guaranteed to locate the best-fit parameters. This depends on the information content of your data and parameter redundancy in the model. To see how reliable your results are, you could perform a sensitivity analysis, to observe how results change as the data set is altered, e.g., by small variations in parameter values. For instance, suppose **hlfit** concludes that ligand binding data requires two binding sites by an F or run test, but only one binding constant is accurately determined as shown by a t test. You can then use **makdat** to make an exact data set with the best-fit parameters found by **hlfit**, use **adderr** to simulate your experiment, then fit the data using **hlfit**. If you do this repeatedly, you can collect weighted sums of squares, parameter estimates, t and F values and run test statistics, so judging the reliability of the result with your own data. This is a Monte Carlo method for sensitivity analysis.

2.6.2 Programs makdat and adderr

makdat makes exact $f(x), g(x, y)$ or $h(x, y, z)$ data. You choose from a library of models or supply your own model, then input parameter values and calculate, e.g., $y = f(x)$ for a range $X_{\text{start}} \le x \le X_{\text{stop}}$, either by fixing $X_{\text{start}}, X_{\text{stop}}$, or by choosing $Y_{\text{start}} = f(X_{\text{start}}), Y_{\text{stop}} = f(X_{\text{stop}})$, then allowing **makdat** to find appropriate values for X_{start} and X_{stop}. You must provide starting estimates for $X_{\text{start}}, X_{\text{stop}}$ to use the second method, which means that you must understand the mathematics of the model. With complicated models or differential equations, fix X_{start} and X_{stop} and observe how the graph of $y = f(x)$ changes as you change parameters and/or end points. When you have a good idea where your end points lie, try the option to fix y and calculate X_{start} and X_{stop}. This is needed when $f(x)$ is not monotonic in the range of interest. Output files from program **makdat** contain exact data, which can then be input into program **adderr** to add random errors.

2.6.3 Example 1: Simulating $y = f(x)$

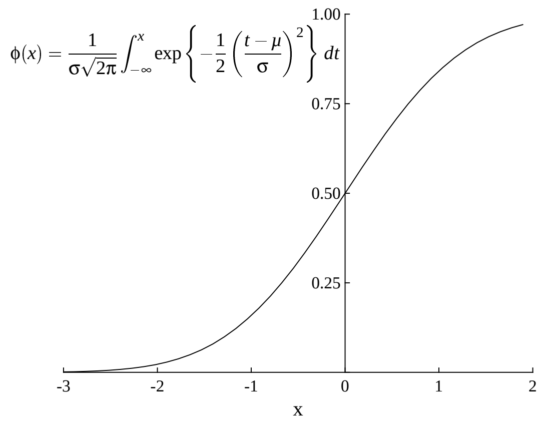


Figure 2.14: The normal cdf

The procedure is to select a model equation, set the model parameters to fixed values, decide on the range of x to explore, then plot the graph and save a file if appropriate. For example, run the program **makdat**, select functions of one variable, pick statistical distributions, choose the normal *cdf*, decide to have a zero constant term, set the mean $p(1) = 0$, fix the standard deviation $p(2) = 1$, input the scaling factor $p(3) = 1$ and then generate figure 2.14. Observe that there are two distinct ways to choose the range of x values; you can simply input the first and last x values, or you can input the first and last y values and let **makdat** find the corresponding x values numerically. This requires skill and an understanding of the mathematical behavior of the function chosen. Once you have simulated a model satisfactorily you can save a curve fitting type file which can be used by program **adderr** to add random error.

To illustrate the process of finding a range of x for simulation when this depends on fixed values of y, that is to find $x = x(y)$ when there is no simple explicit expression for $x(y)$, consider figure 2.15. Here the problem is to find $x = x(y)$ when y is the solution to the Von Bertalanffy growth differential equation

$$\frac{dy}{dx} = Ay^m - By^n,$$

where $A > 0, B > 0$ and $n > m$. After setting the parameters $A = B = m = 1$, $n = 2$, and initial condition $y_0 = 0.001$, for instance, program **makdat** estimated the following results:
$X_{start} = 0, X_{stop} = 5, y_1 = 0.1 : x_1 = 2.3919$,
$X_{start} = 0, X_{stop} = 9, y_2 = 0.9 : x_2 = 6.7679$,
providing the roots required to simulate this equation between the limits $y_1 = 0.1$ and $y_2 = 0.9$.

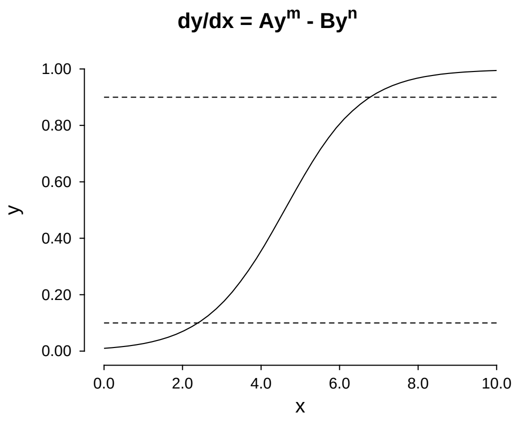


Figure 2.15: Using **makdat** to calculate a range

Note that, when attempting such root-finding calculations, **makdat** will attempt to alter the starting estimates if a root cannot be located by decreasing X_{start} and increasing X_{stop}, but it will not change the sign of these starting estimates. In the event of problems locating roots, there is no substitute for plotting the function to get some idea of the position of the roots, as shown in figure 2.15.

2.6.4 Example 2: Simulating $z = f(x, y)$

Simulating a function of two variables is very straightforward, and as a example we shall generate data for the function

$$z = x^2 - y^2$$

illustrated in figure 2.16. Again use **makdat** but this time select a function of two variables and then choose a polynomial of degree two. Do not include the constant term but choose the set of values $p(1) = 0$, $p(2) = 0$, $p(3) = 1$, $p(4) = 0$ and $p(5) = -1$. Now choose the extreme values of $x = -1$ to $x = 1$ and $y = -1$ to $y = 1$ with 20 divisions, i.e., 400 coordinates in all. Note that you can show the data in figure 2.16 as a surface, a contour, a surface with contours or a bar chart (skyscraper plot). You should plot the wire frame with a monochrome printer but the facet or patch designs can be used with a color printer. After simulating a surface you can save the coordinates for re-use by program **simplot**.

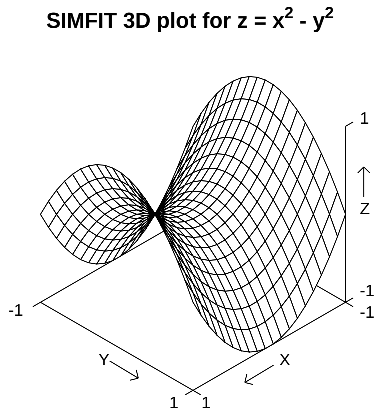


Figure 2.16: A 3D surface plot

2.6.5 Example 3: Simulating experimental error

The output files from program **makdat** contain exact data for $y = f(x)$, which is useful for graphs or data simulation. You may, however, want to add random error to exact data to simulate experimental error. To do this, the output file then becomes an input file for program **adderr**. After adding random error, the input file is left unchanged and a new output file is produced.

$$\text{Model} \longrightarrow \textbf{makdat} \longrightarrow \boxed{\begin{array}{c}\text{Exact}\\\text{data}\end{array}} \longrightarrow \textbf{adderr} \longrightarrow \boxed{\begin{array}{c}\text{Simulated}\\\text{data}\end{array}}$$

There are numerous ways to use program **adderr**, including generating replicates. If in doubt, pick 7% constant relative error with 3–5 replicates, as this mimics many situations. Note: constant relative error cannot be used where $y = 0$ (which invokes a default value). Read the test file `adderr.tfl` into program **adderr** and explore the various ways to add error. In most experiments a useful model for the variance of observations is

$$V(y) = \sigma_0^2 + \sigma_1^2 y^2,$$

so that the error resembles white noise at low response levels with a transition to constant relative error at high response levels. Constant variance ($\sigma_1 = 0$) fails to account for the way variance always increases as the signal increases, while constant relative error ($\sigma_0 = 0$) exaggerates the importance of small response values. However, a useful way to simulate error is to simulate four or five replicates with five to ten percent constant relative error as this is often fairly realistic. Using program **adderr** you can also simulate the effect of outliers or use a variety of error generating probability density functions, such as the Cauchy distribution (page 422) which is a often a better model for experimental error.

Points for plotting can be spaced by a SIMFIT algorithm to ensure continuity under transformations of axes, but to simulate experiments a geometric, or uniform spacing should be chosen. Then exact data simulated by program **makdat** is perturbed by program **adderr**. This is the method used to create many SIMFIT test files, e.g., `mmfit.tf4` from `mmfit.tf3`, as in figure 2.17. There are many ways to use program **adderr**, and care is needed to simulate realistically. If constant relative error is used, it is easy to preserve a mental picture of what is going on, e.g., 10% error conveys a clear meaning. However this type of error generation exaggerates the importance of small y values, biasing the fit in this direction. Constant variance is equally unrealistic, and over-emphasizes the importance of large y values. Outliers can also be simulated.

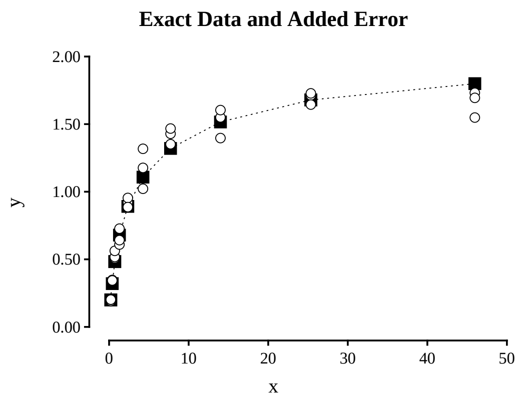


Figure 2.17: Adding random error

2.6.6 Example 4: Simulating differential equations

The best way to see how this is done is to run **deqsol** with the library of models provided. These are supplied with default choices of parameters and ranges so you can quickly see how to proceed. Try, for instance the Briggs-Haldane scheme which has five differential equations for substrate, product and enzyme species. The program can also be used for fitting data. So, to explore the fitting options, choose to simulate/fit a system of two differential equations and select the Lotka-Volterra predator-prey scheme

$$\begin{aligned} dy_1/dx &= p_1y_1 - p_2y_1y_2 \\ dy_2/dx &= -p_3y_2 + p_4y_1y_2. \end{aligned}$$

After you have simulated the system of equations and seen how the phase portrait option works you can try to fit the data sets in the library file `deqsol.tfl.` More advanced users will appreciate that a valuable feature of **deqsol** is that the program can simulate and fit linear combinations of the system of equations as defined by a transformation matrix. This is very valuable in areas like chemical kinetics where only linear combinations of intermediates can be measured, e.g., by spectroscopy. This is described in the `w_readme` files and can be explored using the test file `deqmat.tfl.` Figure 2.18 illustrates the simulation of a typical system, the Lotka-Volterra predator prey equations. After simulating the Lotka-Volterra equations you can select fitting and read in the library test file `deqsol.tfl` with predator-prey data for fitting. Much pleasure and instruction will result from using program **deqsol** with the library models provided and then, eventually, with your own models and experimental data.

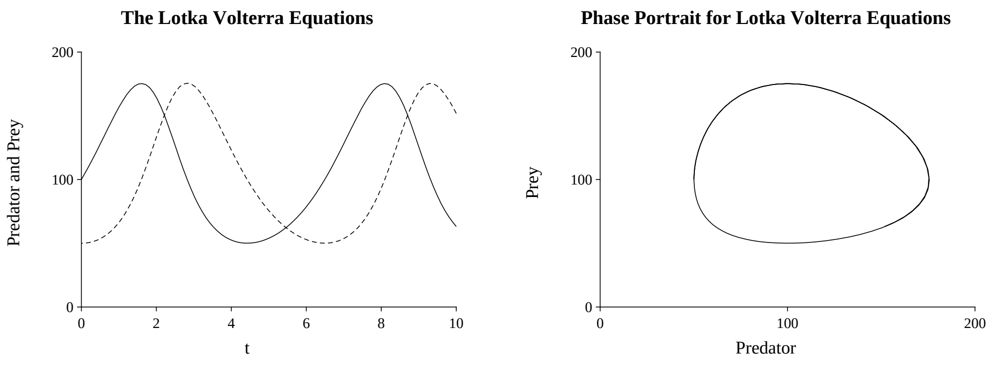


Figure 2.18: The Lotka-Volterra equations and phase plane

2.6.7 Example 5: Simulating user-defined equations

Figure 2.19 illustrates how to use **usermod** simulate a simple system of models. First select to simulate a set

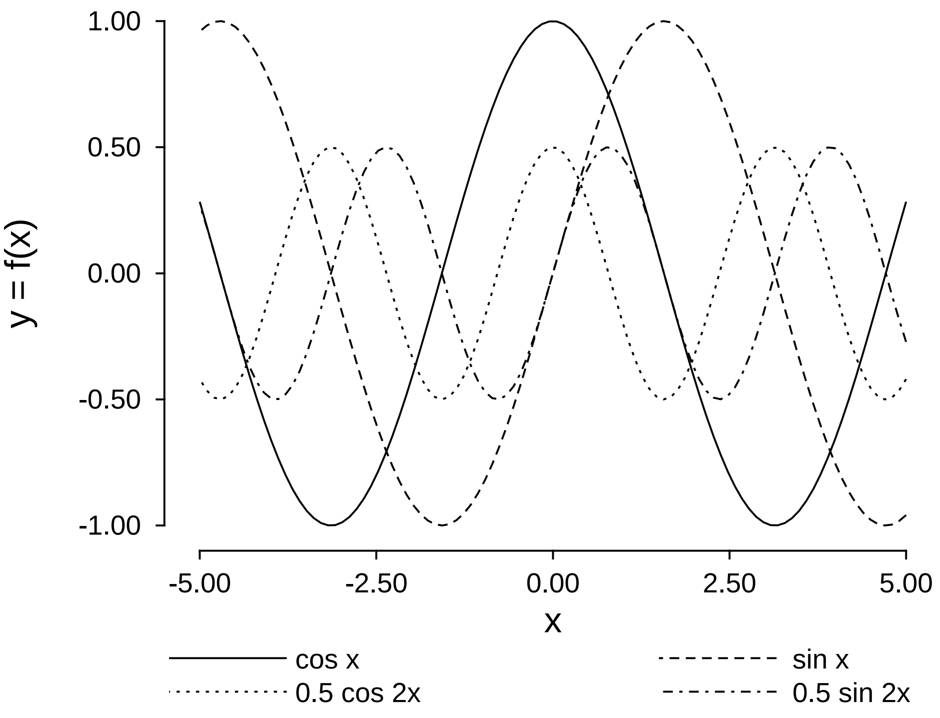


Figure 2.19: Plotting user supplied equations

of four equations, then read in the test file `usermodn.tf1` which defines the four trigonometric functions $f(1) = p_1 \cos x$, $f(2) = p_2 \sin x$, $f(3) = p_3 \cos 2x$, $f(4) = p_4 \sin 2x$.

Part 3

Data analysis techniques

3.1 Introduction

Before attempting to analyze your own experimental results you must be clear as to the nature of your data, as this will dictate the possible types of procedures that can be used. So we begin with the usual textbook classification of data types.

The classification of a variable X with scores x_A and x_B on two objects A and B will usually involve one of the following scales.

1. A *nominal scale* can only have $x_A = x_B$ or $x_A \neq x_B$, such as male or female.
2. An *ordinal scale* also allows $x_A > x_B$ or $x_A < x_B$, for instance bright or dark.
3. An *interval scale* assumes that a meaningful difference can be defined, so that A can be $x_A - x_B$ units different from B, as with temperature in degrees Celsius.
4. A *ratio scale* has a meaningful zero point so that we can say A is x_A/x_B superior to B if $x_A > x_B$ and $x_B \neq 0$, as with temperature in degrees Kelvin.

To many, these distinctions are too restrictive, and variables on nominal and ordinal scales are just known as categorical or qualitative variables, while variables on interval or ratio scales are known as quantitative variables. Again, variables that can only take distinct values are known as discrete variables, while variables that can have any values are referred to as continuous variables. Binary variables, for instance, can have only one of two values, say 0 or 1, while a categorical variable with k levels will usually be represented as a set of k $(0, 1)$ dummy variables where only one can be nonzero at each category level. Alternatively, taking a very broad and simplistic view, we could also classify experiments into those that yield objective measurements using scientific instruments, like spectrophotometers, and those that generate numbers in categories, i.e., counts. Measurements tend to require techniques such as analysis of variance or deterministic model fitting, while counts tend to require analysis of proportions or analysis of contingency tables. Of course, it is very easy to find exceptions to any such classifications; for instance modeling the size of a population using a continuous growth model when the population is a discrete not a continuous variable, but some commonsense is called for here.

3.2 Weighting

Frequently in data analysis functions are defined as weighted sums of n sub-functions as in

$$\begin{aligned} f(x) &= w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x) \\ &= f_1(x)/s_1^2 + f_2(x)/s_2^2 + \cdots + f_n(x)/s_n^2, \end{aligned}$$

where the weighting factors w_i or s_i are nonnegative numbers. There are three distinct ways this is done, and SimFIT users must be careful that weights are being used in the correct way for the procedure selected.

3.2.1 Arbitrary weights

In this situation a user suspects that some data points are more accurate than others, and uses weighting factors to emphasize the importance of accurate observations, or down-weight possible outliers. As this procedure is subjective, the performance of statistical tests may be compromised. An extreme case which is often encountered is where users decide to eliminate selected observations because they probably represent artifacts.

3.2.2 Replicate weights

Weights w_i are used by SimFIT in multivariate statistics where means of replicates are used either to compress data sets, or because individual observations are no longer available. To appreciate the motivation for this type of weighting consider the situation where there are n sample means $\bar{x}_i$, each obtained using a sample size n_i, and it is required to calculate the overall mean $\bar{x}$ as follows

$$\bar{x} = \sum_{i=1}^{n} n_i \bar{x}_i / \sum_{i=1}^{n} n_i.$$

Here the overall mean is obtained as a weighted sum of the individual sample means and we have

$$\begin{aligned} w_i &= n_i / \sum_{i=1}^{n} n_i, \text{ and} \\ \sum_{i=1}^{n} w_i &= 1. \end{aligned}$$

With weights used in this way, setting $w_i = 0$ on the data file is a convenient way to suppress observations while $w_i = n_i$ allows for replicates.

3.2.3 Curve fitting weights

Weights s_i are used by SimFIT in curve fitting to minimize the weighted sum of squares objective function

$$\begin{aligned} WSSQ &= \sum_{i=1}^{n} w_i (y_i - f(x_i|\Theta)^2 \\ &= \sum_{i=1}^{n} \left(\frac{y_i - f(x_i|\Theta}{s_i} \right)^2, \end{aligned}$$

with respect to a parameter vector Θ, where the s_i would be the estimated standard deviation of the responses y_i when the independent variable is x_i, or perhaps some function of the observations or best-fit model. The motivation for this is the hope that this is equivalent to Maximum Likelihood Estimation.

This follows by analogy from the definition of a chi-squared variable with n degrees of freedom as the sum of squares of n standardized normal variates, i.e.

$$\chi^2_n = \sum_{i=1}^{n} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2,$$

but using this analogy could be pushing interpretation too far as, in practise, neither the means μ_i or variances σ_i^2 are available, only estimates. Accordingly, in curve fitting there are essentially four possibilities.

1. **Using the observations.**
   The weights are calculated according to a formula such as

$$w_i = 1/(A + By_i^2),$$

where A represents an error component with constant variance, and B accounts for an error component with constant relative error. If parameters A and B can be estimated independently, and not from the data, and this model for the experimental error can be justified independently then this is a sensible way to proceed. However, estimating A and B at the same time as model fitting decreases the degrees of freedom, greatly decreases the accuracy of parameter estimates, and is very questionable, as weights are changed for each iteration of the optimization.

2. **Using the best-fit model.**
The weights are calculated according to a similar formula such as

$$w_i = 1/(A + Bf(x_i|\hat{\Theta})^2).$$

This is even more questionable than the previous method because it assumes a correct model has been identified, and it means that weights are even more drastically altered at each iteration of the optimization. Setting $A = 0$ and $B = 1$, or some other scalar for example, is known as chi-square minimization, but this procedure greatly exaggerates the importance of small observations.

3. **Using standard errors.**
In this case the weights would be

$$w_i = 1/s_i^2,$$

where the s_i are standard errors calculated from replicates. Perhaps this is the safest way, but only as long as the replicates at each design points are sufficiently numerous to allow a meaningful estimate of the standard errors. An absolute minimum sample size of 3, and preferably very much greater than 5 is required.

4. **Constant weighting.**
Taking the option

$$w_i = 1$$

does not represent absence of weighting: it involves the assumption that the sample variance is constant, and not a function of the independent variable. Just as the assumption of fixed relative error leads to over-emphasizing low value observations, the assumption of constant variance places too much emphasis on large observations when, as every experimentalist knows, there is usually a minimum variance component at low values, but approximately relative error takes over at large values.

3.3 Principles involved when fitting models to data

A frequently occurring situation is where an investigator has a vector of n observations $y_1, y_2, \ldots, y_n$, with errors ϵ_i, at settings of some independent variable x_i which are supposedly known exactly, and wishes to fit a model $f(x)$ to the data in order to estimate m parameters $\theta_1, \theta_2, \ldots, \theta_m$ by minimizing some appropriate function of the residuals r_i. If the true model is $g(x)$ and a false model $f(x)$ has been fitted, then the relationship between the observations, errors and residuals r_i would be

$$\begin{aligned} y_i &= g(x_i) + \epsilon_i \\ r_i &= y_i - f(x_i) \\ &= g(x_i) - f(x_i) + \epsilon_i. \end{aligned}$$

That is, the residuals would be sums of a model error term plus a random error term as follows

$$\begin{aligned} \text{Model Error } &= g(x_i) - f(x_i) \\ \text{Random Error } &= \epsilon_i. \end{aligned}$$

If the model error term is appreciable, then fitting is a waste of time, and if the nature of the error term is not taken into account, any parameters estimated are likely to be biased. An important variation on this theme is when the control variables x_i are not known with high precision, as they would be in a precisely controlled laboratory experiment, but are themselves random variables as in biological experiments, and so best regarded as covariates rather than independent variables. The principle used most often in data fitting is to choose those parameters that make the observations as likely as possible, that is, to appeal to the principle of maximum likelihood. This is seldom possible to do as the true model is generally unknown so that an approximate model has to be used, and the statistical nature of the error term is not usually known with certainty. A further complication is that iterative techniques must often be employed to estimate parameters by maximum likelihood, and these depend heavily on starting estimates and frequently locate false local minima rather than the desired global minimum. Again, the values of parameters to be estimated often have a physical meaning, and so are constrained to restricted regions of parameter space, which means that constrained regression has to be used, and this is much more problematical than unconstrained regression.

3.3.1 Limitations when fitting models

It must be emphasized that, when fitting a function of k variables such as

$$y = f(x_1, x_2, \dots, x_k, \theta_1, \theta_2, \dots, \theta_m)$$

to n data points in order to estimate m parameters, a realistic approach must be adopted to avoid over-interpretation as follows.

- **Independent variables**
  If the data y are highly accurate measurements, i.e. with high signal to noise ratios (page 276), and the variables x can be fixed with high precision, then it is reasonable to regard x as independent variables and attempt to fit models based upon physical principles. This can only be the case in disciplines such as physics and chemistry where the y would be quantities such as absorption of light or concentrations, and the x could be things like temperatures or times. The model would then be formulated according to the appropriate physical laws, such as the law of mass action, and it would generally be based on differential equations.

- **Covariates**
  In biological experiments, the the data y are usually much more noisy and there may even be random variation in the x variables. Then it would be more appropriate to regard the x as covariates and only fit simple models, like low order rational functions or exponentials. In some cases models such as nonlinear growth models could be fitted in order to estimate physically meaningful parameters, such as the maximum growth rate, or final asymptotic size but, in extreme cases, it may only make sense to fit models like polynomials for data smoothing, where the best-fit parameters are purely empirical and cannot be interpreted in terms of established physical laws.

- **Categorical variables**
  Where categorical variables are encountered then parallel shift models must be fitted. In this case each variable with l levels is taken to be equivalent to l dummy indicator variables which can be either 0 or 1. However one of these is then suppressed arbitrarily to avoid aliasing and the levels of categorical variables are simply interpreted as factors that contribute to the regression constant. Clearly this is a very primitive method of analysis which easily leads to over-interpretation where there are more than a couple of variables and more than two or three categories.

In all cases, the number of observations must greatly exceed the number of parameters that are to be estimated, say for instance by a factor of ten.

3.3.2 Fitting linear models

If the assumed model is of the form

$$f(x) = \theta_1\phi_1(x) + \theta_2\phi_2(x) + \cdots + \theta_m\phi_m(x)$$

it is linear in the parameters θ_j, and so can be easily fitted by linear regression if the errors are normally distributed with zero mean and known variance σ_i^2, since maximum likelihood in this case is equivalent to minimizing the weighted sum of squares

$$WSSQ = \sum_{i=1}^{n} \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2$$

with respect to the parameters. SimFIT provides model free fitting by cubic splines, simple linear regression as in

$$f(x) = \theta_1 + \theta_2 x,$$

multilinear regression

$$f(x) = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m,$$

polynomial regression

$$f(x) = \theta_1 + \theta_2 x + \theta_3 x^2 + \cdots + \theta_m x^{m-1},$$

and also transformed polynomial regression, where new variables are defined by

$$X = X(x)$$
$$Y = Y(y)$$

and a polynomial is fitted to $Y(X)$. Models like these are used for data smoothing, preliminary investigation, and fitting noisy data over a limited range of independent variable. That is, in situations where developing meaningful scientific models may not be possible or profitable. With linear models, model discrimination is usually restricted to seeing if some reduced parameter set is sufficient to explain the data upon using the F test, t tests are employed to check for parameter redundancy, and goodness of fit tends to be based on chi-square tests on $WSSQ$ and normality of studentized residuals. The great advantage of linear regression is the attractively simple conceptual scheme and ease of computation. The disadvantage is that the models are not based on scientific laws, so that the parameter estimates do not have a physical interpretation. Another serious limitation is that prediction is not possible by extrapolation, e.g., if growth data are fitted using polynomials and the asymptotic size is required.

3.3.3 Fitting generalized linear models

These models are mostly used when the errors do not follow a normal distribution, but the explanatory variables are assumed to occur linearly in a model sub-function. The best known example would be logistic regression, but the technique can also be used to fit survival models. Because the distribution of errors may follow a non-normal distribution, various types of deviance residuals are used in the maximum likelihood objective function. Sometimes these techniques have special advantages, e.g., predicting probabilities of success or failure as a functions of covariates after binary logistic regression is certain to yield probability estimates between zero and one because the model

$$-\infty < \log \left(\frac{y}{1-y} \right) < \infty$$

implies that $0 < y < 1$.

3.3.4 Fitting nonlinear models

Many models fitted to data are constructed using scientific laws, like the law of mass action, and so these will usually be nonlinear and may even be of rather complex form, like systems of nonlinear differential equations, or convolution integrals, and they may have to be expressed in terms of special functions which have to evaluated by numerical techniques, e.g., inverse probability distributions. Success in this area is heavily dependent on having accurate data over a wide range of the independent variable, and being in possession of good starting estimates. Often, with simple models like low order exponentials

$$f(x) = A_1 \exp(-k_1 x) + A_2 \exp(-k_2 x) + \cdots + A_m \exp(-k_m x),$$

rational functions

$$f(x) = \frac{V_1 x}{K_1 + x} + \frac{V_2 x}{K_2 + x} + \cdots + \frac{V_m x}{K_m + x},$$

or growth models

$$f(x) = \frac{A}{1 + B\exp(-kx)},$$

good starting estimates can be estimated from the data and, where this is possible, SimFIT has a number of dedicated user-friendly programs that will perform all the necessary scaling. However, for experts requiring advanced fitting techniques a special program **qnfit** is provided.

3.3.5 Fitting survival models

There are four main techniques used to analyze survival data.

1. Estimates of proportions of a population surviving as a function of time are available by some technique which does not directly estimate the number surviving in a populations of known initial size, rather, proportions surviving are inferred by indirect techniques such as light scattering for bacterial density or enzyme assay for viable organisms. In such instances the estimated proportions are not binomial variables so fitting survival models directly by weighted least squares is justified, especially where destructive sampling has to be used so that autocorrelations are less problematical. Program **gcfit** is used in mode 2 for this type of fitting (see page 77).
2. A population of individuals is observed and information on the times of censoring (i.e. leaving the group) or failure are recorded, but no covariates are measured. In this case, survival density functions, such as the Weibull model, can be fitted by maximum likelihood, and there are numerous statistical and graphical techniques to test for goodness of fit. Program **gcfit** is used in mode 3 for this type of fitting (see page 253).
3. When there are covariates as well as survival times and censored data, then survival models can be fitted as generalized linear models. The SimFIT GLM simplified interface module is used for this type of analysis (see page 257).
4. The Cox proportional hazards model does not attempt to fit a complete model, but a partial model can be fitted by the method of partial likelihood as long as the proportional hazards assumption is justified independently. Actually, after fitting by partial likelihood, a piece-wise hazard function can be estimated and residuals can then be calculated. The SimFIT GLM simplified interface module is used for this type of analysis (page 259).

3.4 Goodness of fit

After a model has been fitted to data it is important to assess goodness of fit, which can only be done if assumptions are made about the model and the distribution of experimental errors. If a correct linear model is fitted to data, and the errors are independently normally distributed with mean zero and known standard deviation which is used for weighting, then a number of exact statistical results apply. If there are m parameters and n experimental measurements, the sum of weighted squared residuals $WSSQ$ is a chi-square variable with $n - m$ degrees of freedom, the m ratios

$$t_i = \frac{\theta_i - \hat{\theta}_i}{\hat{s}_i}$$

involving the exact parameters θ_i, estimated parameters $\hat{\theta}_i$, and estimated standard errors $\hat{s}_i$ are t distributed with $n - m$ degrees of freedom and, if fitting a model with m_1 parameters results in $WSSQ_1$ but fitting the next model in the hierarchy with m_2 parameters gives the weighted sum of squares $WSSQ_2$, then

$$F = \frac{(WSSQ_1 - WSSQ_2)/(m_2 - m_1)}{WSSQ_2/(n - m_2)}$$

is F distributed with $m_2 - m_1$ and $n - m_2$ degrees of freedom. When $n \gg m$ the weighted residuals will be approximately unit normal variables ($\mu = 0, \sigma = 1$), their signs will be binomially distributed with parameters n and 0.5, the runs minus 1 given n will be binomially distributed with parameters $n - 1$ and 0.5, while the runs given the number of positive and negative signs will follow a more complicated distribution (page 153).

With nonlinear models and weights estimated from replicates at distinct x_i, i.e., not known exactly, statistical tests are no longer exact. SIMFIT programs allow you to simulate results and see how close the statistics are to the exact ones. There are program to evaluate the probability density (or mass) function and the cumulative distribution function for a chosen distribution, as well as calculating percentage points. In addition, you can use program **makmat** to make files containing your statistics, and these numbers can then be tested to see if they are consistent with the chosen distribution.

3.4.1 The chi-square test for goodness of fit

Let $WSSQ$ = weighted sum of squares and $NDOF$ = no. degrees of freedom (no. points - no. parameters). If all $s = 1$, $WSSQ/NDOF$ estimates the (constant) variance σ^2. You can compare it with any independent estimate of the (constant) variance of response y. If you had set s = exact std. dev., $WSSQ$ would be a chi-square variable, and you could consider rejecting a fit if the probability of chi-square exceeding $WSSQ$ (i.e., $P(\chi^2 \geq WSSQ)$) is $<.01(1\%$ significance level) or $<0.05(5\%$ significance level). Where standard error estimates are based on 3–5 replicates, you can reasonably decrease the value of WSSQ by 10–20% before considering rejecting a model by this chi-square test.

3.4.2 The t test for parameter redundancy

The number T = (parameter estimate)/(standard error) can be referred to the t distribution to assess any parameter redundancy, where $P(t \leq -|T|) = P(t \geq |T|) = \alpha/2$. Two tail p values are defined as $p = \alpha$, and parameters are significantly different from 0 if $p < .01(1\%)$ ($< .05(5\%)$). Parameter correlations can be assessed from corresponding elements of the correlation matrix.

3.4.3 The F test for model discrimination

The F test just described is very useful for discriminating between models with up to 3 or 4 parameters. For models with more than 4 parameters, calculated F test statistics are no longer approximately F distributed, but they do estimate the extent to which model error is contributing to excess variance from fitting a deficient model. It is unlikely that you will ever have data that is good enough to discriminate between nonlinear models with much more than 5 or 6 parameters in any case.

3.4.4 Analysis of residuals

The plot of residuals (or better weighted residuals) against dependent or independent variable or best-fit response is a traditional (arbitrary) approach that should always be used, but many prefer the normal or half normal plots. The sign test is weak and should be taken rather seriously if rejection is recommended (P(signs $\leq$ observed) $< .01$ (or .05)). The run test conditional on the sum of positive and negative residuals is similarly weak, but the run test conditional on observed positive and negative residuals is quite reliable, especially if the sample size is fairly large (> 20 ?). Reject if P(runs $\leq$ observed) is $< .01$ (1%) (or $< .05$ (5%))

3.4.5 How good is the fit ?

If you set $s = 1$, $WSSQ/NDOF$ should be about the same as the (constant) variance of y. You can consider rejecting the fit if there is poor agreement in a variance ratio test. If s = sample standard deviation of y (which may be the best choice ?), then $WSSQ$ is approximately chi-square and should be around $NDOF$. Relative residuals do not depend on s. They should not be larger than 25%, there should not be too many symbols ***, ****, or ***** in the residuals table and also, the average relative residual should not be much larger than 10%. These, and the R-squared test, are all convenient tests for the magnitude of the difference between your data and the best-fit curve.

A graph of the best-fit curve should show the data scattered randomly above and below the fitted curve, and the number of positive and negative residuals should be about the same. The table of residuals should be free from long runs of the same signs, and the plot of weighted residuals against independent variable should be like a sample from a normal distribution with $\mu = 0$ and $\sigma = 1$, as judged by the Shapiro-Wilks test, and the normal or half normal plots. The sign, run and Durbin-Watson tests help you to detect any correlations in the residuals.

3.5 Testing for differences between two parameter estimates

This can sometimes be a useful simple procedure when you wish to compare two parameters resulting from a regression, e.g., the final size from fitting a growth curve model, or perhaps two parameters that have been derived from regression parameters e.g., AUC from fitting an exponential model, or LD50 from bioassay. You input the two parameters, the standard error estimates, the total number of experimental observations, and the number of parameters estimated from the regression. A t test (page 136) for equality is then performed using the correction for unequal variances. Such t tests depend on the asymptotic normality of maximum likelihood parameters, and will only be meaningful if the data set is fairly large and the best fit model adequately represents the data. Furthermore, t tests on parameter estimates are especially unreliable because they ignore non-zero covariances in the estimated parameter variance-covariance matrix.

3.6 Testing for differences between several parameter estimates

To take some account of the effect of significant off-diagonal terms in the estimated parameter variance-covariance matrix you will need to calculate a Mahalanobis distance between parameter estimates e.g., to test if two or more curve fits using the same model but with different data sets support the presence of significant treatment effects. For instance, after fitting the logistic equation to growth data by nonlinear regression, you may wish to see if the growth rates, final asymptotic size, half-time, etc. have been affected by the treatment. Note that, after every curve fit, you can select an option to add the current parameters and covariance matrix to your parameter covariance matrix project archive, and also you have the opportunity to select previous fits to compare with the current fit. For instance, you may wish to compare two fits with m parameters, A in the first set with estimated covariance matrix C_A and B in the second set with estimated covariance matrix C_B. The parameter comparison procedure will then perform a t test for each pair of parameters, and also calculate the quadratic form

$$Q = (A - B)^T (C_A + C_B)^{-1}(A - B)$$

which has an approximate chi-square distribution with m degrees of freedom. You should realize that the rule of thumb test using non-overlapping confidence regions is more conservative than the above t test; parameters can still be significantly different despite a small overlap of confidence windows.

This technique must be used with care when the models are sums of functions such as exponentials, Michaelis-Menten terms, High-Low affinity site isotherms, Gaussians, trigonometric terms, and so on. This is because the parameters are only unique up to a permutation. For instance, the terms A_i and k_i are linked in the exponential function

$$f(t) = \sum_{i=1}^{m} A_i \exp(-k_i t)$$

but the order implied by the index i is arbitrary. So, when testing if A_1 from fitting a data set is the same as A_1 from fitting another data set it is imperative to compare the same terms. The user friendly programs **exfit**, **mmfit**, and **hlfit** attempt to assist this testing procedure by rearranging the results into increasing order of amplitudes A_i but, to be sure, it is best to use **qnfit**, where starting estimates and parameter constraints can be used from a parameter limits file. That way there is a better chance that parameters and covariance matrices saved to project archives for retrospective testing for equality of parameters will be consistent, i.e. the parameters will be compared in the correct order.

Figure 3.1 illustrates a common problem, where the same model has been fitted to alternative data sets and it

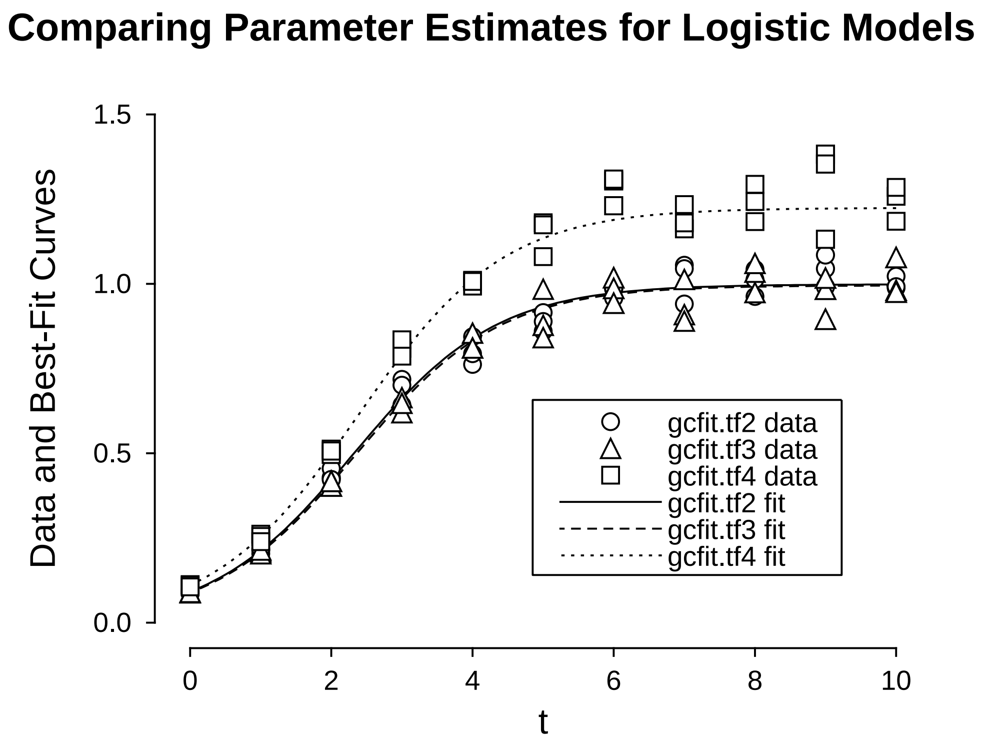


Figure 3.1: Comparing parameter estimates

is wished to decide if one or more parameters differ significantly.

In this case, the logistic model defined as

$$f(t) = \frac{p_1}{1 + p_2 \exp(-p_3 t)}$$

was simulated using **makdat** and **adderr** then fitted by **gcfit**, and the main interest is to decide if the estimated final asymptote i.e. $\hat{p}_1$ differs significantly for the test files `gcfit.tf2` and `gcfit.tf3` which actually have identical parameters $p_1 = 1$, while `gcfit.tf4` has a slightly larger asymptotic value $p_1 = 1.25$, the other parameters being identical $p_2 = 10$ and $p_3 = 1$.

Table 3.1 illustrates how this technique works.
The data were fitted using **gcfit** using the option to store parameter estimates and covariance matrices. Then the global tests for different parameter sets, and t tests for individual parameter differences were performed, leading to the results indicated.

Clearly the parameter estimates for test files `gcfit.tf2` and `gcfit.tf3` indicate no significant differences, while `gcfit.tf4` differed significantly from both of these, due to a larger value for the asymptote p_1 for `gcfit.tf4`.

```
Mahalanobis chi-square, and corrected pairwise t tests for
differences between parameters(A,B) and covariances(Ca,Cb).
Comparison 1: Parameters from gcfit.tf3 (A) and gcfit.tf2 (B)
Q = (A-B)^T(Ca+Cb)^(-1)(A-B) = 2.193, NDOF = 3
Probability(Chi-square >= Q) = 0.5333
Index          A           B       A - B          t     DOF     p
    1     0.9956      0.9989     -0.0033    -0.2567      53  0.7984
    2      10.15        9.89      0.2600     0.7224      40  0.4743
    3     0.9848      0.9881     -0.0033    -0.1164      37  0.9908
Comparison 2: Parameters from gcfit.tf4 (A) and gcffit.tf2 (B)
Q = (A-B)^T(Ca+Cb)^(-1)(A-B) = 749.2, NDOF = 3
Probability(Chi-square >= Q) = 0.0000  Reject H0 at 1% sig.level
Index          A           B       A - B          t     DOF     p
    1      1.224      0.9989      0.2251      19.17      57  0.0000  *****
    2      10.04        9.89        0.15     0.3823      50  0.7038
    3      0.969      0.9881     -0.0191   -0.06294      46  0.9501
Comparison 3: Parameters from gcfit.tf4 (A) and gcfit.tf3 (B)
Q = (A-B)^T(Ca+Cb)^(-1)(A-B) = 1064.0, NDOF = 3
Probability(Chi-square >= Q) = 0.0000  Reject H0 at 1% sig.level
Index          A           B       A - B         t     DOF     p
    1      1.224      0.9956      0.2284     16.21      59  0.0000  *****
    2      10.04       10.15     -0.1100    -0.443      52  0.6596
    3      0.969      0.9848     -0.0158  -0.09271      52  0.9265
```

Table 3.1: Comparing parameter estimates

3.7 Graphical deconvolution of complex models

Many decisions depend on differentiating nested models, e.g., polynomials, or models in sequence of increasing order, e.g., sums of Michaelis-Mentens or exponentials, and you should always use the option in **qnfit**, **exfit**, **mmfit** and **hlfit** (see page 38) to plot the terms in the sum as what can loosely be described as a graphical deconvolution before accepting the results of an F test to support a richer model.

The advantage of the graphical deconvolution technique is that you can visually assess the contribution of individual component functions to the overall sum, as in figure 3.2 which shows the graphical deconvolution of best fit curves as follows.

1. Fitting a sum of three Gaussian *pdfs* to the test file `gauss3.tf1` using **qnfit** (page 79).
2. Fitting a double exponential function to the test file `exfit.tf4` using **exfit** (page 60).
3. Fitting a double binding site curve to the test file `hotcold.tf1` using **mmfit** in isotope displacement mode (page 69).

Graphical deconvolution, which displays graphs for the individual components making up a composite functions defined as the sum of these components, should always be done after fitting sums of monomials, Michaelis-Mentens, High/Low affinity sites, exponentials, logistics or Gaussians, to assess the contribution of the individual components to the overall fit, before accepting statistical evidence for improved fit. Many claims for three exponentials or Michaelis-Mentens would not have been made if this simple graphical technique had been used.

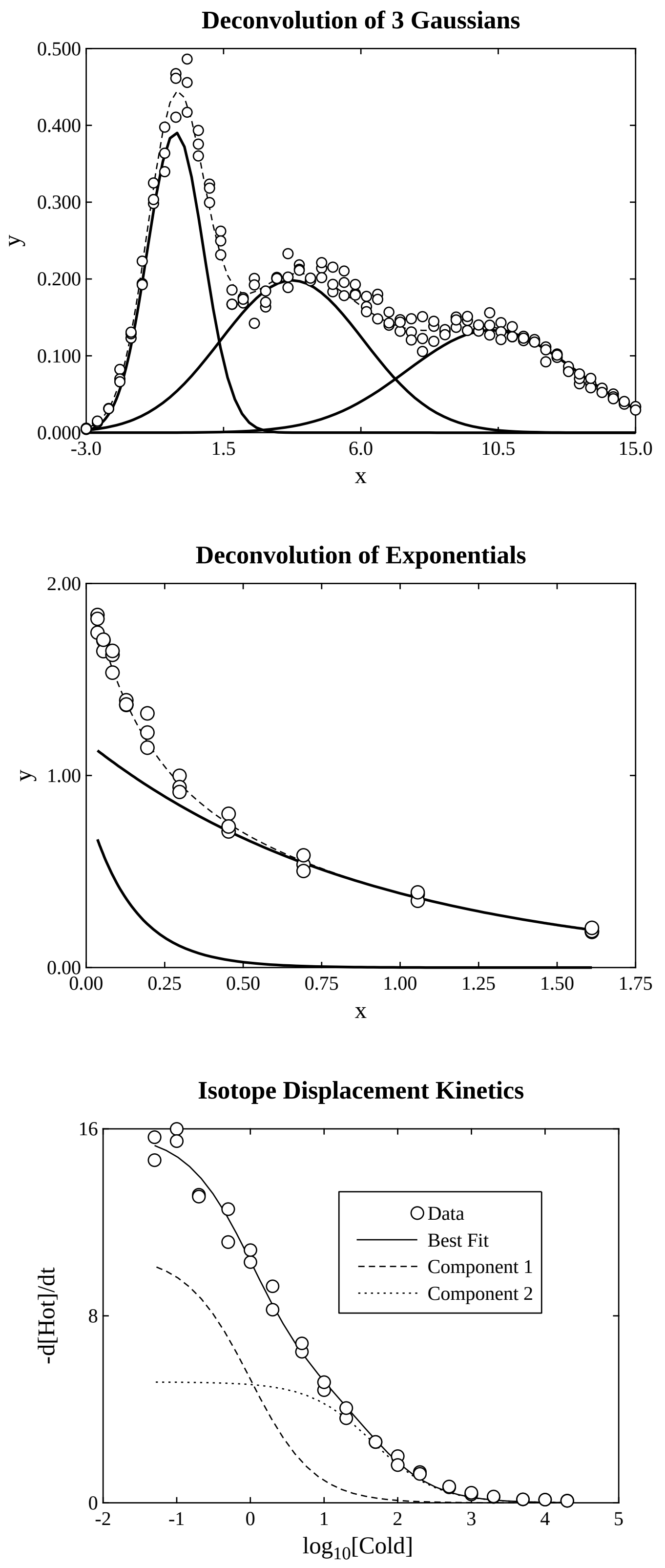


Figure 3.2: Graphical deconvolution of complex models

Part 4

Linear models

4.1 Introduction

A linear model is one of the form

$$f(x) = \theta_1\phi_1(x) + \theta_2\phi_2(x) + \cdots + \theta_n\phi_n(x)$$

where the n parameters θ_i are to be estimated, while the n functions $\phi_i(x)$ are known functions of the independent variable x. Examples would be polynomials or multilinear models, and models of this type are particularly easy to fit and often yield unique solutions, which explains why they are often used in situations where they are not strictly justified.

4.2 Linear regression

Program **linfit** fits a multilinear model in the form

$$y = \beta_0x_0 + \beta_1x_1 + \beta_2x_2 + \cdots + \beta_mx_m,$$

where $x_0 = 1$, but you can choose interactively whether or not to include a constant term β_0, you can decide which variables are to be included, and you can use a weighting scheme if this is required. For each regression sub-set, you can observe the parameter estimates and standard errors, R-squared, Mallows C_p (page 65), and $ANOVA$ table, to help you decide which combinations of variables are the most significant. Unlike nonlinear regression, multilinear regression, which is based on the assumptions

$$\begin{aligned} Y &= X\beta + \epsilon, \\ \mathrm{E}(\epsilon) &= 0, \\ \mathrm{Var}(\epsilon) &= \sigma^2 I, \end{aligned}$$

allows us to introduce the hat matrix

$$H = X(X^TX)^{-1}X^T,$$

then define the leverages h_{ii}, which can be used to asses influence, and the studentized residuals

$$R_i = \frac{r_i}{\hat{\sigma}\sqrt{1 - h_{ii}}},$$

which may offer some advantages over ordinary residuals r_i for goodness of fit assessment from residuals plots. In the event of weighting being required, Y, X and ϵ above are simply replaced by $W^{\frac{1}{2}}Y$, $W^{\frac{1}{2}}X$, and $W^{\frac{1}{2}}\epsilon$.

Model discrimination is particularly straightforward with multilinear regression, and it is often important to answer questions like these.

- Is a particular parameter well-determined ?
- Should a particular parameter be included in the regression ?
- Is a particular parameter estimate significantly different between two data sets ?
- Does a set of parameter estimates differ significantly between two data sets ?
- Are two data sets with the same variables significantly different ?

To assess the reliability of any parameter estimate, SIMFIT lists the estimated standard error, the 95% confidence limits, and the two tail p value for a t test (page 136). If the p value for a parameter is appreciably greater than 0.05, the parameter can be regarded as indistinguishable from zero, so you should consider suppressing the the corresponding variable from the regression and fitting again. To select a minimum significant set of variables for regression you should perform the F test (page 155) and, to simplify the systematic use of this procedure, SIMFIT allows you to save sets of parameters and objective functions after a regression so that these can be recalled retrospectively for the F test. With large numbers of variables it is very tedious to perform all subsets regression and a simple expedient would be to start by fitting all variables, then consider for elimination any variables with ill-defined parameter estimates. It is often important to see if a particular parameter is significantly different between two data sets, and SIMFIT provides the facility to compare any two parameters in this way, as long as you know the values, standard errors, and numbers of experimental points. This procedure (page 36) is very over-simplified, as it ignores other parameters estimated in the regression, and their associated covariances. A more satisfactory procedure is to compare full sets of parameters estimated from regression using the same model with different data sets, and also using the estimated variance-covariance matrices. SIMFIT provides the facility to store parameters and variance-covariance matrices in this way (page 36), and this is a valuable technique to asses if two data sets are significantly different, assuming that if two sets of parameter estimates are significantly different, then the data sets are significantly different.

Two well documented and instructive data sets will be found in `linfit.tf1` and `linfit.tf2` but be warned; real life phenomena are nonlinear, and the multilinear model is seldom the correct one. In fact the data in `linfit.tf1` have a singular design matrix, which deserves comment. It sometimes happens that data have been collected at settings of the variables that are not independent. This is particularly troublesome with what are known as (badly) designed experiments, e.g., 0 for female, 1 for male and so on. Another common error is where percentages or proportions in categories are taken as variables in such a way that a column in the design matrix is a linear combination of other columns, e.g., because all percentages add up to 100 or all proportions add up to 1. When the SIMFIT regression procedures detect singularity you will be warned that the covariance matrix is singular and that the parameter standard errors should be ignored. However, with multilinear regression, the fitting still takes place using the singular value decomposition (SVD) and parameters and standard errors are printed. However only some parameters will be estimable and you should redesign the experiment so that the covariance matrix is of full rank and all parameters are estimable. If SIMFIT warns you that a covariance matrix cannot be inverted or that SVD has to be used then you should not carry on regardless: the results are likely to be misleading so you should redesign the experiment so that all parameters are estimable.

As an example, after fitting the test file `linfit.tf2`, table 4.1 results. From the table of parameter estimates it is clear that the estimated parameter confidence limits all include zero, and that all the parameter p values all exceed 0.05. So none of the parameters are particularly well-determined. However, from the C_p value, half normal residuals plot and ANOVA table, with overall p value less than 0.05 for the F value, it appears that a multilinear model does fit the overall data set reasonably well. The fact that the leverages are all of similar size and that none of the studentized residuals are of particularly large absolute magnitude (all less than 2) suggests that none of the data points are could be considered as outliers. Note that program **linfit** also lets you explore generalized linear modelling (GLM), reduced major axis regression (minimizing the sum of areas of triangles formed between the best-fit line and data points), orthogonal regression (minimizing the sum of squared distances between the best-fit line and the data points), and robust regression in the L_1 or L_∞ norms as, alternatives to the usual L_2 norm.

```
 Number of parameters = 5, Rank = 5, Number of points = 13, NDOF = 8
 Residual-SSQ = 47.8636, Mallows' Cp = 5.0, R-squared = 0.9824
 Parameter     Value   Lower95%cl Upper95%cl  Std.Error    p
  Constant   62.4054    -99.1786    223.989    70.071  0.3991 ***
    B(1)      1.5511    -0.16634    3.26855    0.74477  0.0708   *
    B(2)    0.510168    -1.15889    2.17923   0.723788  0.5009 ***
    B(3)    0.101909    -1.63845    1.84227   0.754709  0.8959 ***
    B(4)    0.144061    -1.77914    1.49102   0.709052  0.8441 ***

Number  Y-value     Theory     Residual      Leverage  Studentized
    1      78.5    78.4952   0.00476042      0.550285   0.00290214
    2      74.3    72.7888       1.5112      0.333243     0.756625
    3     104.3    105.971     -1.67094      0.576942     -1.05027
    4      87.6    89.3271      -1.7271      0.295237    -0.841081
    5      95.9    95.6492     0.250756      0.357601     0.127906
    6     109.2    105.275      3.92544      0.124156      1.71482
    7     102.7    104.149     -1.44867      0.367076     -0.74445
    8      72.5     75.675     -3.17499       0.40854      -1.6878
    9      93.1    91.7217      1.37835      0.294305       0.6708
   10     115.9    115.618     0.281548      0.700403     0.210293
   11      83.8     81.809      1.99098      0.425508      1.07391
   12     113.3    112.327     0.972989      0.262983     0.463352
   13     109.4    111.694     -2.29433       0.30372     -1.12411

 ANOVA
 Source       NDOF          SSQ        Mean-SSQ       F-value      p
 Total          12      2715.76
 Regression      4       2667.9         666.975       111.479   0.0000
 Residual        8      47.8636         5.98295
```

Table 4.1: Multilinear regression

4.3 Polynomial regression

Program **polnom** sequentially fits polynomials of degree $n = 0, 1, 2, 3, 4, 5, 6$ of the type

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n,$$

using Chebyshev polynomials $T_i(.)$ as follows

$$T(\bar{x}) = 0.5 A_1 T_0(\bar{x}) + A_2 T_1(\bar{x}) + A_3 T_2(\bar{x}) + \cdots + A_{n+1} T_n(\bar{x})$$

for numerical stability, where

$$\bar{x} = \frac{(x - x_{min}) - (x_{max} - x)}{x_{max} - x_{min}}.$$

Note that, as $x_{min} \le x \le x_{max}$, then $-1 \le \bar{x} \le 1$, and after fitting, **polnom** maps $T(\bar{x})$ back into $p(x)$. Polynomials are not mathematical models as such because they are too flexible and cannot model the sort of horizontal asymptotes frequently encountered with experimental data, e.g. growth curves. However, as long as low degrees are used and curves with turning points are treated with care, they can be used for calibration or deciding if a there is a trend in noisy data sets. Here a major issue is deciding on the minimum degree that is justified statistically, and table 4.2 illustrates how **polnom** does this. In this case the data are from `polnom.tf1` suggesting that degree 2 is justified, and the analysis using `e02adf.tf1` indicates very clearly from inspecting graphs for degrees 0, 1, 2, and 3 that here degree 3 is required, as will be seen from figure 4.1.

```
n  Chebyshev coefficients A0, A1,..., An

0  0.3111269
1  16.033596  7.9079669
2  12.736903  4.8193935   -1.44559
3  12.734680  4.8131684    -1.4591  -0.008377
4  12.761856   4.834225   -1.43871  -0.055083  -0.059560
5  12.654161  4.6601972   -1.38581  -0.087456  -0.035275  0.2297923

n  SIGMA  change   WSSQ %change P(C>=WSSQ) 5%  F-val (F>=F-val) 5%
0 36.70338        22901.34         0.0000    no
1 8.083255 77.98 1045.424 95.44  0.0000    no   334.500 0.0000 yes
2 0.991432 87.73 14.74406 98.59  0.4700   yes  1048.572 0.0000 yes
3 1.025319  3.42  14.7179  0.18  0.3977   yes  0.024878 0.8769  no
4 1.051113  2.52 14.36291  2.41  0.3488   yes  0.321311 0.5805  no
5 0.999964  4.87 11.99915 16.46  0.4457   yes  2.363927 0.1501  no

  Results for weighted fitting (w = 1/s^2)
  Parameter    Value    Std.Error  Lower95%cl  Upper95%cl    p
   p(0)      0.103471  0.00320912   0.0966304    0.110311  0.0000
   p(1)       2.12035   0.0197309     2.07829      2.1624  0.0000
   p(2)     -0.115647  0.00357137   -0.123259   -0.108035  0.0000
```

Table 4.2: Results from polynomial regression

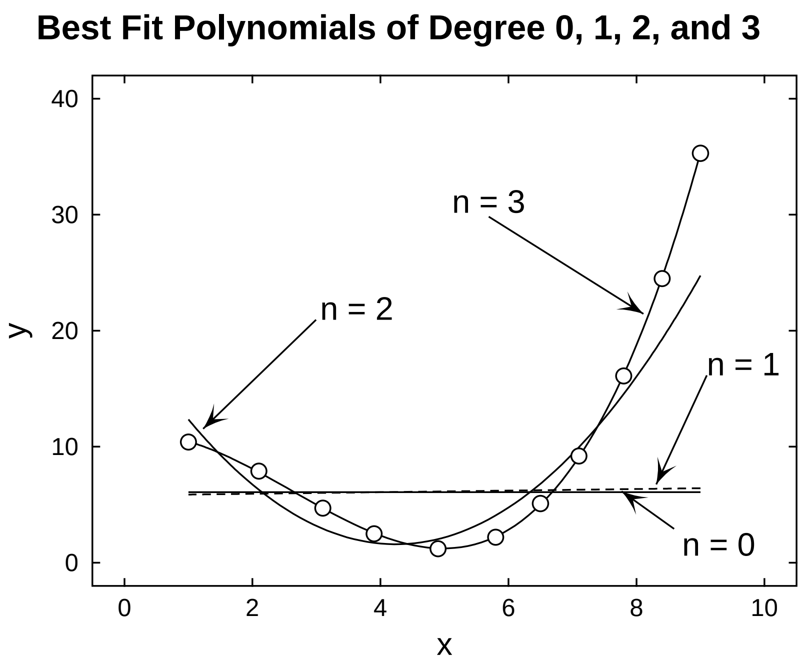


Figure 4.1: Plots from polynomial regression

4.4 Robust regression

Robust techniques are required when in the linear model

$$Y = X\beta + \epsilon$$

the errors ϵ are not normally distributed. There are several alternative techniques available, arising from a consideration of the objective function to be minimized by the fitting technique. One approach seeks to suppress the well known effect that outliers bias parameter estimates by down-weighting extreme observations, and yet another technique uses regression on ranks. First we consider the p-norm of a n-vector x, which is defined as

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p},$$

and the objective functions required for maximum likelihood estimation. There are three cases.

1. The L_1 norm.
   If the errors are bi-exponentially distributed (page 423) then the correct objective function is the sum of the absolute values of all the residuals
   $$\sum_{i=1}^{n} |Y - X\hat{\beta}|.$$
2. The L_2 norm.
   If the errors are normally distributed with known variances σ_i (page 420) then the correct objective function, just as in weighted least squares, is
   $$\sum_{i=1}^{n} [(Y - X\hat{\beta})/\sigma_i]^2.$$
3. The L_∞ norm.
   If the errors are uniformly distributed (page 420) then the correct objective function is the largest absolute residual
   $$\max_n |Y - X\hat{\beta}|.$$

Although SIMFITprovides options for L_1 and L_∞ fitting by iterative techniques, parameter standard error estimates and analysis of variance tables are not calculated. The techniques can be used to assess how serious the effects of outliers are in any given data set, but otherwise they should be reserved for situations where either a bi-exponential or uniform distribution of errors seems more appropriate than a normal distribution of errors.

In actual experiments it is often the case that there are more errors in the tails of the distribution than are consistent with a normal distribution, so that the error distribution is more like a Cauchy distribution (page 422) than a normal distribution. For such circumstances a variety of techniques exist for dampening down the effect of such outliers. One such technique is bounded influence regression, which leads to M-estimates , but it should be obvious that

$$\boxed{\text{Garbage In = Garbage Out}}$$

.

No automatic technique can extract meaningful results from bad data and, where possible, serious experimentalists should, of course, strive to identify outliers and remove them from the data set, rather than use automatic techniques to suppress the influence of outliers by down-weighting. Robust regression is a technique for when all else fails, outliers cannot be identified independently and deleted, so the experimentalist is forced to analyze noisy data sets where analysis of the error structure is not likely to be meaningful as sufficient replicates cannot be obtained. An abstract of the NAG G02HAF documentation is now given to clarify the options, and this should be consulted for more details and references.

If r_i are calculated residuals, w_i are estimated weights, $\psi(.)$ and $\chi(.)$ are specified functions, $\phi(.)$ is the unit normal density and $\Phi(.)$ the corresponding distribution function, σ is a parameter to be estimated, and α_1, α_2 are constants, then there are three main possibilities.

1. Schweppe regression

$$\begin{aligned}\sum_{i=1}^{n} \psi(r_i/(\sigma w_i))w_i x_{ij} &= 0, \qquad j = 1, 2, \ldots, m\\ \Phi(\alpha_1) &= 0.75\\ \alpha_2 &= \frac{1}{n}\sum_{i=1}^{n} w_i^2 \int_{-\infty}^{\infty} \chi(z/w_i)\phi(z)\,dz\end{aligned}$$

2. Huber regression

$$\begin{aligned}\sum_{i=1}^{n} \psi(r_i/\sigma)x_{ij} &= 0, \qquad j = 1, 2, \ldots, m\\ \Phi(\alpha_1) &= 0.75\\ \alpha_2 &= \int_{-\infty}^{\infty} \chi(z)\phi(z)\,dz\end{aligned}$$

3. Mallows regression

$$\begin{aligned}\sum_{i=1}^{n} \psi(r_i/\sigma)w_i x_{ij} &= 0, \qquad j = 1, 2, \ldots, m\\ \frac{1}{n}\sum_{i=1}^{n} \Phi(\alpha_1/\sqrt{w_i}) &= 0.75\\ \alpha_2 &= \frac{1}{n}\sum_{i=1}^{n} w_i \int_{-\infty}^{\infty} \chi(z)\phi(z)\,dz\end{aligned}$$

The estimate for σ can be obtained at each iteration by the median absolute deviation of the residuals

$$\hat{\sigma} = \operatorname*{median}_i(|r_i|)/\alpha_1$$

or as the solution to

$$\sum_{i=1}^{n} \chi(r_i/(\hat{\sigma} w_i))w_i^2 = (n-k)\alpha_2$$

where k is the column rank of X.

For the iterative weighted least squares regression used for the estimates there are several possibilities for the functions ψ and χ, some requiring additional parameters c, h_1, h_2, h_3, and d.

(a) Unit weights, equivalent to least-squares.

$$\psi(t) = t, \qquad \chi(t) = t^2/2$$

(b) Huber's function

$$\psi(t) = \max(-c, \min(c, t)), \qquad \chi(t) = \begin{cases} t^2/2, & |t| \le d\\ d^2/2, & |t| > d\end{cases}$$

(c) Hampel's piecewise linear function

$$\psi_{h_1,h_2,h_3}(t) = -\psi_{h_1,h_2,h_3}(-t) = \begin{cases} t, & 0 \le t \le h_1 \\ h_1, & h_1 \le t \le h_2 \\ h_1(h_3 - t)/(h_3 - h_2), & h_2 \le t \le h_3 \\ 0, & h_3 < t \end{cases} \qquad \chi(t) = \begin{cases} t^2/2, & |t| \le d \\ d^2/t, & |t| > d \end{cases}$$

(d) Andrew's sine wave function

$$\psi(t) = \begin{cases} \sin t, & -\pi \le t \le \pi \\ 0, & |t| > \pi \end{cases} \qquad \chi(t) = \begin{cases} t^2/2, & |t| \le d \\ d^2/2, & |t| > d \end{cases}$$

(e) Tukey's bi-weight

$$\psi(t) = \begin{cases} t(1 - t^2)^2, & |t| \le 1 \\ 0, & |t| > 1 \end{cases} \qquad \chi(t) = \begin{cases} t^2/2, & |t| \le d \\ d^2/2, & |t| > d \end{cases}$$

Weights w_i require the definition of functions $u(.)$ and $f(.)$ as follows.

(i) Krasker-Welsch weights

$$\begin{aligned} u(t) &= g_1(c/t) \\ g_1(t) &= t^2 + (1 - t^2)(2\Phi(t) - 1) - 2t\phi(t) \\ f(t) &= 1/t \end{aligned}$$

(ii) Maronna's weights

$$\begin{aligned} u(t) &= \begin{cases} c/t^2, & |t| > c \\ 1, & |t| \le c \end{cases} \\ f(t) &= \sqrt{u(t)} \end{aligned}$$

Finally, in order to estimate the parameter covariance matrix, two diagonal matrices D and P are required as follows.

1. Average over the r_i

Schweppe

$$D_i = \left(\frac{1}{n} \sum_{j=1}^{n} \psi'(r_j/(\hat{\sigma} w_i) \right) wi$$

$$P_i = \left(\frac{1}{n} \sum_{j=1}^{n} \psi^2(r_j/(\hat{\sigma} w_i) \right) w_i^2$$

Mallows

$$D_i = \left(\frac{1}{n} \sum_{j=1}^{n} \psi'(r_j/\hat{\sigma}) \right) w_i$$

$$P_i = \left(\frac{1}{n} \sum_{j=1}^{n} \psi^2(r_j/(\hat{\sigma}) \right) w_i^2$$

2. Replace expected value by observed

Schweppe

$$D_i = \psi'(r_i/(\hat{\sigma} w_i) wi$$

$$P_i = \psi^2(r_i/(\hat{\sigma} w_i) w_i^2$$

Mallows

$$D_i = \psi'(r_i/\hat{\sigma}) w_i$$

$$P_i = \psi^2(r_i/(\hat{\sigma}) w_i^2$$

Table 4.3 illustrates the use of robust regression using the test file `g02haf.tf1`. Note that the output lists all the settings used to configure NAG routine G02HAF and, in addition, it also presents the type of results usually

```
  G02HAF settings: INDW > 0, Schweppe with Krasker-Welsch weights
                   IPSI = 2, Hampel piecewise linear
                   ISIG > 0, sigma using the chi function
                   INDC = 0, Replacing expected by observed
  H1 = 1.5,   H2 = 3.0,  H3 = 4.50
CUCV = 3.0, DCHI = 1.5, TOL = 0.00005

 Number of parameters = 3, Rank = 3, Number of points = 8, NDOF = 5
 Residual-SSQ = 0.464, Mallows' Cp = 3.0, R-squared = 0.9844
 Final sigma value = 0.2026
 Parameter       Value     95% conf. limits      Std.error     p
  Constant       4.042       3.944       4.141     0.03842   0.0000
    B(1)         1.308       1.238       1.378     0.02720   0.0000
    B(2)        0.7519      0.6719      0.8319      0.0312   0.0000

Number     Y-value     Theory   Residual   Weighting
     1         2.1      1.982     0.1179      0.5783
     2         3.6      3.486     0.1141      0.5783
     3         4.5      4.599    -0.9872      0.5783
     4         6.0      6.103  -0.002564      0.5783
     5         1.3      1.426    -0.1256      0.4603
     6         1.9      2.538    -0.6385      0.4603
     7         6.7      6.659    0.04103      0.4603
     8         5.5      5.546   -0.04615      0.4603

 ANOVA
 Source       NDOF          SSQ    Mean SSQ     F-value       p
 Total           7        29.66
 Regression      2        29.19         14.6       157.3     0.0000
 Residual        5       0.4639      0.09278
```

Table 4.3: Robust regression

associated with standard multilinear regression. Of course these calculations should be interpreted with great caution if the data sample has many outliers, or has errors that depart widely from a normal distribution. It should be noted that, in the SimFIT implementation of this technique, the starting estimates required for the iterations used by the robust estimation are first calculated automatically by a standard multilinear regression. Another point worth noting is that users of all SimFIT multilinear regression analysis can either supply a matrix with a first column of $x_1 = 1$ and suppress the option to include a constant in the regression, or omit the first column for $x_1 = 1$ from the data file, whereupon SimFIT will automatically add such a column, and do all the necessary adjustments for degrees of freedom.

4.5 Regression on ranks

It is possible to perform regression where observations are replaced by ranks, as illustrated for test data `g08raf.tf1`, and `g08rbf.tf1` in table 4.4.
It is assumed that a monotone increasing differentiable transformation h exists such that

$$h(Y_i) = x_i^T \beta + \epsilon_i$$

for observations Y_i given explanatory variables x_i, parameters β, and random errors ϵ_i, when the following can be estimated, and used to test $H_0 : \beta = 0$.

```
File: G08RAF.TF1 (1 sample, 20 observations)
parameters = 2, distribution = logistic

CTS = 8.221, NDOF = 2, P(chi-sq >= CTS) = 0.0164
      Score   Estimate   Std.Err.    z-value    p
     -1.048    -0.8524      1.249    -0.6824  0.4950  ***
      64.33     0.1139    0.04437      2.567  0.0103

CV matrices: upper triangle for scores, lower for parameters
     0.67326     -4.1587
      1.5604      533.67
    0.012160   0.0019686

File: G08RBF.TF1 (1 sample, 40 observations)
parameters = 1, gamma = 0.00005

CTS = 2.746, NDOF = 1, P(chi-sq >= CTS) = 0.0975
      Score   Estimate   Std.Err.    z-value    p
      4.584      5.990     0.3615      1.657  0.0975  *

CV matrices: upper triangle for scores, lower for parameters
   7.6526
  0.13067
```

Table 4.4: Regression on ranks

① $X^T a$, the score statistic.

② $X^T(B - A)X$, the estimated variance covariance matrix of the scores.

③ $\hat{\beta} = MX^T a$, the estimated parameters.

④ $M = (X^T(B - A)X)^{-1}$, the estimated variance covariance matrix of $\hat{\beta}$.

⑤ $CTS = \hat{\beta}^T M^{-1}\hat{\beta}$, the χ^2 test statistic.

⑥ $\hat{\beta}_i/\sqrt{M_{ii}}$, the approximate z statistics.

Here M and a depend on the ranks of the observations, while B depends on the the distribution of ϵ, which can be assumed to be normal, logistic, extreme value, or double-exponential, when there is no censoring. However, table 4.4 also displays the results from analyzing `g08rbf.tf1`, which contains right censored data. That is, some of the measurements were capped, i.e., an upper limit to measurement existed, and all that can be said is that an observation was at least as large as this limit. In such circumstances a parameter γ must be set to model the distribution of errors as a generalized logistic distribution, where as γ tends to zero a skew extreme value is approached, when γ equals one the distribution is symmetric logistic, and when γ tends to infinity the negative exponential distribution is assumed.

Part 5

Generalized linear models (GLM)

5.1 Introduction

This technique is intermediate between linear regression, which is trivial and gives uniquely determined parameter estimates but is rarely appropriate, and nonlinear regression, which is very hard and does not usually give unique parameter estimates, but is justified with normal errors and a known model. The SimFIT generalized models interface can be used from **gcfit**, **linfit** or **simstat** as it finds many applications, ranging from bioassay to survival analysis.

To understand the motivation for this technique, it is usual to refer to a typical doubling dilution experiment in which diluted solutions from a stock containing infected organisms are plated onto agar in order to count infected plates, and hence estimate the number of organisms in the stock. Suppose that before dilution the stock had N organisms per unit volume, then the number per unit volume after $x = 0, 1, \ldots, m$ dilutions will follow a Poisson dilution with $\mu_x = N/2^x$. Now the chance of a plate receiving no organisms at dilution x is the first term in the Poisson distribution, that is $\exp(-\mu_x)$, so if p_x is the probability of a plate becoming infected at dilution x, then

$$p_x = 1 - \exp(-\mu_x),\ x = 1, 2, \ldots, m.$$

Evidently, where the p_x have been estimated as proportions from y_x infected plates out of n_x plated at dilution x, then N can be estimated using

$$\log[-\log(1 - p_x)] = \log N - x \log 2$$

considered as a maximum likelihood fitting problem of the type

$$\log[-\log(1 - p_x)] = \beta_0 + \beta_1 x$$

where the errors in estimated proportions $p_x = y_x/n_x$ are binomially distributed. So, to fit a generalized linear model, you must have independent evidence to support your choice for an assumed error distribution for the dependent variable Y from the following possibilities:

❏ normal

❏ binomial

❏ Poisson

❏ gamma

in which it is supposed that the expectation of Y is to be estimated, i.e.,

$$E(Y) = \mu.$$

The associated *pdfs* are parameterized as follows.

$$
\begin{aligned}
\text{normal}: f_Y &= \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\\
\text{binomial: } f_Y &= \binom{N}{y}\pi^y(1-\pi)^{N-y}\\
\text{Poisson: } f_Y &= \frac{\mu^y\exp(-\mu)}{y!}\\
\text{gamma: } f_Y &= \frac{1}{\Gamma(\nu)}\left(\frac{\nu y}{\mu}\right)^{\nu}\exp\left(-\frac{\nu y}{\mu}\right)\frac{1}{y}
\end{aligned}
$$

It is a mistake to make the usual unwarranted assumption that measurements imply a normal distribution, while proportions imply a binomial distribution, and counting processes imply a Poisson distribution, unless the error distribution assumed has been verified for your data. Another very questionable assumption that has to made is that a predictor function η exists, which is a linear function of the m covariates, i.e., independent explanatory variables, as in

$$\eta = \sum_{j=1}^{m}\beta_j x_j.$$

Finally, yet another dubious assumption must be made, that a link function $g(\mu)$ exists between the expected value of Y and the linear predictor. The choice for

$$g(\mu) = \eta$$

depends on the assumed distribution as follows. For the binomial distribution, where y successes have been observed in N trials, the link options are the logistic, probit or complementary log-log

$$
\begin{aligned}
\text{logistic: } \eta &= \log\left(\frac{\mu}{N-\mu}\right)\\
\text{probit: } \eta &= \Phi^{-1}\left(\frac{\mu}{N}\right)\\
\text{complementary log-log: } \eta &= \log\left(-\log\left(1-\frac{\mu}{N}\right)\right).
\end{aligned}
$$

Where observed values can have only one of two values, as with binary or quantal data, it may be wished to perform binary logistic regression. This is just the binomial situation where y takes values of 0 or 1, N is always set equal to 1, and the logistic link is selected. However, for the normal, Poisson and gamma distributions the link options are

$$
\begin{aligned}
\text{exponent: } \eta &= \mu^a\\
\text{identity: } \eta &= \mu\\
\text{log: } \eta &= \log(\mu)\\
\text{square root: } \eta &= \sqrt{\mu}\\
\text{reciprocal: } \eta &= \frac{1}{\mu}.
\end{aligned}
$$

In addition to these possibilities, you can supply weights and install an offset vector along with the data set, the regression can include a constant term if requested, the constant exponent a in the exponent link can be altered, and variables can be selected for inclusion or suppression in an interactive manner. However, note that the same strictures apply as for all regressions: you will be warned if the SVD has to be used due to rank deficiency and you should redesign the experiment until all parameters are estimable and the covariance matrix has full rank, rather than carry on with parameters and standard errors of limited value.

5.2 GLM examples

The test files to investigate the GLM functions are:
`glm.tf1`: normal error and reciprocal link
`glm.tf2`: binomial error and logistic link (logistic regression)
`glm.tf3`: Poisson error and log link
`glm.tf4`: gamma error and reciprocal link
where the data format for k variables, observations y and weightings s is

$$x_1, x_2, \ldots, x_k, y, s$$

except for the binomial error which has

$$x_1, x_2, \ldots, x_k, y, N, s$$

for y successes in N independent Bernoulli trials. Note that the weights w used are actually $w = 1/s^2$ if advanced users wish to employ weighting, e.g., using s as the reciprocal of the square root of the number of replicates for replicate weighting, except that when $s \leq 0$ the corresponding data points are suppressed. Also, observe the alternative measures of goodness of fit, such as residuals, leverages and deviances. The residuals r_i, sums of squares SSQ and deviances d_i and overall deviance DEV depend on the error types as indicated in the examples.

GLM example 1: G02GAF, normal errors

Table 5.1 has the results from fitting a reciprocal link with mean but no offsets to `glm.tf1`. Note that the

```
 Number of parameters = 2, Rank = 2, Number of points = 5, NDOF = 3
 Parameter       Value      95% conf. limits       Std.error      p
  Constant     -0.02387     -0.03272     -0.01503     0.002779   0.0033
      B(1)      0.06381      0.05542      0.07221     0.002638   0.0002
WSSQ = 0.3872, S = 0.1291, A = 1.0

Number      Y-value      Theory    Dev-resid     Leverage
     1        25.00       25.04    -0.03866       0.9954
     2         10.0       9.639      0.3613       0.4577
     3          6.0       5.968     0.03198       0.2681
     4          4.0       4.322     -0.3221       0.1666
     5          3.0       3.388     -0.3878       0.1121
```

Table 5.1: GLM example 1: normal errors

scale factor (S = σ^2) can be input or estimated using the residual sum of squares SSQ defined as follows

$$\text{normal error: } r_i = y_i - \hat{\mu}_i$$
$$SSQ = \sum_{i=1}^{n} r_i.$$

GLM example 2: G02GBF, binomial errors

Table 5.2 shows the results from fitting a logistic link and mean but no offsets to `glm.tf2`. The estimates are

```
 Number of parameters = 2, Rank = 2, Number of points = 3, NDOF = 1
 Parameter        Value      95% conf. limits       Std.error     p
  Constant       -2.868      -4.415      -1.322       0.1217    0.0270
      B(1)      -0.4264      -2.457       1.604       0.1598    0.2283 ***
Deviance = 0.07354

Number      Y-value      Theory    Dev-resid     Leverage
     1         19.0       18.45       0.1296       0.7687
     2         29.0        30.1      -0.2070        0.422
     3         24.0       23.45       0.1178       0.8092
```

Table 5.2: GLM example 2: binomial errors

defined as follows

$$\begin{aligned}\text{binomial error: } d_i &= 2\left\{y_i \log\left(\frac{y_i}{\hat{\mu}_i}\right) + (t_i - y_i)\log\left(\frac{t_i - y_i}{t_i - \hat{\mu}_i}\right)\right\}\\ r_i &= \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}\\ DEV &= \sum_{i=1}^{n} d_i.\end{aligned}$$

GLM example 3: G02GCF, Poisson errors

Table 5.3 shows the results from fitting a log link and mean but no offsets to `glm.tf3`. The definitions are

$$\begin{aligned}\text{Poisson error: } d_i &= 2\left\{y_i \log\left(\frac{y_i}{\hat{\mu}_i}\right) - (y_i - \hat{\mu}_i)\right\}\\ r_i &= \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}\\ DEV &= \sum_{i=1}^{n} d_i,\end{aligned}$$

but note that an error message is output to warn you that the solution is overdetermined, i.e., the parameters and standard errors are not unique. To understand this, we point out that the data in `glim.tf3` are the representation of a contingency table using dummy indicator variables (page 55) as will be clear from table 5.4. Thus, in order to obtain unique parameter estimates, it is necessary to impose constraints so that the resulting constrained system is of full rank. Let the singular value decomposition (SVD) P^* be represented, as in G02GKF, by

$$P^* = \begin{pmatrix} D^{-1}P_1^T \\ P_0^T \end{pmatrix},$$

and suppose that there are m parameters and the rank is r, so that there need to be $n_c = m - r$ constraints, for example, in a m by n_c matrix C where

$$C^T\beta = 0.$$

Then the constrained estimates $\hat{\beta}_c$ are given in terms of the SVD parameters $\hat{\beta}_{svd}$ by

$$\begin{aligned}\hat{\beta}_c &= A\hat{\beta}_{svd}\\ &= (I - P_0(C^TP_0)^{-1}C^T)\hat{\beta}_{svd},\end{aligned}$$

while the variance-covariance matrix V is given by

$$V = AP_1D^{-2}P_1^TA^T,$$

```
 Number of parameters = 9, Rank = 7, Number of points = 15, NDOF = 8
 Parameter        Value     95% conf. limits      Std.error     p
  Constant        2.598       2.538       2.657     0.02582   0.0000
      B(1)        1.262       1.161       1.363     0.04382   0.0000
      B(2)        1.278       1.177       1.378     0.04362   0.0000
      B(3)      0.05798    -0.09595      0.2119     0.06675   0.4104 ***
      B(4)        1.031      0.9036       1.158     0.05509   0.0000
      B(5)        0.291      0.1223      0.4598     0.07317   0.0041
      B(6)       0.9876      0.8586       1.117     0.05593   0.0000
      B(7)        0.488      0.3322      0.6437     0.06754   0.0001
      B(8)      -0.1996     -0.4080    0.008754    -0.09035   0.0582   *
Deviance = 9.038, A = 1.0

Number     Y-value      Theory    Dev-resid    Leverage
     1       141.0       133.0       0.6875      0.6035
     2        67.0       63.47       0.4386      0.5138
     3       114.0       127.4       -1.207      0.5963
     4        79.0       77.29       0.1936      0.5316
     5        39.0       38.86      0.02218      0.4820
     6       131.0       135.1      -0.3553      0.6083
     7        66.0       64.48       0.1881      0.5196
     8       143.0       129.4        1.175      0.6012
     9        72.0       78.52      -0.7465      0.5373
    10        35.0       39.48      -0.7271      0.4882
    11        36.0        39.9      -0.6276      0.3926
    12        14.0       19.04       -1.213      0.2551
    13        38.0       38.21     -0.03464      0.3815
    14        28.0       23.19       0.9675      0.2825
    15        16.0       11.66        1.203      0.2064
```

Table 5.3: GLM example 3: Poisson errors

```
Test file loglin.tf1   Test file glm.tf3
141  67 114  79  39    1 0 0 1 0 0 0 0 141 1
131  66 143  72  35    1 0 0 0 1 0 0 0  67 1
 36  14  38  28  16    1 0 0 0 0 1 0 0 114 1
                       1 0 0 0 0 0 1 0  79 1
                       1 0 0 0 0 0 0 1  39 1
                       0 1 0 1 0 0 0 0 131 1
                       0 1 0 0 1 0 0 0  66 1
                       0 1 0 0 0 1 0 0 143 1
                       0 1 0 0 0 0 1 0  72 1
                       0 1 0 0 0 0 0 1  35 1
                       0 0 1 1 0 0 0 0  36 1
                       0 0 1 0 1 0 0 0  14 1
                       0 0 1 0 0 1 0 0  38 1
                       0 0 1 0 0 0 1 0  28 1
                       0 0 1 0 0 0 0 1  16 1
```

Table 5.4: GLM contingency table analysis: 1

provided that $(C^T P_0^{-1})$ exists. This approach is commonly used in log-linear analysis of contingency tables, but it can be tedious to first fit the overdetermined Poisson GLM model then apply a matrix of constraints as just described. For this reason SɪᴍFɪT provides an automatic procedure (page 147) to calculate the dummy indicator matrix from the contingency table then fit a log-linear model and apply the further constraints that the row sum and column sum are zero. Table 5.5 illustrates how this is done with `loglin.tf1`.

```
Number of rows = 3, Number of columns = 5
Deviance (D) = 9.038, NDOF = 8, p = P(chi-sq>=D) = 0.3391
Parameter    Estimate  Std.Err.  ..95% con. lim....    p
Constant       3.983    0.0396      3.89     4.07    0.0000
Row   1       0.3961    0.0458      0.29     0.502   0.0000
Row   2       0.4118    0.0457      0.306    0.517   0.0000
Row   3      -0.8079    0.0622     -0.951   -0.664   0.0000
Col   1       0.5112    0.0562      0.382    0.641   0.0000
Col   2      -0.2285    0.0727     -0.396    0.0608  0.0137   *
Col   3       0.468     0.0569      0.337    0.599   0.0000
Col   4     -0.03155    0.0675     -0.187    0.124   0.6527 ***
Col   5      -0.7191    0.0887     -0.924   -0.515   0.0000
     Data       Model      Delta  Residual  Leverage
      141      132.99       8.01    0.6875    0.6035
       67       63.47       3.53    0.4386    0.5138
      114      127.38     -13.38   -1.2072    0.5963
       79       77.29       1.71    0.1936    0.5316
       39       38.86       0.14    0.0222    0.4820
      131      135.11      -4.11   -0.3553    0.6083
       66       64.48       1.52    0.1881    0.5196
      143      129.41      13.59    1.1749    0.6012
       72       78.52      -6.52   -0.7465    0.5373
       35       39.48      -4.48   -0.7271    0.4882
       36       39.90      -3.90   -0.6276    0.3926
       14       19.04      -5.04   -1.2131    0.2551
       38       38.21      -0.21   -0.0346    0.3815
       28       23.19       4.81    0.9675    0.2825
       16       11.66       4.34    1.2028    0.2064
```

Table 5.5: GLM contingency table analysis: 2

GLM example 4: G02GDF, gamma errors

Table 5.6 shows the results from fitting a reciprocal link and mean but no offsets to `glm.tf4`. Note that with gamma errors, the scale factor (ν^{-1}) can be input or estimated using the degrees of freedom, k, and

$$\hat{\nu}^{-1} = \sum_{i=1}^{n} \frac{[(y_i - \hat{\mu}_i]/\hat{\mu}_i]^2}{N - k}.$$

$$\text{gamma: } d_i = 2\left\{\log(\hat{\mu}_i) + \left(\frac{y_i}{\hat{\mu}_i}\right)\right\}$$

$$r_i = \frac{3(y_i^{\frac{1}{3}} - \hat{\mu}_i{}^{\frac{1}{3}})}{\hat{\mu}_i{}^{\frac{1}{3}}}$$

$$DEV = \sum_{i=1}^{n} d_i$$

```
Number of parameters = 2, Rank = 2, Number of points = 10, NDOF = 8
 Parameter        Value      95% conf. limits       Std.error     p
  Constant        1.441    -0.08812          2.97       0.663   0.0615   *
      B(1)       -1.287      -2.824        0.2513      0.6669   0.0898   *
Adjusted Deviance = 3.503E+01, S = 1.074E+00, A = 1.000E+00

Number      Y-value      Theory    Dev-resid     Leverage
     1          1.0        6.48       -1.391          0.2
     2          0.3        6.48       -1.923          0.2
     3         10.5        6.48       0.5236          0.2
     4          9.7        6.48       0.4318          0.2
     5         10.9        6.48       0.5678          0.2
     6         0.62       0.694      -0.1107          0.2
     7         0.12       0.694       -1.329          0.2
     8         0.09       0.694       -1.482          0.2
     9          0.5       0.694        -0.31          0.2
    10         2.14       0.694        1.366          0.2
```

Table 5.6: GLM example 4: gamma errors

5.3 The SIMFIT simplified Generalized Linear Models interface

Although generalized linear models have widespread use, specialized knowledge is sometimes required to prepare the necessary data files, weights, offsets, etc. For this reason, there is a simplified SIMFIT interface to facilitate the use of GLM techniques in such fields as the following.

- Bioassay, assuming a binomial distribution and using logistic, probit, or log-log models to estimate percentiles, such as the LD50 (page 110).
- Logistic regression and binary logistic regression.
- Logistic polynomial regression, generating new variables interactively as powers of an original covariate.
- Contingency table analysis, assuming Poisson errors and using log-linear analysis to quantify row and column effects (page 143).
- Survival analysis, using the exponential, Weibull, extreme value, and Cox (i.e., proportional hazard) models (page 253).

Of course, by choosing the advanced interface, users can always take complete control of the GLM analysis, but for many purposes the simplified interface will prove much easier to use for many routine applications. Some applications of the simplified interface will now be presented.

5.4 Logistic regression

Logistic regression is an application of the previously discussed GLM procedure assuming binomial errors and a logistic link. It is widely used in situations where there are binary variables and estimates of odds ratios or log odds ratios are required. A particularly useful application is in binary logistic regression where the y_i values are all either 0 or 1 and all the N_i values are equal to 1, so that a probability $\hat{p}_i$ is to be estimated as a function of some variables. Frequently the covariates are qualitative variables which can be included in the model by defining appropriate dummy indicator variables. For instance, suppose a factor has m levels, then we can define m dummy indicator variables $x_1, x_2, \dots, x_m$ as in Table 5.7. The data file would be set up as if to estimate all m parameters for the m factor levels but because only $m - 1$ of the dummy indicator variables

Level	x_1	x_2	x_3	...	x_m
1	1	0	0	...	0
2	0	1	0	...	0
3	0	0	1	...	0
...	...	...	...	...	...
m	0	0	0	...	1

Table 5.7: Dummy indicators for categorical variables

are independent, one of them would have to be suppressed if a constant were to be fitted, to avoid aliasing, i.e., the model would be overdetermined and the parameters could not be estimated uniquely. Suppose, for instance, that the model to be fitted was for a factor with three levels, i.e.,

$$\log\left\{\frac{E(y)}{1-E(y)}\right\} = a_0 + a_1x_1 + a_2x_2 + a_3x_3$$

but with x_1 suppressed. Then the estimated parameters could be interpreted as log odds ratios for the factor levels with respect to level 1, the suppressed reference level. This is because for probability estimates $\hat{p}_1$, $\hat{p}_2$ and $\hat{p}_3$ we would have the odds estimates

$$\begin{aligned}\frac{\hat{p}_1}{1-\hat{p}_1} &= \exp(a_0)\\ \frac{\hat{p}_2}{1-\hat{p}_2} &= \exp(a_0+a_2)\\ \frac{\hat{p}_3}{1-\hat{p}_3} &= \exp(a_0+a_3)\end{aligned}$$

and estimates for the corresponding log odds ratios involving only the corresponding estimated coefficients

$$\begin{aligned}\log\left\{\frac{\hat{p}_2/(1-\hat{p}_2)}{\hat{p}_1/(1-\hat{p}_1)}\right\} &= a_2\\ \log\left\{\frac{\hat{p}_3/(1-\hat{p}_3)}{\hat{p}_1/(1-\hat{p}_1)}\right\} &= a_3.\end{aligned}$$

Even with quantitative, i.e., continuous data, the best-fit coefficients can always be interpreted as estimates for the log odds ratios corresponding to unit changes in the related covariates.

As an example of simple binary logistic regression, fit the data in test file `logistic.tf1` to obtain the results shown in table 5.8. The parameters are well determined and the further step was taken to calculate an expected frequency, given the parameter estimates. It frequently happens that, after fitting a data set, users wish to predict the binomial probability using the parameters estimated from the sample. That is, given the model

$$\begin{aligned}\log\left(\frac{E(y)}{1-E(y)}\right) &= \beta_0 + \beta_1x_1 + \beta_2x_2 + \ldots + \beta_mx_m\\ &= \eta,\end{aligned}$$

where y is recorded as either 0 (failure) or 1 (success) in a single trial, then the binomial probability p would be estimated as

$$\hat{p} = \frac{\exp(\hat{\eta})}{1+\exp(\hat{\eta})},$$

where $\hat{\eta}$ is evaluated using parameter estimates with user supplied covariates. In this case, with a constant term and $x_1 = x_2 = 1$, then $\hat{p} = 0.04761$.

```
 Number of parameters = 3, Rank = 3, Number of points = 39, NDOF = 36
 Parameter       Value     95% conf. limits      Std.error    p
  Constant      -9.520     -16.06      -2.981       3.224   0.0055
     B(1)        3.877     0.9868       6.768       1.425   0.0100
     B(2)        2.647     0.7975       4.496      0.9119   0.0063
Deviance = 2.977E+01

x(0) = 1.0, coefficient = -9.520 (the constant term)
x(1) = 1.0, coefficient =  3.877
x(2) = 1.0, coefficient =  2.647
Binomial N = 1
y(x) =  4.761E-02, Binomial probability p = 0.04761
```

Table 5.8: Binary logistic regression

5.5 Conditional binary logistic regression with stratified data

A special case of multivariate conditional binary logistic regression is in matched case control studies, where the data consist of strata with cases and controls, and it is wished to estimate the effect of covariates, after allowing for differing baseline constants in each stratum. Consider, for example, the case of s strata with n_k cases and m_k controls in the kth stratum. Then, for the jth person in the kth stratum with c-dimensional covariate vector x_{jk}, the probability $P_k(x_{jk})$ of being a case is

$$P_k(x_{jk}) = \frac{\exp(\alpha_k + \beta^T x_{jk})}{1 + \exp(\alpha_k + \beta^T x_{jk})}$$

where α_k is a stratum specific constant. Estimation of the c parameters β_i can be accomplished by maximizing the conditional likelihood, without explicitly estimating the constants α_k.

As an example, fit the test file `strata.tf1` to obtain the results shown in table 5.9. Note that the input file

```
Number of parameters = 2, Number of points = 7, NDOF = 5
 Parameter       Value     95% conf. limits      Std.error    p
     B(1)      -0.5223     -4.096       3.051        1.39   0.7226 ***
     B(2)      -0.2674     -2.446       1.911      0.8473   0.7651 ***
 Deviance = 5.475E+00

    Strata     Cases  Controls
         1         2         2
         2         1         2
```

Table 5.9: Conditional binary logistic regression

must use the s variable, i.e. the last column in the data file, to indicate the stratum to which the observation corresponds, and since the model fitted is incomplete, goodness of fit analysis is not available.

Part 6

Nonlinear models: Simple fitting

6.1 Introduction

Linear regression is trivial and gives unique solutions, but constrained nonlinear regression is extremely complicated and does not give unique solutions. So you might ask: *Why bother with nonlinear regression ?* The answer is that nature is nonlinear, so nonlinear regression is the only approach open to honest investigators. Sometimes it is possible to transform data and use linear techniques, as in generalized linear interactive modelling (GLM), but this just bypasses the central issue; finding a mathematical model derived using established physical laws, and involving constants that have a well defined physical meaning. Logistic regression, for instance, involving fitting a polynomial to transformed data, may seem to work; but the polynomial coefficients have no meaning. Estimating rate constants, on the other hand, allows comparisons to be made of kinetic, transport or growth processes under different treatments, and helps to explain experimental results in terms of processes such as diffusion or chemical reaction theory.

Nonlinear regression involves the following steps.

1. Obtaining data for responses $y_i, i = 1, 2, \ldots, n$ at exactly known values of fixed variables x_i.
2. Estimating weighting factors $w_i = 1/s_i^2$, where the s_i are standard deviations, or smoothed estimates, obtained by analyzing the behaviour of replicates at the fixed x_i values if possible, or $s_i = 1$ otherwise.
3. Selecting a sensible deterministic model from a set of plausible mathematical models for
$$y_i = f(x_i, \mathbf{\Theta}) + \epsilon_i,$$
where $\mathbf{\Theta} = \theta_1, \theta_2, \ldots, \theta_k$ are parameters, and ϵ_i are uncorrelated errors with zero mean.
4. Choosing meaningful starting parameter estimates for the unknown parameter vector $\mathbf{\Theta}$.
5. Normalizing the data so that internal parameters, objective function and condition number of the Hessian matrix are of order unity (in internal coordinates) at the solution point.
6. Assessing goodness of fit by examining the weighted residuals
$$r_i = (y_i - f(x_i, \hat{\mathbf{\Theta}}))/s_i$$
where $\hat{\mathbf{\Theta}}$ is the best fit parameter vector.
7. Investigating parameter redundancy by examining the weighted sum of squared residuals $WSSQ$
$$WSSQ = \sum_{i=1}^{n} r_i^2,$$
and the estimated parameter variance-covariance matrix.

Curve fitting is controversial, so the SIMFIT philosophy will be stated at this point.

Weighting should only be attempted by users who have at least four replicates per design point and are prepared to investigate the relative effects of alternative weighting schemes.
Caution is needed when interpreting goodness of fit statistics, and users should demand convincing evidence before concluding that models with more than say four parameters are justified.
If there are no parameter constraints, a modified Gauss-Newton or Levenburg-Marquardt method can be used, but if constraints are required a sequential quadratic programming or quasi-Newton method should be used.
You must have good data over a wide range to define asymptotes etc., fit all replicates, not means, and use sensible models and starting estimates.

6.1.1 User friendly curve fitting programs

Unfortunately, x_i cannot be fixed exactly, w_i have to be estimated, we are never certain that $f(.)$ is the correct model, experimental errors are not uncorrelated and normally distributed, and $WSSQ$ minimization is is not guaranteed to give a unique or sensible solution with nonlinear models. Nevertheless SIMFIT has these linear and nonlinear regression programs that greatly simplify model fitting and parameter estimation.

linfit	linear/multi-linear regression and generalized linear modelling (GLM)
exfit	sum of exponentials, choosing from 6 possible types (unconstrained)
gcfit	growth models (exponential, monomolecular, Richards, Von Bertalanffy, Gompertz, Logistic, Preece-Baines) with or without constant terms (unconstrained)
hlfit	sum of high/low affinity ligand binding sites with a constant term (constrained)
mmfit	sum of Michaelis-Menten functions (constrained)
polnom	polynomials (Chebyshev) in sequence of increasing degree (unconstrained)
rffit	positive $n : n$ rational functions (constrained)
sffit	saturation function for positive or negative binding cooperativity (constrained)
csafit	flow cytometry histograms with stretch and shift (constrained)
inrate	Hill-n/Michaelis-Menten/line/quadratic/lag-phase/monomolecular (unconstrained)

The user-friendly nonlinear regression programs calculate starting estimates, scale data into internal coordinates, then attempt to minimize the objective function $WSSQ/NDOF$, which has expectation 1 with correct model and weights. However, if incorrect models or weights are used, or $WSSQ/NDOF \ll$ 1.0E-6, or $\gg$ 1.0E6, the programs may not converge. If you have insufficient replicates to estimate weights and have set $s = 1$, the programs do unweighted regression, replacing the chi-square test by calculation of the average $cv\%$

$$cv\% = 100\frac{\sqrt{WSSQ/NDOF}}{(1/n)\sum_{i=1}^{n}|y_i|}, \text{ and } NDOF = n - \text{no. of parameters.}$$

These programs must be supplied with all observations, not means of replicates or else biased statistics will be output and, after fitting, options are available to plot residuals, identify outliers, calculate error bars interactively from groups of replicates (arranged in nondecreasing order), etc.

6.1.2 IFAIL and IOSTAT error messages

As curve fitting is iterative you are likely to encounter error messages when curve fitting as follows. IFAIL errors flag computational failure, and will occur if the data lead to a singularity in evaluating some expression. For instance, the formula $y = 1/x$ will lead to overflow when x becomes so small that y would exceed the largest number allowed on your computer. IFAIL messages look like this:

```
FATAL : IFAIL = 1 from C05AZF/ZSOLVE.
```

which means that a fault leading to `IFAIL = 1` on exit from `C05AZF` has occurred in subroutine `ZSOLVE`. The order of severity of SIMFIT error messages is

```
ADVICE < CAUTION < WARNING « FATAL
```

then self-explanatory text. If a nonzero IFAIL value is returned and you want to know what it means, you can look it up in the NAG library handbook by simply searching for the routine on the web. For instance, searching for C05AZF should lead you to an appropriate web document, e.g.,
`http://www.nag.co.uk/numeric/fl/manual/C05/C05azf.pdf`,
where you can find out what C05AZF does and what the IFAIL message indicates.

IOSTAT errors flag input or output failure, and will occur if a program is unable to read data correctly from files, e.g., because the file is formatted incorrectly, end-of-file has been encountered leading to negative IOSTAT, or data has become corrupted, e.g., with letters instead of numbers.

The following sections take each of the user friendly programs in turn and suggest ways you can practise with the test files. Finally we briefly turn to specialized models, and comprehensive curve fitting for experts.

6.2 Exponential functions

Graphs for the exponential function

$$f(t) = A_1 \exp(-k_1 t) + A_2 \exp(-k_2 t) + \cdots + A_n \exp(-k_n t) + C$$

are shown in figure 6.1. Note that all these curves can be fitted by **exfit**, but with different strategies for initial

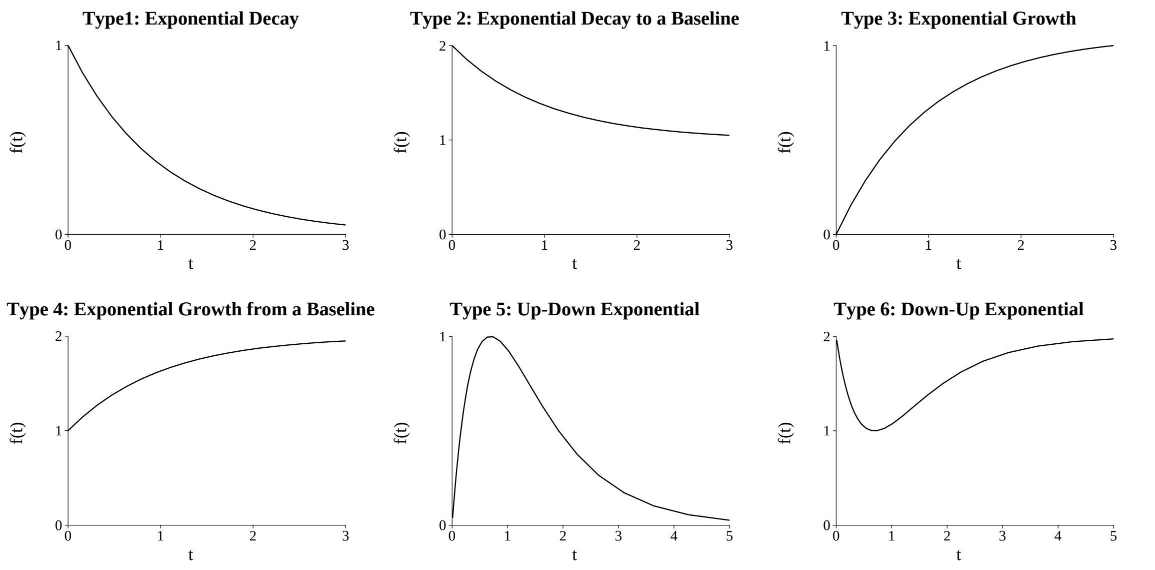


Figure 6.1: Alternative types of exponential functions

parameter estimates and scaling, depending on curve type.

Types 1 and 2 are the familiar exponential declines to zero or an asymptote that are often used for pharmacokinetic elimination studies.

Types 3 and 4 are typically used for growth or accumulation situations.

Types 5 and 6 are used in absorption elimination or chemical intermediate studies and require at least two exponentials. For intermediate cases where compartments are not empty or chemical species are not zero at zero time, and it usual to follow fitting these models by a relaxation step to accommodate variants of the schemes illustrated.

The test file `exfit.tf4` has data for the function

$$f(t) = \exp(-t) + \exp(-5t)$$

obtained by using **adderr** to add error to the exact data in `exfit.tf3` prepared by **makdat**. So you read this data into **exfit**, select models of type 1, and then request **exfit** to fit 1 exponential then 2 exponentials, using a short random search. The result is shown in figure 6.2, namely, the fit with two exponentials is sufficiently better than the fit with 1 exponential that we can assume an acceptable model to be

$$f(t) = A_1 \exp(-k_1 t) + A_2 \exp(-k_2 t).$$

Now do it again, but this time pay more attention to the goodness of fit criteria, residuals, parameter estimates and statistical analysis. Get the hang of the way SIMFIT does goodness of fit, residuals display, graph plotting and statistics, because all the curve fitting programs adopt a similar approach. As for the other test files, `exfit.tf1` and `exfit.tf2` are for 1 exponential, while `exfit.tf5` and `exfit.tf6` are double exponentials for models 5 and 6. Linked sequential exponential models should be fitted by program **qnfit** not program **exfit**, since the time constants and amplitudes are not independent in such cases. Program **exfit** can also be used in pharmacokinetics to estimate time to half maximum response and AUC.

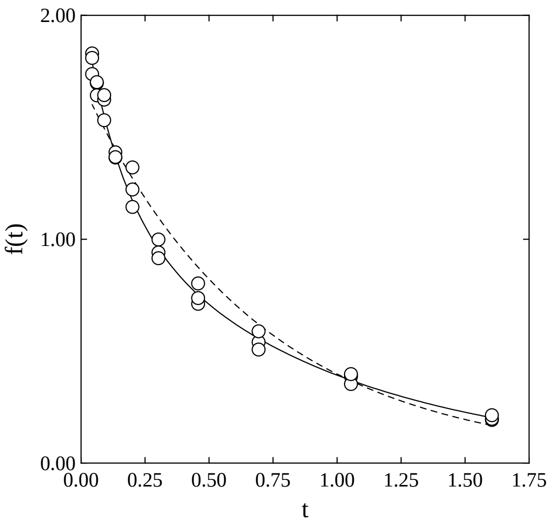


Figure 6.2: Fitting exponential functions

6.3 How to interpret parameter estimates

The meaning of the results generated by program **exfit** after fitting two exponentials to `exfit.tf4` will now be explained, as a similar type of analysis is generated by all the user-friendly curve fitting programs. Consider, first of all Table 6.1 listing parameter estimates which result from fitting two exponentials. The first column

```
Parameter      Value       Std.Error    Lower95%cl    Upper95%cl     p
   A(1)      0.852552     0.0677105     0.713371      0.991733    0.0000
   A(2)       1.17644     0.0747539      1.02278        1.3301    0.0000
   k(1)       6.79336      0.854392      5.03713       8.54959    0.0000
   k(2)       1.11206     0.0510998      1.00702       1.21709    0.0000
   AUC        1.18339     0.0147104      1.15316       1.21363    0.0000
AUC is the area under the curve from t = 0 to t = infinity
Initial time point (A)   = 0.035983
Final time point   (B)   = 1.611
Area over range (A,B)    = 0.938322
Average over range (A,B) = 0.595754
```

Table 6.1: Fitting two exponentials: 1. parameter estimates

gives the estimated values for the parameters, i.e., the amplitudes $A(i)$ and decay constants $k(i)$, although it must be appreciated that the pairwise order of these is arbitrary. Actually program **exfit** will always try to rearrange the output so that the amplitudes are in increasing order, and a similar rearrangement will also occur with programs **mmfit** and **hlfit**. For situations where $A(i) > 0$ and $k(i) > 0$, the area from zero to infinity, i.e.

the AUC, can be estimated, as can the area under the data range and the average function value (page 268) calculated from it. The parameter AUC is not estimated directly from the data, but is a secondary parameter estimated algebraically from the primary parameters. The standard errors of the primary parameters are obtained from the inverse of the estimated Hessian matrix at the solution point, but the standard error of the AUC is estimated from the partial derivatives of AUC with respect to the primary parameters, along with the estimated variance-covariance matrix (page 116). The 95% confidence limits are calculated from the parameter estimates and the t distribution (page 422), while the p values are the two-tail probabilities for the estimates, i.e., the probabilities that parameters as extreme or more extreme than the estimated ones could have resulted if the true parameter values were zero. The windows defined by the confidence limits are useful for a quick rule of thumb comparison with windows from fitting the same model to another data set; if the windows are disjoint then the corresponding parameters differ significantly, although there are more meaningful tests (page 36). Clearly, parameters with $p < 0.05$ are well defined, while parameters with $p > 0.05$ must be regarded as ill-determined.

Expert users may sometimes need the estimated correlation matrix

$$C_{ij} = \frac{CV_{i,j}}{\sqrt{CV_{ii}CV_{jj}}},$$

where $-1 \le C_{ij} \le 1$, $C_{ii} = 1$, which is shown in Table 6.2

```
Parameter correlation matrix
 1.0000
-0.8757  1.0000
-0.5962  0.8995  1.0000
-0.8478  0.9485  0.8199  1.0000
```

Table 6.2: Fitting two exponentials: 2. correlation matrix

6.4 How to interpret goodness of fit

Table 6.3, displaying the results from analyzing the residuals after fitting two exponentials to `exfit.tf4`, is typical of many SIMFIT programs. Residuals tables should always be consulted when assessing goodness of fit. Several points should be remembered when assessing such residuals tables, where there are N observations y_i, with weighting factors s_i, theoretical values $f(x_i)$, residuals $r_i = y_i - f(x_i)$, weighted residuals r_i/s_i, and where k parameters have been estimated.

- The chi-square test (page 143) using

$$WSSQ = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i)}{s_i} \right)^2$$

  is only meaningful if the weights defined by the s_i supplied for fitting are good estimates of the standard deviations of the observations at that level of the independent variable; say means of at least five replicates. Inappropriate weighting factors will result in a biased chi-square test. Also, if all the s_i are set equal to 1, unweighted regression will be performed and an alternative analysis based on the coefficient of variation will be performed.

- The R^2 value is the square of the correlation coefficient (page 190) between data and best fit points. It only represents a meaningful estimate of that proportion of the fit explained by the regression for simple unweighted linear models, and should be interpreted with restraint when nonlinear models have been fitted.

```
Analysis of weighted residuals: WSSQ          = 24.397
p = P(chi-sq. >= WSSQ)                        = 0.5533
R-squared, [corr.coeff.(best-fit,observed)]^2 = 0.9934
Largest  absolute relative residual           =  11.99%
Smallest absolute relative residual           =   0.52%
Average  absolute relative residual           =   3.87%
Absolute relative residuals in range 0.1-0.2  =   3.33%
Absolute relative residuals in range 0.2-0.4  =   0.00%
Absolute relative residuals in range 0.4-0.8  =   0.00%
Absolute relative residuals > 0.8             =   0.00%
Number of negative residuals (m)              = 15
Number of positive residuals (n)              = 15
Number of runs observed (r)                   = 16
p = P(runs =< r : given m and n)              = 0.5759
5% lower tail point                           = 11
1% lower tail point                           = 9
p = P(runs =< r : given m plus n)             = 0.6445
p = P(signs =< least number observed)         = 1.0000
Durbin-Watson test statistic                  = 1.8061
Shapiro-Wilks W statistic                     = 0.9387
p = Significance level of W                   = 0.0840
Akaike AIC (Schwarz SC) statistics            = 1.79794 (7.40273)
Verdict on goodness of fit                    = incredible
```

Table 6.3: Fitting two exponentials: 3. goodness of fit statistics

- The results based on the absolute relative residuals a_i defined using machine precision ϵ as

$$a_i = \frac{2|r_i|}{\max(\epsilon, |y_i| + |f(x_i)|)}$$

do not have statistical relevance, but they do have obvious empirical justification, and they must be interpreted with commonsense, especially where the data and/or theoretical values are very small.

- The probability of the number of runs observed given m negative and n positive residuals is a very useful test for randomly distributed runs (page 153), but the probability of runs given $N = m + n$, and also the overall sign test (page 152) are weak, except for very large data sets.

- The Durbin-Watson test statistic

$$DW = \frac{\sum_{i=1}^{N-1} (r_{i+1} - r_i)^2}{\sum_{i=1}^{N} r_i^2}$$

is useful for detecting serially correlated residuals, which could indicate correlated data or an inappropriate model. The expected value is 2.0, and values less than 1.5 suggest positive correlation, while values greater than 2.5 suggest negative serial correlation.

- Where N, the number of data points, significantly exceeds k, the number of parameters estimated, the weighted residuals are approximately normally distributed, and so the Shapiro-Wilks test (page 133) should be taken seriously.

- The Akaike AIC statistic

$$AIC = N \log(WSSQ/N) + 2k$$

and Schwarz Bayesian criterion SC

$$SC = N\log(WSSQ/N) + k\log N$$

are only really meaningful if minimizing $WSSQ$ is equivalent to Maximum Likelihood Estimation. Note that only differences between AIC with the same data, i.e. fixed N, are relevant, as in the evidence ratio ER, defined as $ER = \exp[(AIC(1) - AIC(2))/2]$.

- The final verdict is calculated from an empirical look-up table, where the position in the table is a weighted mean of scores allocated for each of the tests listed above. It is qualitative and rather conservative, and has no precise statistical relevance, but a good result will usually indicate a well-fitting model.
- As an additional measure, plots of residuals against theory, and half-normal residuals plots (figure10.1) can be displayed after such residuals analysis, and they should always be inspected before concluding that any model fits satisfactorily.
- With linear models, SɪᴍFIT also calculates studentized residuals and leverages, while with generalized linear models (page 49), deviance residuals can be tabulated.

6.5 How to interpret model discrimination results

After a sequence of models have been fitted, tables like Table 6.4 are generated. First of all, note that the above

```
WSSQ-previous (WSSQ1)               = 224.923
WSSQ-current (WSSQ2)                = 24.397
Number of parameters-previous (M1)  = 2
Number of parameters-current (M2)   = 4
Number of data points (NPTS)        = 30
Akaike AIC-previous                 = 64.4368
Akaike AIC-current                  = 1.79794
Evidence ratio (ER)                 = 3.99817E+13
Schwarz SC-previous                 = 67.2392
Schwarz SC-current                  = 7.40273
Mallows Cp                          = 213.701
Mallows ratio (Cp/M1)               = 106.851
Numerator degrees of freedom        = 2
Denominator degrees of freedom      = 26
F test statistic (FS)               = 106.851
p = P(F >= FS)                      = 0.0000
1 - p = P(F =< FS)                  = 1.0000
5% upper tail point                 = 3.36902
1% upper tail point                 = 5.52633

Conclusion based on F test
Reject previous model at 1% significance level
There is strong support for the extra parameters
Tentatively accept the current best fit model
```

Table 6.4: Fitting two exponentials: 4. model discrimination statistics

model discrimination analysis is only strictly applicable for nested linear models with known error structure, and should be interpreted with restraint otherwise. Now, if $WSSQ_1$ with m_1 parameters is the previous (possibly deficient) model, while $WSSQ_2$ with m_2 parameters is the current (possibly superior) model, so that $WSSQ_1 > WSSQ_2$, and $m_1 < m_2$, then

$$F = \frac{(WSSQ_1 - WSSQ_2)/(m_2 - m_1)}{WSSQ_2/(N - m_2)}$$

should be F distributed (page 423) with $m_2 - m_1$ and $N - m_2$ degrees of freedom, and the F test (page 155) for excess variance can be used. Alternatively, if $WSSQ2/(N - m_2)$ is equivalent to the true variance, i.e., model 2 is equivalent to the true model, the Mallows C_p statistic

$$C_p = \frac{WSSQ_1}{WSSQ_2/(N - m_2)} - (N - 2m_1)$$

can be considered. This has expectation m_1 if the previous model is sufficient, so values greater than m_1, that is $C_p/m_1 > 1$, indicate that the current model should be preferred over the previous one. However, graphical deconvolution, as illustrated on page 38, should always be done wherever possible, as with sums of exponentials, Michaelis-Mentens, High-Low affinity sites, sums of Gaussians or trigonometric functions, etc., before concluding that a higher order model is justified on statistical grounds.

6.6 High/low affinity ligand binding sites

The binding of ligands to receptors can be defined in terms of a binding polynomial $p(x)$ in the free ligand activity x, as follows

$$\begin{aligned} p(x) &= 1 + K_1x + K_2x^2 + \cdots + K_nx^n \\ &= 1 + A_1x + A_1A_2x^2 + \cdots + \prod_{i=1}^{n} A_ix^n \\ &= 1 + \binom{n}{1}B_1x + \binom{n}{2}B_1B_2x^2 + \cdots + \binom{n}{n}\prod_{i=1}^{n} B_ix^n, \end{aligned}$$

where the only difference between these alternative expressions concerns the meaning and interpretation of the binding constants. The fractional saturation is just the scaled derivative of the log of the polynomial with respect to $\log(x)$. If the binding polynomial has all real factors, then the fractional saturation y as a function of free ligand is indistinguishable from independent high/low affinity sites or uniformly negative cooperativity with Hill slope H everywhere less than or equal to unity.

So, as the high/low sites model is just a weighted sum of 1-site models, the general model for a mixture of n high/low affinity sites is

$$f(x) = \frac{a_1K_1x}{1 + K_1x} + \frac{a_2K_2x}{1 + K_2x} + \cdots + \frac{a_nK_nx}{1 + K_nx} + C$$

but usually it is only possible to differentiate between the cases $n = 1$ and $n = 2$. The test files `hlfit.tf1` and `hlfit.tf2` are for 1 site, while `hlfit.tf3` has data for 2 sites and `hlfit.tf4` has the same data with added error. When fitting this model you should normalize the data if possible to zero baseline, that is $f(0) = 0$, or alternatively $C = 0$, and explore whether two independent sites give a better fit than one site. So, read `hlfit.tf4` into program **hlfit**, ask for lowest order 1, highest order 2 and the case where C is not varied but is fixed at $C = 0$. The outcome is

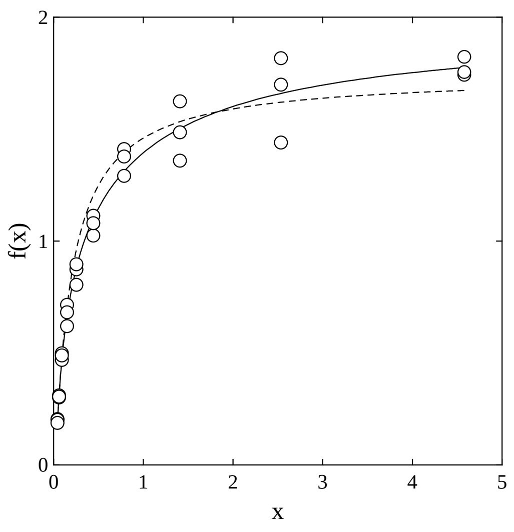


Figure 6.3: Fitting high/low affinity sites

illustrated in figure 6.3 and, from the statistics, you will learn that independent low and high affinity sites are justified in this case. To interpret the parameter estimates, you take the values for K_1 and K_2 as estimates for the respective association constants, and a_1 and a_2 as the relative number of sites in the respective categories. The total number of sites is proportional to $a_1 + a_2$, and has got nothing to do with n, which is the number of distinct binding types that can be deduced from the binding data. Concentration at half saturation is also estimated by **hlfit**, but cooperative binding should be fitted by program **sffit**.

6.7 Cooperative ligand binding

If the n zeros of the binding polynomial are α_i then the fractional saturation y can be expressed as

$$y = \left(\frac{x}{n}\right) \sum_{i=1}^{n} \frac{1}{x - \alpha_i},$$

but further discussion depends on the nature of the zeros.

First observe that, for a set of m groups of receptors, each with n_i independent binding sites and binding constant k_i, then the zeros are all real and

$$p(x) = \prod_{i=1}^{m} (1 + k_i x)^{n_i},$$

$$\text{and } y = \frac{1}{\sum_{i=1}^{m} n_i} \sum_{i=1}^{m} \frac{n_i k_i x}{1 + k_i x},$$

so y is just the sum of simple binding curves, giving concave down double reciprocal plots, etc.

However, if the binding polynomial has complex conjugate zeros, the Hill slope may exceed unity and there may be evidence of positive cooperativity. The way to quantify the sign of cooperativity is to fit the appropriate order n saturation function $f(x)$ to the binding data, i.e.,

$$f(x) = Zy + C,$$

$$\text{where } y = \left(\frac{1}{n}\right) \frac{d \log(p(x))}{d \log(x)}$$

to determine the binding constants, where Z accounts for proportionality between site occupation and response, and C is a background constant. Note that the Hill slope cannot exceed the Hill slope of any of the factors of the binding polynomial, so further calculations are required to see if the binding data show evidence of positive or negative cooperativity.

To use **sffit** you should really have some idea about the total number of binding sites on the macromolecule or receptor, i.e., n, and suspect cooperative interaction between the sites, i.e., if **hlfit** cannot fit the data. The appropriate model for cooperative ligand binding to macromolecules is

$$f(x) = \frac{Z(\phi_1 x + 2\phi_2 x^2 + \cdots + n\phi_n x^n)}{n(1 + \phi_1 x + \phi_2 x^2 + \cdots + \phi_n x^n)} + C$$

where Z is a scaling factor and C is a baseline correction. In this formulation, the ϕ_i are overall binding constants, but the alternative definitions for binding constants, and the convention for measuring deviations from noncooperative binding in terms of the Hessian of the binding polynomial, are in the tutorial. Test files for program **sffit** are `sffit.tf1` and `sffit.tf2` for 1 site and `sffit.tf3` and `sffit.tf4` for 2 sites, and note that concentration at half saturation is also estimated. Always try to normalize your data so that $Z = 1$ and $C = 0$. Such normalizing is done by a combination of you finding the normalizing parameters independently, and/or using a preliminary fit to estimate them, followed by scaling your data using **editfl**.

6.8 Cooperativity analysis

After fitting a model, program **sffit** outputs the binding constant estimates in all the conventions and, when $n > 2$ it also outputs the zeros of the best fit binding polynomial and those of the Hessian of the binding polynomial $h(x)$, defined as

$$h(x) = np(x)p''(x) - (n-1)p'(x)^2$$

since it is at positive zeros of the Hessian that cooperativity changes take place. This because the Hill slope H is the derivative of the log odds with respect to chemical potential, i.e.,

$$\begin{aligned} H &= \frac{d\log[y/(1-y)]}{d\log(x)} \\ &= 1 + \frac{xh(x)}{p'(x)\,(np(x) + xp'(x))} \end{aligned}$$

and positive zeros of $h(x)$ indicate points where the theoretical one-site binding curve coinciding with the actual saturation curve at that x value has the same slope as the higher order saturation curve, which are therefore points of cooperativity change. The SimFIT cooperativity procedure allows users to input binding constant estimates retrospectively to calculate zeros of the binding polynomial and Hessian, and also to plot species fractions.

6.9 Ligand binding species fractions

The species fractional populations s_i which are defined for $i = 0, 1, \ldots, n$ as

$$s_i = \frac{K_i x^i}{K_0 + K_1 x + K_2 x^2 + \cdots + K_n x^n}$$

with $K_0 = 1$, are interpreted as the proportions of the receptor in the various states of ligation as a function of ligand activity. The species fractions defined as $y_i = is_i/n$ for $i = 1, 2, \ldots, n$ are the contributions of the species to the overall saturation. Note that $\sum_{i=0}^{n} s_i = 1$ while $\sum_{i=1}^{n} y_i = (1/n)d\log p/d\log x$.

Such expressions are very useful when analyzing cooperative ligand binding data as in figure 6.4. They can be generated from the best fit binding polynomial after fitting binding curves with program **sffit**, or by input of binding constants into program **simstat**. At the same time other important analytical results like factors of the Hessian and minimax indexHill slope Hill slope are also calculated. The species fractional populations can be also used in a probability model to interpret ligand binding in several interesting ways. For this purpose, consider a random variable U representing the probability of a receptor existing in a state with i ligands bound. Then the the probability mass function, expected values and variance are

$$\begin{aligned} P(U = i) &= s_i \;(i = 0, 1, 2, \ldots, n), \\ E(U) &= \sum_{i=0}^{n} is_i, \\ E(U^2) &= \sum_{i=0}^{n} i^2 s_i, \\ V(U) &= E(U^2) - [E(U)]^2 \\ &= x\left(\frac{p'(x) + xp''(x)}{p(x)}\right) - \left(\frac{xp'(x)}{p(x)}\right)^2 \\ &= n\frac{dy}{d\log x}, \end{aligned}$$

as fractional saturation y is $E(U)/n$. In other words, the slope of a semi-log plot of fractional saturation data indicates the variance of the number of occupied sites, namely; all unoccupied when $x = 0$, distribution with

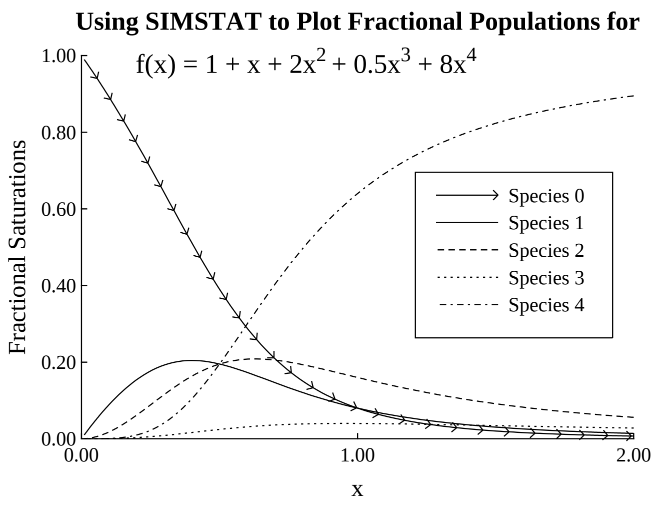


Figure 6.4: Ligand binding species fractional populations

variance increasing as a function of x up to the maximum semi-log plot slope, then finally approaching all sites occupied as x tends to infinity. You can input binding constants into the statistical calculations procedure to see how they are mapped into all spaces, cooperativity coefficients are calculated, zeros of the binding polynomial and Hessian are estimated, Hill slope is reported, and species fractions and binding isotherms are displayed, as is done automatically after every $n > 1$ fit by program **sffit**.

6.10 Michaelis-Menten kinetics

A mixture of independent isoenzymes, each separately obeying Michaelis-Menten kinetics is modeled by

$$v(S) = \frac{V_{max_1} S}{K_{m_1} + S} + \frac{V_{max_2} S}{K_{m_2} + S} + \cdots + \frac{V_{max_n} S}{K_{m_n} + S}$$

and again only the cases $n = 1$ and $n = 2$ need to be considered. The appropriate test files are `mmfit.tf1` and `mmfit.tf2` for one enzyme, but `mmfit.tf3` and `mmfit.tf4` for 2 isoenzymes. Read `mmfit.tf4` into **mmfit** and decide if 1 or 2 enzymes are needed to explain the data. There is a handy rule of thumb that can be used to decide if any two parameters are really different (page 36). If the 95% confidence limits given by SIMFIT do not overlap at all, it is very likely that the two parameters are different. Unfortunately, this test is very approximate and nothing can be said with any certainty if the confidence limits overlap. When you fit `mmfit.tf4` try to decide if V_{max_1} and V_{max_2} are different. What about K_{m_1} and K_{m_2} ? Program **mmfit** also estimates concentration at half maximum response, i.e., $EC50$ and $IC50$ (page 113), but **rffit**. should be used if the data show substrate activation or substrate inhibition.

6.10.1 Extrapolating Michaelis-Menten kinetics

Best fit curves from **qnfit** can be extrapolated beyond data limits but more complicated extrapolation is sometimes required. For instance, fitting the mixed, or noncompetitive inhibition model

$$v(S, I) = \frac{VS}{K(1 + I/K_{is}) + (1 + I/K_{ii})S}$$

as a function of two variables is straightforward but, before the computer age, people used to fit this sort of model using linear regression to the double reciprocal form

$$\frac{1}{v} = \frac{1}{V}\left(1 + \frac{I}{K_{ii}}\right) + \frac{K}{V}\left(1 + \frac{I}{K_{is}}\right)$$

which is still sometimes used to demonstrate the intersection point of the best-fit lines. Figure 6.5 was obtained by fitting the mixed model to `inhibit.tf1`, plotting sections through the best-fit surface at fixed inhibitor

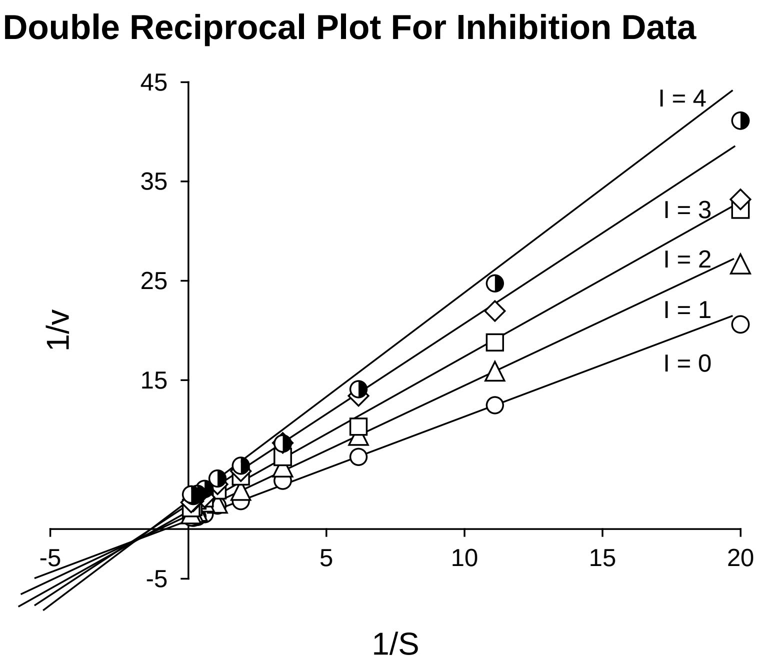


Figure 6.5: Extrapolation

concentration, and saving these as files. The files (referenced by the library file `inhibit.tfl`) were plotted in double reciprocal space using **simplot**, and figure 6.5 was created by overlaying extra lines over each best-fit line and extending these beyond the fixed point at

$$\frac{1}{S} = -\frac{1}{K}\left(\frac{K_{is}}{K_{ii}}\right), \frac{1}{v} = \frac{1}{V}\left(1 - \frac{K_{is}}{K_{ii}}\right).$$

Original best-fit lines were then suppressed, and other cosmetic changes were implemented. For obvious mathematical reasons, extrapolation for this model in transformed space cannot be generated by requesting an extended range for a best-fit curves in the original space. To avoid extrapolated lines passing through plotting symbols, the option to plot extra lines in the background, i.e., before the data, was selected.

6.11 Isotope displacement kinetics

The rational function models just discussed for binding and kinetics represent the saturation of binding sites, or flux through active sites, and special circumstances apply when there is no appreciable kinetic isotope effect. That is, the binding or kinetic transformation process is the same whether the substrate is labeled or not. This allows experiments in which labeled ligand is displaced by unlabeled ligand, or where the flux of labeled substrate is inhibited by unlabeled substrate. Since the ratios of labeled ligand to unlabeled ligand in the bound state, free state, and in the total flux are equal, a modified form of the previous equations can be used to model the binding or kinetic processes. For instance, suppose that total substrate, S say, consists of labeled substrate, $[Hot]$ say, and unlabeled substrate, $[Cold]$ say. Then the flux of labeled substrate will be given by

$$-\frac{d[Hot]}{dt} = \frac{V_{max_1}[Hot]}{K_{m_1} + [Hot] + [Cold]} + \frac{V_{max_2}[Hot]}{K_{m_2} + [Hot] + [Cold]} + \cdots + \frac{V_{max_n}[Hot]}{K_{m_n} + [Hot] + [Cold]}$$

So, if $[Hot]$ is kept fixed and $[Cold]$ is regarded as the independent variable, then program **mmfit** can be used to fit the resulting data, as shown for the test file `hotcold.tf1` in figure 6.6. Actually this figure was obtained by fitting the test file using program **qnfit**, which allows users to specify the concentration of fixed $[Hot]$. It also allows users to appreciate the contribution of the individual component species to the overall sum, by plotting the deconvolution, as illustrated. Graphical deconvolution (page 38) should always be done if it is necessary to decide on the activities of kinetically distinct isoenzymes or proportions of independent High/Low affinity binding sites. Note that an important difference between using **mmfit** in this mode rather than in straightforward kinetic mode is that the kinetic constants are modified in the following sense: the apparent V_{max} values estimated are actually the true values multiplied by the concentration of labeled substrate, while the apparent K_m values estimated are the true ones plus the concentration of labeled substrate. A similar analysis is possible for program **hlfit** as well as for programs **sffit** and **rffit**, except that here some further algebra is required, since the models are not linear summations of 1:1 rational functions. Note that, in isotope displacement mode, concentration at half maximum response can be used as an estimate for $IC50$, allowing for the ratio of labeled to unlabeled ligand, if required (page 113).

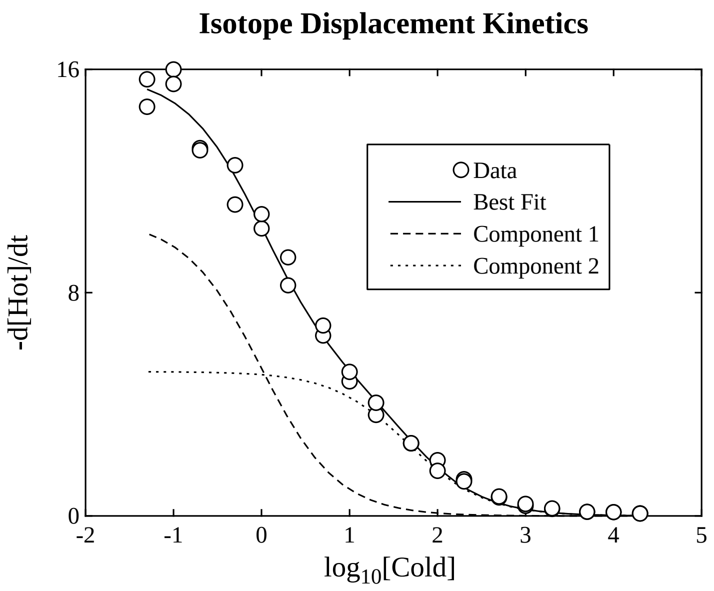


Figure 6.6: Isotope displacement kinetics

6.12 Positive rational functions

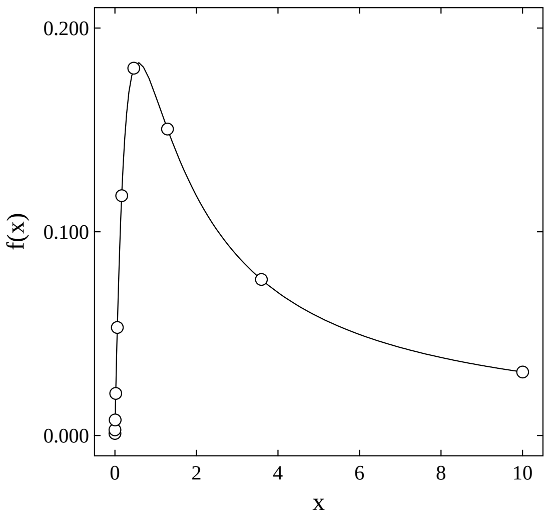


Figure 6.7: Fitting positive rational functions

Deviations from Michaelis-Menten kinetics can be fitted by the $n:n$ positive rational function

$$f(x) = \frac{\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_n x^n}{\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n}$$

where $\alpha_i \geq 0$, $\beta_0 = 1$, and $\beta_i \geq 0$. In enzyme kinetics a number of special cases arise in which some of these coefficients should be set to zero. For instance, with dead-end substrate inhibition we would have $\alpha_0 = 0$ and $\alpha_n = 0$. The test files for **rffit** are all exact data, and the idea is that you would add random error to simulate experiments. For the time being we will just fit one test file, `rffit.tf2`, with substrate inhibition data. Input `rffit.tf2` into program **rffit**, then request lowest degree $n = 2$ and highest degree $n = 2$ with $\alpha_0 = 0$ and $\alpha_2 = 0$. Note that these are called A(0) and A(N) by the program. You will get the fit shown in figure 6.7. Now you could try what happens if you fit all the test files with unrestricted rational functions of orders 1:1, 2:2 and 3:3. Also, you could pick any of these and see what happens if random error is added. Observe that program **rffit** does all manner of complicated operations to find starting estimates and scale your data, but be warned; fitting positive rational functions is extremely difficult and demands specialized knowledge. Don't be surprised if program **rffit** finds a good fit with coefficients that bear no resemblance to the actual ones.

6.13 Plotting positive rational functions

In the days before computers were used for data analysis, deviations from Michaelis-Menten kinetics, such as sigmoidicity, substrate inhibition, and substrate activation, were diagnosed by graphical techniques based upon plotting and extrapolating in transformed space, so a few remarks about the limitations of such techniques are appropriate. Of course such graphical techniques should only be used for illustration nowadays and decisions about deviations from Michaelis-Menten kinetics should only be based on the sort of statistical procedures provided by SIMFIT.

6.13.1 Scatchard plots

Kinetic measurements have zero rate at zero substrate concentration ($\alpha_0 = 0$), but this is not always possible with ligand binding experiments. Displacement assays, for instance, always require a baseline to be estimated which can have serious consequences as figure 6.8 illustrates. If a baseline is substantial, then the correct

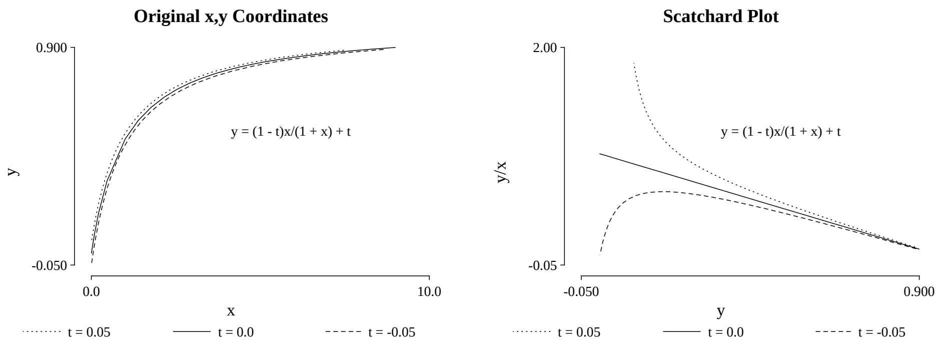


Figure 6.8: Original plot and Scatchard transform

procedure is to estimate it independently, and subtract it from all values, in order to fit using $C = 0$. Alternatively, **hlfit** can be used to estimate C in order to normalize to zero baseline. Figure 6.8 is intended to serve as a warning as to possible misunderstanding that can arise with a small baseline correction that is overlooked or not corrected properly. It shows plots of

$$y = \frac{(1-t)x}{1+x} + t$$

for the cases with $t = 0.05$ (positive baseline), $t = 0$ (no baseline) and $t = -0.05$ (negative baseline). The plots cannot be distinguished in the original space, but the differences near the origin are exaggerated in Scatchard space, giving the false impression that two binding sites are present. Decisions as to whether one or more binding sites are present should be based on the statistics calculated by SIMFIT programs, not on the plot shapes in transformed axes.

6.13.2 Semi-log plots

Whereas the Scatchard, double-reciprocal, and related plots seriously distort and over-emphasize deviations from Michaelis-Menten kinetics, the semi-logarithmic plot merely stretches the horizontal axis, making it easier to appreciate the goodness of fit. It is especially useful as logarithmic spacing spacing of the independent variable is often used to optimize tests for model discrimination. For instance, figure 6.9 illustrates a substrate inhibition curve and the semilogarithmic transform, which is the best way to view such fits.

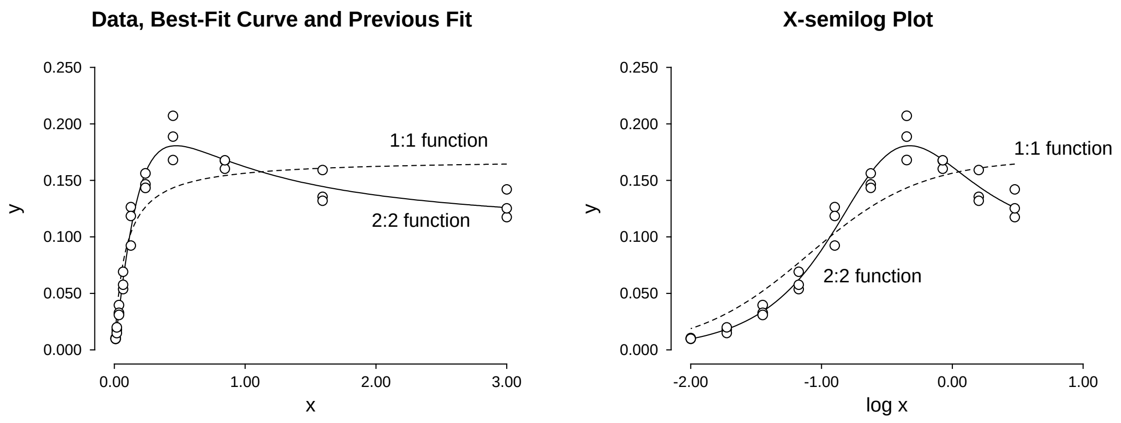


Figure 6.9: Substrate inhibition plot and semilog transform

6.13.3 Asymptotic forms

The 3:3 rational function with $\alpha_0 = 0$ has 27 possible double reciprocal plots, and it is seldom possible to justify degrees for $n > 3$ anyway by statistical tests, although there may be compulsive chemical evidence, e.g. hemoglobin with $n = 4$. Now the coefficients α_i and β_j will be functions of rate or binding constants, so limiting expressions are often sought to obtain information about kinetic mechanisms by fitting simpler equations, e.g. a 1:1 function with $\alpha_0 = 0$ to the whole data set or just the extreme values. The interpretation of the parameters of a best fit single Michaelis-Menten 1:1 hyperbola fitted in this way may be aided by considering three special formulations.

$$\begin{aligned} TPH &= \frac{\alpha_1\alpha_n x}{\alpha_n\beta_0 + \alpha_1\beta_n x}, \\ OH(0) &= \frac{\alpha_1^2 x}{\alpha_1\beta_0 + (\alpha_1\beta_1 - \alpha_2\beta_0)x}, \\ OH(\infty) &= \frac{\alpha_n^2 x}{(\alpha_n\beta_{n-1} - \alpha_{n-1}\beta_n) + \alpha_n\beta_n x}. \end{aligned}$$

The theoretical parent hyperbola TPH is what would be required if the high degree model is almost indistinguishable from a 1:1 function because all the Sylvester dialytic eliminants are effectively zero, indicating a reduction in degree by cancelation of common factors. To the extent that this procedure is justified, the two parameters that can be estimated are the apparent kinetic constants

$$V_{max} = \alpha_n/\beta_n, \text{ and } K_m = \alpha_n\beta_0/(\alpha_1\beta_1).$$

The osculating hyperbola at zero $OH(0)$ is the best 1:1 approximant to a rational function at the origin, as it has second order contact and can model a sigmoid curve. Irrespective of which technique is used to fit the data for low x the only parameters that can be estimated are the apparent kinetic constants

$$V_{max} = \alpha_1^2/(\alpha_1\beta_1 - \alpha_2\beta_0), \text{ and } K_m = \alpha_1\beta_0/(\alpha_1\beta_1 - \alpha_2\beta_0).$$

The osculating hyperbola at infinity $OH(\infty)$ is the best 1:1 approximant to a rational function as the independent variable increases without limit, as it has second order contact and can model substrate inhibition curves. Irrespective of which technique is used to fit the data for high x the only parameters that can be estimated are the apparent kinetic constants

$$V_{max} = \alpha_n/\beta_n, \text{ and } K_m = (\alpha_n\beta_{n-1} - \alpha_{n-1}\beta_n)/(\alpha_n\beta_n).$$

6.13.4 Sigmoidicity

A positive rational function will be sigmoid if

$$\alpha_2\beta_0^2 - \alpha_1\beta_0\beta_1 - \alpha_0\beta_0\beta_2 + \alpha_0\beta_1^2 > 0,$$

but in the usual case $\alpha_0 = 0$ it is possible to define satisfactory measures of sigmoidicity, which can be explained by reference to figure 6.10.

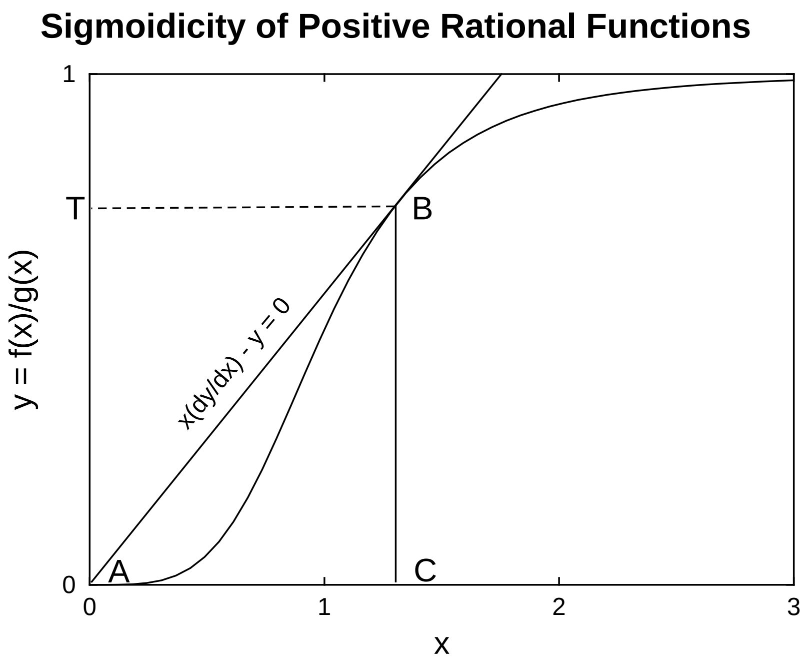


Figure 6.10: Definition of sigmoidicity

The point labeled C is the first positive root of

$$xy' - y = 0$$

and the point labeled $T = y(C)$ is the y coordinate where the tangent from the origin touches the curve. Consider then the expressions

$$\Delta_1 = \frac{T}{\max(y), \text{ for } x \geq 0}$$
$$\Delta_2 = \frac{Area(ABC)}{\int_0^C y(t)dt}$$

where Δ_1 and Δ_2 both increase as sigmoidicity increases. It can be shown that, for fractional saturation functions of order n, the following inequality applies

$$T \leq \frac{n-1}{n},$$

while, at least for the cases $n = 2, 3, 4$, the positive rational function curve of maximum sigmoidicity is the normalized Hill equation

$$y = \frac{x^n}{1 + x^n}.$$

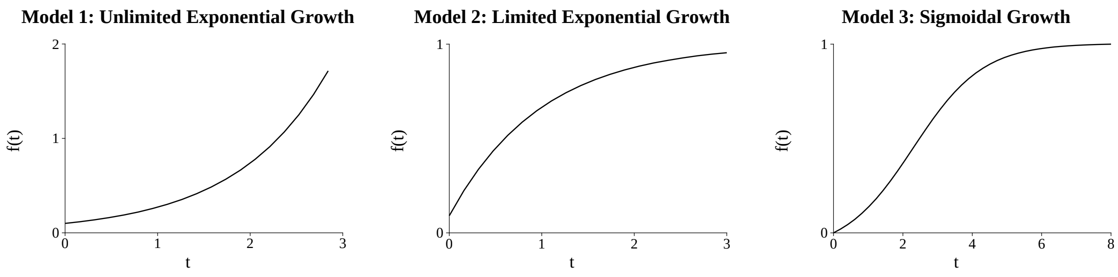


Figure 6.11: Typical growth curve models

6.14 Nonlinear growth curves

Three typical growth curve shapes are shown in figure 6.11. Model 1 is exponential growth, which is only encountered in the early phase of development, Model 2 is limited exponential growth, concave down to an asymptote fitted by the monomolecular model (Type 3 of figure 6.1), and several models can fit sigmoidal profiles as for Model 3 in figure 6.11, e.g., the logistic equation

$$f(t) = \frac{A}{1 + B\exp(-kt)}.$$

Select [Fit] from the main menu then **gcfit** to fit growth curves. Input gcfit.tf2 then fit models 1, 2 and 3 in sequence to obtain figure 6.12, i.e., the exponential model gives a very poor fit, the monomolecular model leads to an improved fit, but the logistic is much better. This is the usual sequence of fitting with **gcfit** but it does much more. It can fit up to ten models sequentially and gives many statistics, such as maximum growth rate, to assist advanced users. The reason why there are alternative models, such as those of Gompertz, Richards, Von Bertalanffy, Preece and Baines, etc., is that the logistic model is often too restrictive, being symmetrical about the mid point, so generating biased fit. The table of compared fits displayed by **gcfit** helps in model selection, however, none of these models can accommodate turning points, and all benefit from sufficient data to define the position of the horizontal asymptote.

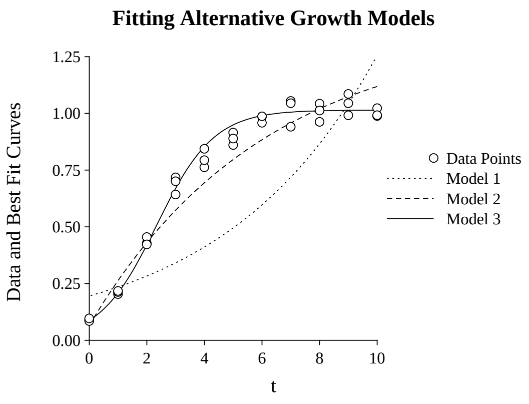


Figure 6.12: Using **gcfit** to fit growth curves

Most growth data are monotonically increasing observations of size $S(t)$ as a function of time t, from a small value of size S_0 at time $t = 0$ to a final asymptote S_∞ at large time. Decay data are fitted like growth models but in reversed order (see page 77). The usual reason for fitting models is to compare growth rates between different populations, or to estimate parameters, e.g., the maximum growth rate, maximum size, time to reach half maximum size, etc. The models used are mostly variants of the Von Bertalanffy allometric differential equation

$$dS/dt = AS^\alpha - BS^\beta,$$

which supposes that growth rate is the difference between anabolism and catabolism expressed as power functions in size. This equation defines monotonically increasing $S(t)$ profiles and can be fitted by **deqsol** or **qnfit**, but a number of special cases leading to explicit integrals are frequently encountered. These have the benefit that parameters estimated from the data have a physical meaning, unlike fitting polynomials where the parameters have no meaning and cannot be used to estimate final size, maximum growth rate and so on. Clearly, the following models should only be fitted when data cover a sufficient time range to allow meaningful estimates for S_0 and S_∞.

1. Exponential model $dS/dt = kS$
   $$S(t) = A\exp(kt),\ \text{where } A = S_0$$
2. Monomolecular model $dS/dt = k(A - S)$
   $$S(t) = A[1 - B\exp(-kt)],\ \text{where } B = 1 - S_0/A$$
3. Logistic model $dS/dt = kS(A - S)/A$
   $$S(t) = A/[1 + B\exp(-kt)],\ \text{where B } = A/S_0 - 1$$
4. Gompertz model $dS/dt = kS[\log(A) - \log(S)]$
   $$S(t) = A\exp[-B\exp(-kt)],\ \text{where } B = log(A/S_0)$$
5. Von Bertalanffy 2/3 model $dS/dt = \eta S^{2/3} - \kappa S$
   $$S(t) = [A^{1/3} - B\exp(-kt)]^3$$
   $$\text{where } A^{1/3} = \eta/\kappa, B = \eta/\kappa - S_0^{1/3}, k = \kappa/3$$
6. Model 3 with constant $f(t) = S(t) - C$
   $$df/dt = dS/dt = kf(t)(A - f(t))/A$$
   $$S(t) = A/[1 + B\exp(-kt)] + C$$
7. Model 4 with constant $f(t) = S(t) - C$
   $$df/dt = dS/dt = kf(t)[\log(A) - \log(f(t))]$$
   $$S(t) = A\exp[-B\exp(-kt)] + C$$
8. Model 5 with constant $f(t) = S(t) - C$
   $$df/dt = dS/dt = \eta f(t)^{2/3} - \kappa f(t)$$
   $$S(t) = [A^{1/3} - B\exp(-kt)]^3 + C$$
9. Richards model $dS/dt = \eta S^m - \kappa S$
   $$S(t) = [A^{1-m} - B\exp(-kt)]^{[1/(1-m)]}$$
   $$\text{where } A^{1-m} = \eta/\kappa, B = \eta/\kappa - S_0^{1-m}, k = \kappa(1-m)$$
   $$\text{if } m < 1 \text{ then } \eta, \kappa, A \text{ and } B \text{ are } > 0$$
   $$\text{if } m > 1 \text{ then } A > 0 \text{ but } \eta, \kappa \text{ and } B \text{ are } < 0$$
10. Preece and Baines model $f(t) = \exp[k_0(t - \theta)] + \exp[k_1(t - \theta)]$
    $$S(t) = h_1 - 2(h_1 - h_\theta)/f(t)$$

In mode 1, **gcfit** fits a selection of these classical growth models, estimates the maximum size, maximum and minimum growth rates, and times to half maximum response, then compares the fits. As an example, consider table 6.5, which is an abbreviated form of the results file from fitting `gcfit.tf2`, as described by figure 6.12. This establishes the satisfactory fit with the logistic model when compared to the exponential and monomolecular models.

Figure 6.13 shows typical derived plots as follows.

- Data with and best-fit curve $\hat{S}$ and asymptote $\hat{S}_\infty$;
- Derivative of best-fit curve $d\hat{S}/dt$; and
- Relative rate $(1/\hat{S})d\hat{S}/dt$.

```
 Results for model  1
 Parameter  Value      Std.error    Lower95%cl  Upper95%cl    p
   A        0.196289   0.0274736    0.140256    0.252321  0.0000
   k        0.184039   0.0184007    0.146511    0.221568  0.0000

 Results for model  2
 Parameter    Value      Std.error  Lower95%cl  Upper95%cl    p
   A         1.32842    0.115824     1.09188     1.56497  0.0000
   B         0.94899  0.00951585    0.929557    0.968424  0.0000
   k        0.170044   0.0289602    0.110899    0.229188  0.0000
  t-half     3.76839    0.642108     2.45703     5.07975  0.0000

 Results for model  3

 Parameter  Value      Std.error  Lower95%cl  Upper95%cl    p
   A        0.998913  0.00785511    0.982871     1.01496  0.0000
   B         9.89009    0.333001     9.21001     10.5702  0.0000
   k        0.988137   0.0267846    0.933436     1.04284  0.0000
  t-half     2.31904   0.0450705       2.227     2.41109  0.0000
 Residuals and Goodness of Fit
 Max. observed data value   = 1.0856  Theoretical asymptote   = 0.9989
 Max. observed/Th.asymptote = 1.087
 Max. observed growth rate  = 0.24    Best fit curve maximum  = 0.2468
 Time max. rate observed    = 2.0     Best fit curve time     = 2.3189
 Min. observed growth rate  = 0.0005  Best fit curve minimum  = 0.0005
 Time min. rate observed    = 10.0    Best fit curve time     = 10.0

Summary
Model   WSSQ DOF WSSQ/DOF P(C>=W) P(R=<r) NR>10% NR>40%  Av.r% Verdict

 1  4718.41 31  152.207  0.0000  0.0000    29    17    40.03 Very  bad
 2  541.932 30  18.0644  0.0000  0.0748    20     0    12.05 Very poor
 3  39.5957 30  1.31986  0.1129  0.5000     0     0     3.83 Incredible
```

Table 6.5: Fitting nonlinear growth models

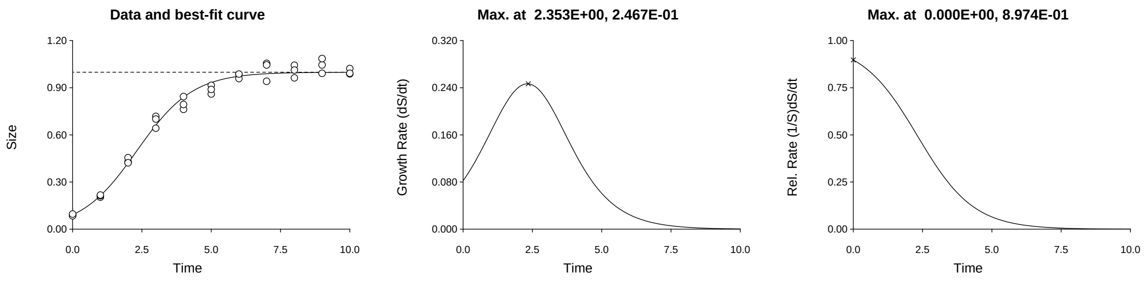


Figure 6.13: Estimating growth curve parameters

6.15 Nonlinear survival curves

In mode 2, **gcfit** fits a sequence of survival curves, where it is assumed that the data are uncorrelated estimates of fractions surviving $0 \le S(t) \le 1$ as a function of time $t \ge 0$, e.g. such as would result from using independent samples for each time point. However, as normalizing data to $S(0) = 1$ can introduce bias, mode 2 allows an amplitude factor to be estimated (see page 77).

It is important to realize that, if any censoring has taken place, the estimated fraction should be corrected for this. In other words, you start with a population of known size and, as time elapses, you estimate the fraction surviving by any sampling technique that gives estimates corrected to the original population at time zero.

The test files `weibull.tf1` and `gompertz.tf1` contain some exact data, which you can fit to see how mode 2 works. Then you can add error to simulate reality using program **adderr**. Note that you prepare your own data files for mode 2 using the same format as for program **makfil**, making sure that the fractions are between zero and one, and that only nonnegative times are allowed. It is probably best to do unweighted regression with this sort of data (i.e. all $s = 1$) unless the variance of the sampling technique has been investigated independently.

In survival mode the time to half maximum response is estimated with 95% confidence limits and this can used to estimate $LD50$ (page 113). The survivor function is $S(t) = 1 - F(t)$, the *pdf* is $f(t)$, i.e. $f(t) = -dS/dt$, the hazard function is $h(t) = f(t)/S(t)$, and the cumulative hazard is $H(t) = -\log(S(t))$. Plots are provided for $S(t), f(t), h(t), \log[h(t)]$ and, as in mode 1, a summary is given to help choose the best fit model from the following list, all of which decrease monotonically from $S(0) = 1$ to $S(\infty) = 0$ with increasing time.

$$\begin{aligned}
\text{1. Exponential model } S(t) &= \exp(-At)\\
f(t) &= AS(t)\\
h(t) &= A\\
\text{2. Weibull model } S(t) &= \exp[-(At)^B]\\
f(t) &= AB[(At)^{B-1}]S(t)\\
h(t) &= AB(At)^{B-1}\\
\text{3. Gompertz model } S(t) &= \exp[-(B/A)\{\exp(At) - 1\}]\\
f(t) &= B\exp(At)S(t)\\
h(t) &= B\exp(At)\\
\text{4. Log-logistic model } S(t) &= 1/[1 + (At)^B]\\
f(t) &= AB(At)^{B-1}/[1 + (At)^B]^2\\
h(t) &= AB(At)^{B-1}/[1 + (At)^B]
\end{aligned}$$

Note that, in modes 3 and 4, **gcfit** provides options for using such survival models to analyze survival times, as described on page 253.

6.16 Nonlinear decay curves

It is often required to model data that are decaying monotonically from a positive starting size at zero time to a final value of zero or a limiting background level. The most usual case is where it is intended to use a simple growth or survival model in order to estimate parameters such as

- The starting size;
- The time to half decay;
- The maximum rate of decay; or sometimes

- The final asymptotic size.

Program **gcfit** can be used in modes 1 or 2 for this purpose with certain limitations.

When mode 1 is used and a data set is provided where $\alpha \le t \le \beta$ and $S(\alpha) > S(\beta)$ then **gcfit** reverses the data to generate a new variable T from the original variable t using

$$T = \alpha + \beta - t$$

then rearranging the data into increasing order of T. Program **gcfit** then fits the reversed data set using the normal growth curve models and outputs best-fit parameters referring to the reversed data, except that outputs involving time, such as the half time or plots, refer to the original time. Of course the estimated maximum asymptotic size, that is $\hat{A}$ or $\hat{A} + \hat{C}$, will then refer to the starting size at time zero, and $\hat{C}$ will be the estimated final size.

When mode 2 is used there will be no such issues as the data will be in the correct order. Accordingly it will usually be best to fit decay data using one of the survival models as long as it realized that it will be necessary to allow **gcfit** to estimate the starting amplitude $\hat{S}(0)$ and also to note that these models are asymptotic to zero size at infinite time.

6.17 Accuracy of growth/decay/survival parameter estimates

Program **gcfit** first normalizes data so that the parameters to be estimated are of order unity, then it calculates starting estimates using the first few and last few points. After unconstrained regression using explicit formulas for the models, derivatives, and Jacobians, the parameter estimates are re-scaled into units that refer to the original data. If the model is a good approximation and the data are extensive and of high quality some of the parameters, such as the maximum size, can be estimated with reasonable precision. However, the half-time is often difficult to estimate accurately, and the minimax slopes can usually not be estimated very accurately. This problem is seriously compounded by using a model with insufficient flexibility leading to a biased fit. Each case must be decided upon after experimentation but, in general, the Gompertz model model would usually be preferred in mode 1, and the Weibull or Gompertz model in mode 2.

Note that **gcfit** calculates the half-times by substituting the m parameter estimates $\theta(i), i = 1, 2, \ldots, m$ into the formula

$$t_{1/2} = f(\Theta)$$

required for the model chosen, but then the confidence limits are calculated using the estimated covariance matrix and the propagation of errors formula

$$\hat{V}(t_{1/2}) = \sum_{i=1}^{m} \left(\frac{\partial f}{\partial \theta_i}\right)^2 \hat{V}(\theta_i) + 2 \sum_{i=2}^{m} \sum_{j=1}^{i-1} \frac{\partial f}{\partial \theta_i} \frac{\partial f}{\partial \theta_j} \hat{CV}(\theta_i, \theta_j).$$

As the estimated covariance matrix is already based upon a linearizing assumption and the above variance expression only uses the early terms of a Taylor series expansion, these confidence limits should be interpreted with restraint.

Part 7

Nonlinear models: Advanced fitting

Eventually there always comes a time when users want extra features, like the following.

a) Interactive choice of model, data sub-set or weighting scheme.
b) Choice of optimization technique.
c) Fixing some parameters while others vary in windows of restricted parameter space.
d) Supplying user-defined models with features such as special functions, functions of several variables, root finding, numerical integration, Chebyshev expansions, etc.
e) Simultaneously fitting a set of equations, possibly linked by common parameters.
f) Fitting models defined parametrically, or as functions of functions.
g) Estimating the eigenvalues and condition number of the Hessian matrix at solution points.
h) Visualizing the weighted sum of squares and its contours at solution points.
i) Inverse prediction, i.e., nonlinear calibration.
j) Estimating first and second derivatives or areas under best fit curves.
k) Supplying starting estimates added to the end of data files.
l) Selecting sets of starting estimates from parameter limits files.
m) Performing random searches of parameter space before commencing fitting.
n) Estimating weights from the data and best fit model.
o) Saving parameters for excess variance F tests in model discrimination.

Program **qnfit** is provided for such advanced curve fitting. The basic version of program **qnfit** only supports quasi-Newton optimization, but some versions allow the user to select modified Gauss-Newton or sequential quadratic programming techniques. Users must be warned that this is a very advanced piece of software and it demands a lot from users. In particular, it scales parameters but doesn't scale data. To ensure optimum operation, users should appreciate how to scale data correctly, especially with models where parameters occur exponentially. They should also understand the mathematics of the model being fitted and have good starting estimates. In expert mode, starting estimates and limits are appended to data files to facilitate exploring parameter space. Test files `qnfit.tf1` (1 variable), `qnfit.tf2` (2 variables) and `qnfit.tf3` (3 variables) have such parameter windows. Alternatively, parameter limits files can be supplied, preferably as library files like `qnfit.tfl.` Several cases will now be described as examples of advanced curve fitting .

7.1 Fitting a function of one variable using qnfit

If we read the test file `gauss3.tf1` into **qnfit** and select to fit a sum of three Gaussian probability density functions with variable amplitudes (page 467) then the initial fit to the data using the parameter limits appended to the data file, i.e. the overlay before fitting, can be viewed as in the left hand sub-figure of figure 7.1. After curve fitting has been completed the right hand sub-figure displays the final best-fit curve, and the details from the fitting are in table 7.1. Note that the constant term was held fixed at zero, as was required by the limits appended to the test file. After successful fitting, users may wish to report estimated parameters with central confidence limits, which is easily done as the appropriate t values for the correct degrees of freedom

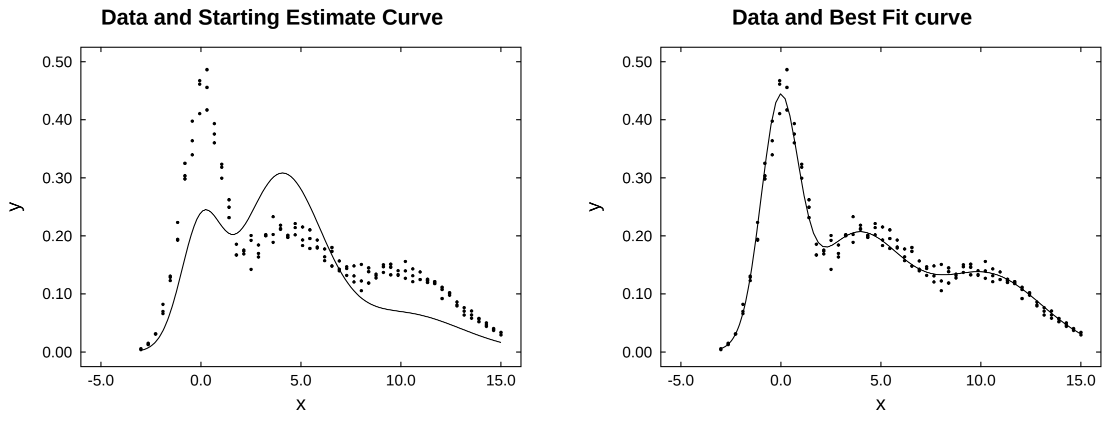


Figure 7.1: Fitting a sum of three Gaussians

are always appended to the end of the **qnfit** best-fit parameters table.

After a fit has been completed there are numerous further options provided for studying the fit or plotting the results in alternative ways. For instance, figure 7.2 shows in the left hand figure the way that models consisting

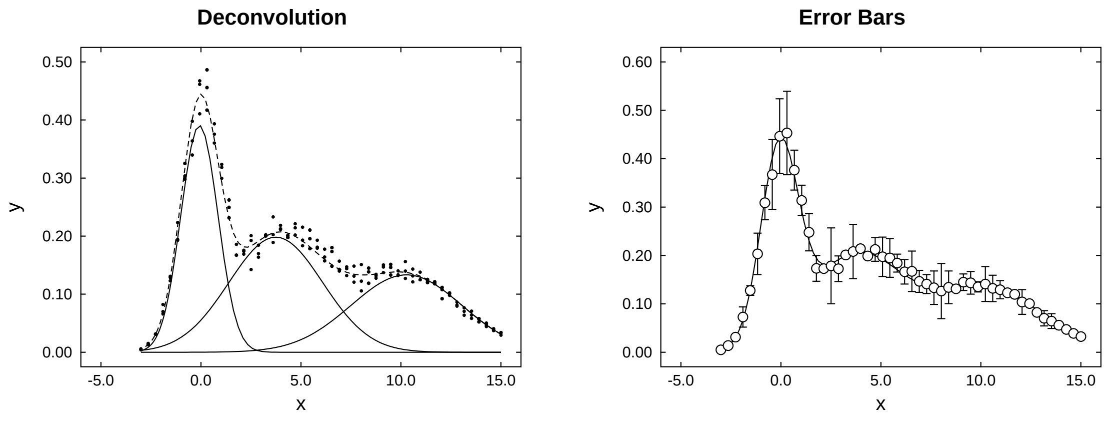


Figure 7.2: Further plots after fitting a sum of three Gaussians

of sums of independent sub-models can be plotted along with then sub-models to indicate the contribution of the separate components to the fit, a sort of graphical deconvolution, while the right hand sub-figure illustrates how error bars can be calculated interactively.

```
Number Low-Limit High-Limit   Value Std.Error Lower95%cl Upper95%cl   p
     1     0.0       2.0   0.907544 0.0216242   0.864794   0.950293 0.0000
     2     0.0       2.0    1.16432  0.042174    1.08095     1.2477 0.0000
     3     0.0       2.0   0.925188 0.0301306   0.865622   0.984755 0.0000
     4    -2.0       2.0 -0.0729757 0.0155719   -0.10376 -0.0421911 0.0000
     5     2.0       6.0     3.7451 0.0508154    3.64464    3.84556 0.0000
     6     8.0      12.0    10.2774 0.0964136    10.0868     10.468 0.0000
     7     0.1       2.0   0.926406 0.0143311   0.898074   0.954738 0.0000
     8     0.1       3.0    2.34329 0.0705682    2.20378     2.4828 0.0000
     9     0.1       4.0    2.76907 0.0626377    2.64524     2.8929 0.0000
     parameter(10) is the excluded constant term
 For 50,90,95,99% con. lim. use [parameter value +/- t(alpha/2)*std.err.]
 t(.25) = 0.676, t(.05) = 1.656, t(.025) = 1.977, t(.005) = 2.611

Goodness of fit

Analysis of residuals:   WSSQ                  = 271.113
p = P(chi-sq. >= WSSQ)                         = 0.0000 Reject at 1% level
R-squared [corr.coeff.(best-fit,observed)]^2 = 0.9773
Largest  absolute relative residual            =     32.21%
Smallest absolute relative residual            =      0.13%
Average  absolute relative residual            =      6.32%
Absolute relative residuals in range 0.1-0.2 =     20.00%
Absolute relative residuals in range 0.2-0.4 =      2.67%
Absolute relative residuals in range 0.4-0.8 =      0.00%
Absolute relative residuals > 0.8              =      0.00%
Number of negative residuals (m)               = 75
Number of positive residuals (n)               = 75
Number of runs observed (r)                    = 69
p = P(runs =< r : given m and n)               = 0.1434
5% lower tail point                            = 65
1% lower tail point                            = 61
p = P(runs =< r : given m plus n)              = 0.1628
p = P(signs =< least number observed)          = 1.0000
Durbin-Watson test statistic                   = 1.6926
Shapiro-Wilks W statistic                      = 0.9897
p = Significance level of W                    = 0.3417
Akaike AIC (Schwarz SC) statistics             = 106.785 (133.881)
Verdict on goodness of fit                     = excellent
```

Table 7.1: Parameters for best-fit Gaussians

7.2 Fitting a mixture of two normal distributions

Often samples consist of a mixture of distributions. For instance a sample of heights of subjects drawn from a homogeneous population, i.e. of the same age and medical condition, could appear to be be approximately normally distributed but would actually consist of two sub-populations, male and female. In reality, special techniques exist for analyzing certain cases where populations cannot be physically separated into sub-groups but can be resolved into supposed sub-populations using the method of maximum likelihood. However, the curve fitting approach will be discussed in this tutorial because, in principle, it can be used for arbitrary mixtures of any any distributions, not just normal distributions.

For instance, to explain how to use SimFIT program **qnfit** for this purpose, consider the simplest case of a sample arising from a mixture of two normally distributed sub populations, so that a sample partitioned into histogram bins could be approximately modeled by the expression

$$f(x) = \frac{t}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left\{\frac{x-\mu_1}{\sigma_1}\right\}^2\right) + \frac{1-t}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left\{\frac{x-\mu_2}{\sigma_2}\right\}^2\right)$$

where $0 \le t \le 1$ and the parameters $t, \mu_1, \mu_2, \sigma_1, \sigma_2$ must be estimated by fitting to the histogram bins. The serious problem with this approach is that the shape of the histogram, and therefore the best-fit parameters, will depend on the number of bins chosen. It should therefore be obvious that a very large sample will be necessary, and meaningful parameter estimates can only be expected when μ_1 and μ_2 are widely separate, σ_1 and σ_2 are similar and less than the difference between μ_1 and μ_2, and the partitioning parameter t must obey $t \approx 0.5$. Of course the constraints $\sigma_1 > 0, \sigma_2 > 0$ also must be imposed.

7.2.1 Fitting histogram data

The data file `qnfit_data.tf6` can be selected from **qnfit** by clicking on the [Demo] button, and it is listed below after extracting as a table using the [Results] button on the main SimFIT menu.

```
Data file qnfit_data.tf4
 10        3
-3.6   0.0375    1.0
-2.8   0.0625    1.0
-2.0   0.2000    1.0
-1.2   0.2000    1.0
-0.4   0.1000    1.0
 0.4   0.1250    1.0
 1.2   0.1250    1.0
 2.0   0.2500    1.0
 2.8   0.1250    1.0
 3.6   0.0250    1.0
begin{limits}
-5.0     -1.0    0.0
 0.1      0.8    5.0
 0.1      0.4    0.9
 0.0      1.0    5.0
 0.1      1.2    5.0
end{limits}
```

This file was created by reading a mixed sample of 50 $N(-1.5, 1)$ numbers and 50 $N(1.5, 1)$ numbers from program **rannum** into the exhaustive analysis of a vector routine available under [Data exploration] from the [Statistics] option on the main SimFIT menu. This indicates that the mixed sample is not consistent with a single normal distribution and this step should be taken before fitting any data set because, if the sample is consistent with a single normal distribution, there is little point in trying to fit a sum of two non–identical distributions. This procedure also gives the option of plotting a histogram and then, having chosen the number of bins required, it can create a curve fitting file either unweighted or weighted by the square root of the bin size. Unless a very large sample is under investigation and there are no empty bins an unweighted file should be created. Note in particular that, as the best-fit curve integrates to unity over the data range $(-\infty, \infty)$, the option to normalize the histogram to area 1 must also be chosen.

After reading in the data file `qnfit_data.tf6` the model file `qnfit_model.tf6` should be selected, and this contains the following definition for a sum of two normal distributions.

```
%
Sum of two normal pdfs
A = -(1/2)[(x - p(1))/p(2)]^2
B = -(1/2)[(x - p(4))/p(5)]^2
f(x) = {[1 - p(3)]exp(A)/p(2) + p(3)exp(B)/p(5)}/sqrt{2*pi)
%
1 equation
1 variable
5 parameters
%
begin{expression}
A = -0.5*[(x - p(1))/p(2)]^2
B = -0.5*[(x - p(4))/p(5)]^2
C = [1.0 - p(3)]*exp(a)/p(2) + p(3)*exp(b)/p(5)
f(1) = C/root2pi
end{expression}
%
```

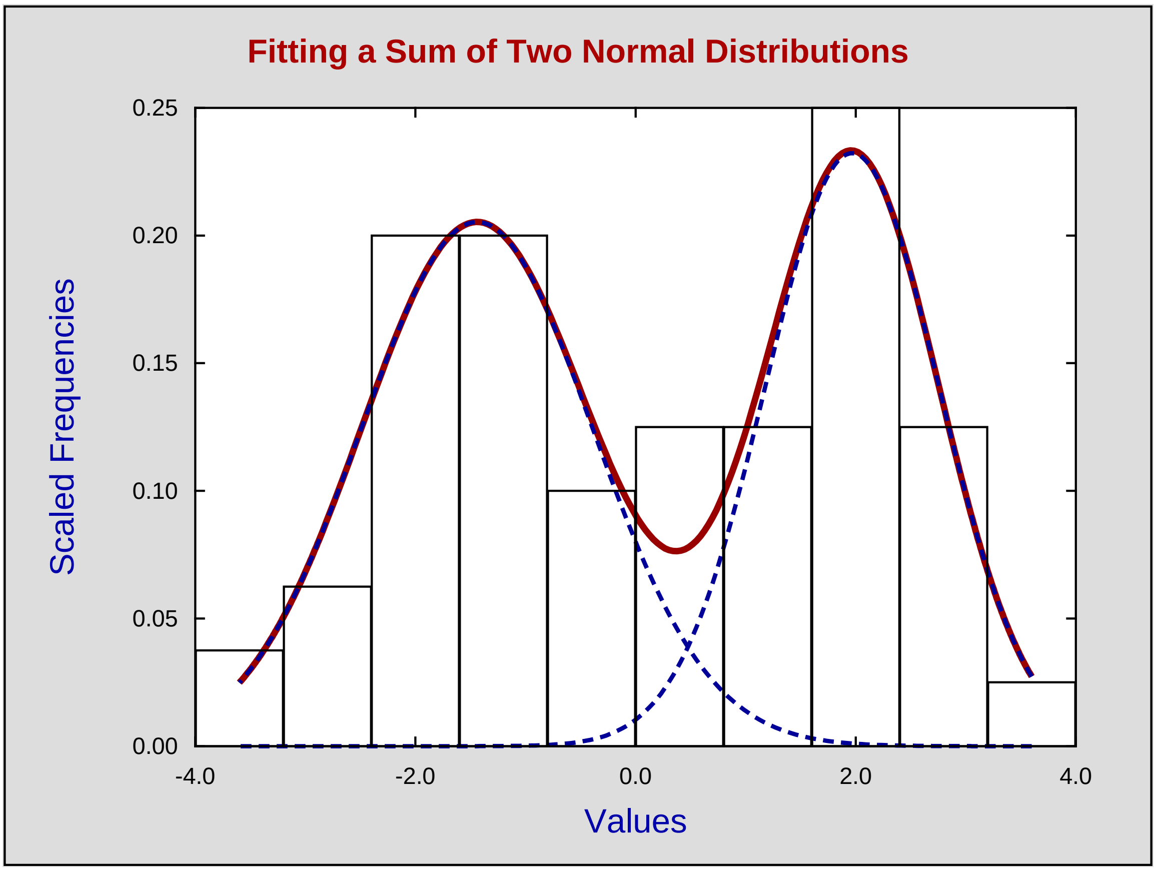
The best fit results table follows.

Number	Low-Limit	High-Limit	Value	Std.Error	Lower95%cl	Upper95%cl	p
1	-5.0	0.0	-1.44100	0.172367	-1.88408	-0.99792	0.0004
2	0.1	5.0	1.05066	0.181518	0.58405	1.51726	0.0022
3	0.1	0.9	0.45929	0.061382	0.30150	0.61707	0.0007
4	0.0	5.0	1.96743	0.133634	1.62392	2.31095	0.0000
5	0.1	5.0	0.78877	0.135524	0.44039	1.13714	0.0021

It might be required to plot the best-fit curve superimposed on the sample histogram and the following steps are required to do this.

1. Request a plot in the usual way then choose [Advanced] and transfer to advanced editing.
2. The plot displayed will have symbols for the mid–points of the histogram which need to be changed.
3. From the [Data] options choose to plot bars instead of symbols.
4. The bar type, fill–style, color, and width can be altered if required.

A typical plot resulting from this editing is shown next and clearly shows that, with such dense and well–separated accurate data, a reasonable fit has been achieved. The profile of the two contributing sub–groups was obtained by using the SIMFIT library built–in equation instead of the model file and finally requesting graphical deconvolution



7.2.2 Fitting a cumulative frequency

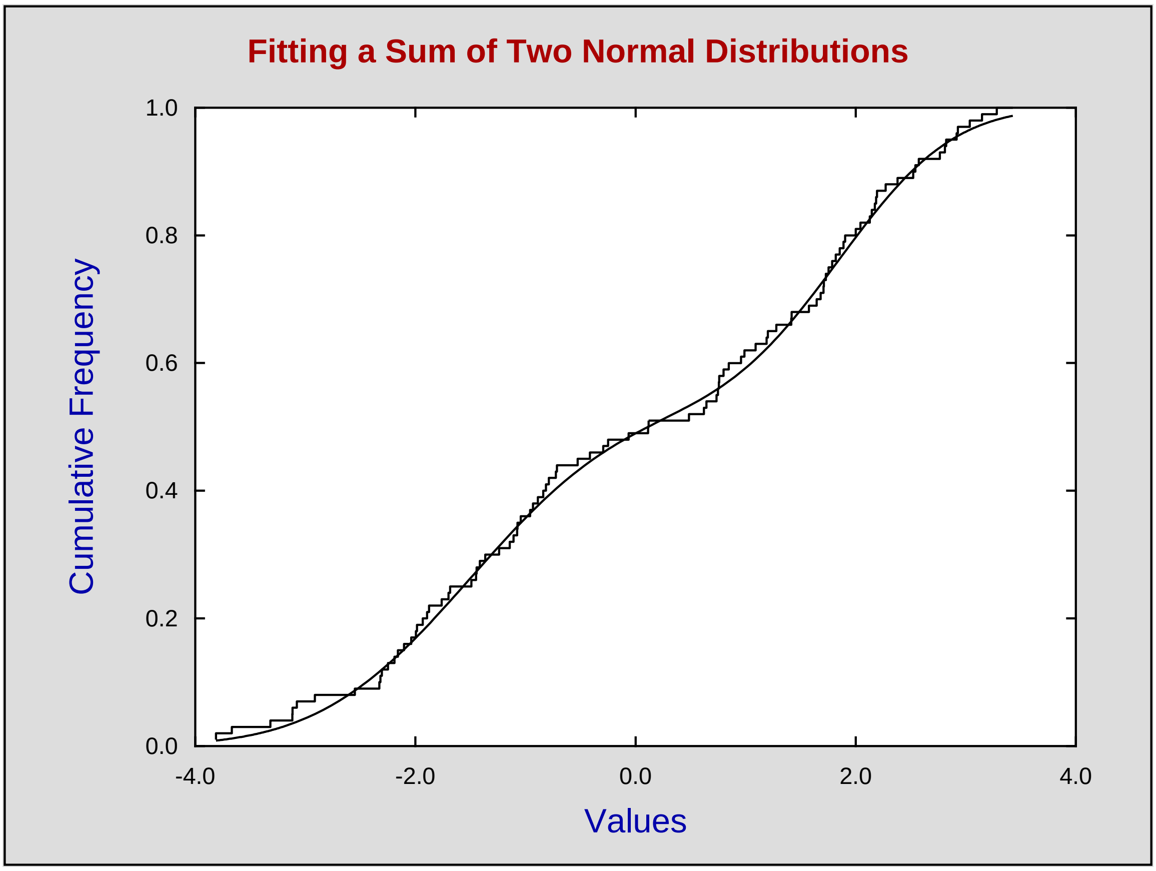
Often data are only available in partitioned form. For instance, counts from channels in flow cytometry are effectively in the form of histogram bins, so the analysis by fitting *pdfs* is all that is possible despite the fact that the results will depend on the number of bins. However, when a sample is available it is possible to fit a sum of two normal *cdfs* as discussed next, and this does not depend on partitioning into bins.

Read test file `normal.tf3` into the exhaustive analysis of a vector procedure exactly as with Example 6 but this time choose to export a *cdf* type curve fitting file. This test file is called `qnfit_data.tf7` and the model file `qnfit_model.tf7` created using SIMFIT **usermod** is as below.

```
%
Sum of two normal distributions
p(3)Phi((x - p(1))/p(2)) + (1 - p(3))Phi((x - p(4))/p(5))
%
1 equation
1 variable
5 parameters
%
begin{expression}
A = p(3)normalcdf((x - p(1))/p(2))
B = (1.0 - p(3))normalcdf((x - p(4))/p(5))
f(1) = A + B
end{expression}
%
```

The table of parameter estimates is displayed next followed by a plot of the data with best–fit curve.

Number	Low-Limit	High-Limit	Value	Std.Error	Lower95%cl	Upper95%cl	p
1	-5.0	0.0	-1.49012	0.034982	-1.55957	-1.42067	0.0000
2	0.1	5.0	1.08482	0.036551	1.01226	1.15738	0.0000
3	0.1	0.9	0.52956	0.010863	0.50799	0.55112	0.0000
4	0.0	5.0	1.85840	0.028793	1.80124	1.91556	0.0000
5	0.1	5.0	0.81238	0.030455	0.75192	0.87284	0.0000



To compare the results from fitting the *pdfs* and *cdfs* we can define the sums of squares SSQ between the parameter estimates $\hat{p}_i$ and the population parameters p_i as

$$SSQ = \sum_{i=1}^{5}(\hat{p}_i - p_i)^2$$

and note that

for the *pdfs*: $SSQ = 0.271$, and $\sqrt{SSQ} = 0.520$, while
for the *cdfs*: $SSQ = 0.171$, and $\sqrt{SSQ} = 0.414$

a slightly better result from fitting the *cdfs*.

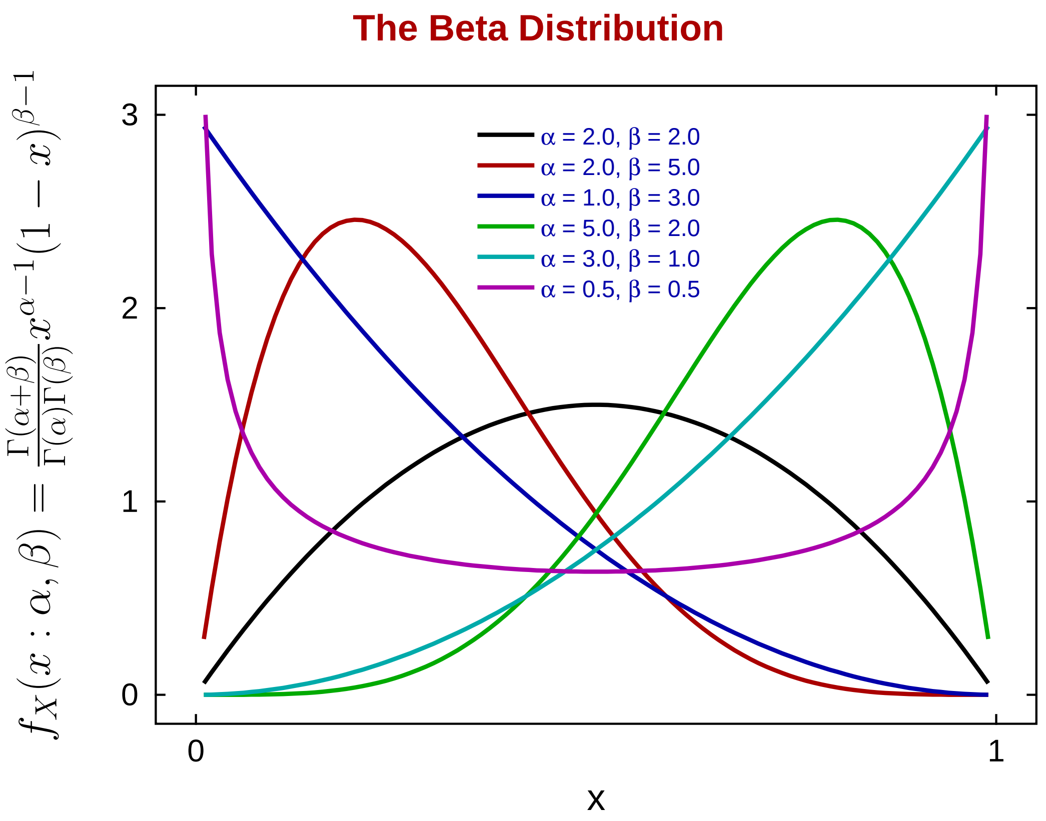
In order to succeed in estimating convincing parameter estimates there must be a very large sample with well–separated means, similar variances that do not cause too much overlap, and approximately equally sized sub–groups. Then fitting a *cdf* will give a unique set of parameter estimates as opposed to the way that fitting *pdfs* is dependent on the number of bins, but a visual display of the contribution by sub–groups is perhaps easier judged by superimposing a best fit curve on a histogram.

7.3 Fitting a beta distribution to a sample of observations

A random variable X $(0 \le x \le 1)$ with the following pdf $f_X(x : \alpha, \beta)$ and cdf $F_X(x : \alpha, \beta)$

$$
\begin{aligned}
f_X(x : \alpha, \beta) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \\
F_X(x : \alpha, \beta) &= \int_0^x f_X(t : \alpha, \beta)\, dt \\
&= I_x(\alpha, \beta)
\end{aligned}
$$

with parameters $\alpha > 0$ and $\beta > 0$, where $I_x(\alpha, \beta)$ is the regularized incomplete beta distribution, is referred to as a beta random variable. The widespread use of this distribution in data analysis arises not because many experimental observations do actually arise from a beta distribution, but because it is often a convenient unimodal distribution that serves well as an approximation in many situations, such as those involving the estimation of proportions. Some idea of the variation in the profile of a beta distribution as a function of the shape parameters α and β will be clear for the next figure.

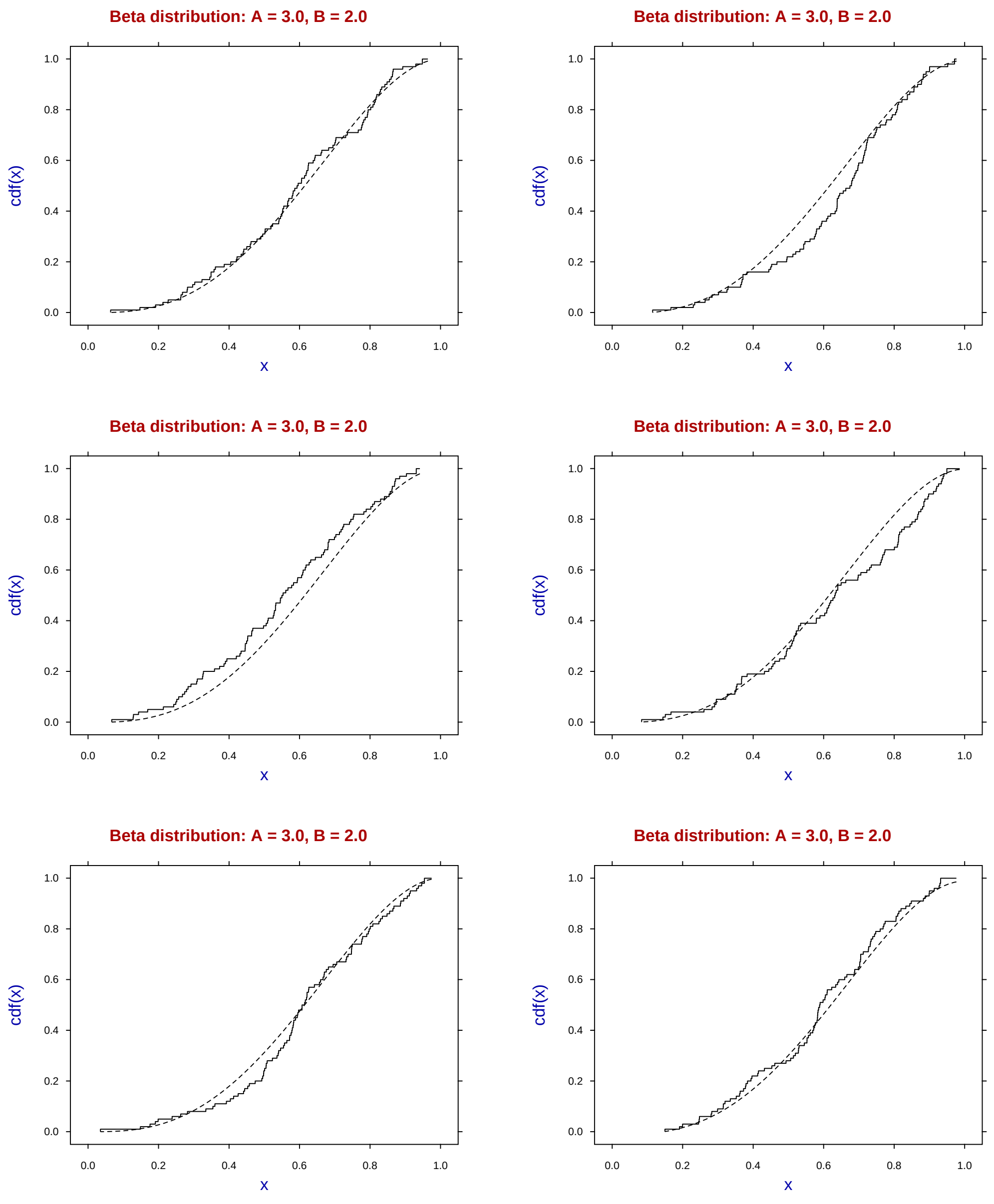


The wide variation in shape that is possible is what makes this a valuable empirical model for fitting arbitrary data that can be projected into the interval (0,1) in order to estimate and visualize skew and kurtosis. In addition the inversion of shape leading to poles at the extremes is often useful in some situations. This document explains how to use SimFIT to simulate pseudo–random beta variables and fit a beta distribution to observations using constrained weighted least squares in order to estimate goodness of fit.

7.3.1 Generating random samples

When fitting a specified probability distribution to a sample of observations it is valuable first to simulate random samples for the distribution, then observe how the values for random observations change as the parameters vary. Random samples can then be plotted as histograms or cumulative distributions to get a feel as to how well your data can be modeled by the distribution and what are likely to be reasonable parameters.

From the main SIMFIT menu choose [Simulate] then [Generate random numbers and walks] which opens up program **rannum**, and then select to generate sequences of random numbers for the stated distribution and parameters. As an example, consider the following six samples with 100 observations using a beta distribution with $\alpha = 3$ and $\beta = 2$ which demonstrates some rather surprising issues.



It will be seen that, even with a fairly large sample, it is possible to get seemingly large systematic deviations (from a Kolomogorov–Smirnov perspective) between the sample cumulative distribution (solid step curve) and the theoretical distribution (dotted curves) due to the unavoidable pseudo serial correlation in the sample cumulative. This must be kept in mind when assessing the goodness of fit by subjective graphical inspection of this type.

Of course things are no better with displaying the theoretical PDF overlayed on a histograms, as histogram shape depends on the number of bins chosen. At this point it should be noted that the data exploration option in program **simstat** allows users to examine such PDF and CDF overlays for chosen distributions using any sample of observations.

7.3.2 Parameter estimation for statistical distributions

Before describing methods to estimate parameters for selected statistical distributions, such as the beta distribution from samples of observations, three points should be considered.

1. Experimental observations do not often follow statistical distributions exactly, rather distributions are assumed for convenience. For instance, the distribution of biological variables such as height, weight, blood pressure, etc., in populations are often analyzed as if the data followed a Gaussian distribution, which may appear reasonable in practise but is impossible mathematically, because the Gaussian distribution assumes $-\infty \le X \le \infty$.
2. Mathematical statistics is based on such precisely defined variables but everything that is measured experimentally has unavoidable observational error in addition to natural variation.
3. Many methods for parameter estimation depend on sample moments and it is well know that, apart from perhaps the first moment in some situations, higher moments are themselves parameter estimates with large variances, that is, are very innacurate.

For such reasons there is something to be said for estimating parameters by constrained nonlinear regression which offers the possibility of calculating parameter confidence limits and assessing goodness of fit by residuals analysis. That is, arranging a single sample of observations into a form suitable for fitting a statistical distribution as if it were a model for constrained weighted least squares fitting. Usually this means fitting a PDF to a histogram, or a CDF to a sample cumulative. If a very large sample is available, or observations are only available as already partitioned into bins, then fitting a histogram could be considered, as long as it is realized that the result will depend on the number of bins chosen.

7.3.3 Preparing samples of observations for curve fitting

Data must be available as vector, that is, a single column of values with no labels or missing values, and then this is input into the SimFIT program for exhaustive analysis of a sample. This can be opened from the main SimFIT menu by choosing [Data exploration]. There are then two options that will prove useful, and both are available using program **rannum**.

1. **Exhaustive analysis of an arbitrary vector**

   This allows you to create a PDF file for fitting by choosing the number of bins required then creating a PDF curve fitting file where the histogram area is scaled to one. Alternatively you can create a CDF curve fitting file. From this procedure you can also calculate the sample moments if these are needed to estimate starting estimates.

2. **Comparing data with a known distribution**

   A distribution is chosen then the parameters are varied until a reasonable fit is apparent when the data are displayed as a PDF–histogram or CDF–cumulative plot. The values chosen can then be used as starting estimates.

Another issue that is often considered is the minimum sample size that is required to begin to justify concluding that a specified distribution with the estimate parameters does reasonably represent the data. Rules of thumb such as . . . *at least ten times the number of parameters* . . . or similar are often suggested which would mean 20 for the beta distribution, but experience indicates a minimum sample size of about 100. So we now turn to a worked example using a beta distribution with a sample size of 100 and $\alpha = 3$ and $\beta = 2$, where the mode is shifted slightly to the right.

7.3.4 Fitting a beta pdf

A random sample contained in the file `beta32_data.tf1` was generated by program **rannum** then transformed into a pdf–fitting histogram file with area one by the option to perform exhaustive analysis of a vector, leading to the curve–fitting data file `beta32_pdf.tf1` shown below.

```
beta pdf fitting file generated by RANNUM: A = 3, B = 2
10   3
0.18144613   0.60634121   1
0.26390795   0.84887769   1
0.34636977   0.72760945   1
0.42883159   1.8190236    1
0.51129342   1.8190236    1
0.59375524   1.2126824    1
0.67621706   1.3339507    1
0.75867888   1.3339507    1
0.84114070   1.6977554    1
0.92360252   0.72760945   1
begin{limits}
1   1   5
1   1   5
end{limits}
```

The first column contains the centers of the ten histogram bins, and the second column contains the scaled frequencies, while the third column (with weights equal to one) indicates that unweighted fitting is to be used.

The section starting with the token `begin{limits}` and ending with the token `end{limits}` gives the lower limits, the starting estimates, then the upper limits to be used by program **qnfit** for constrained nonlinear regression in the EXPERT mode, i.e. where such estimates are appended to the data file.

The results from fitting by the SIMFIT quasi–Newton constrained optimization technique are shown next.

Number	Low-Limit	High-Limit	Value	Std.Error	Lower95%cl	Upper95%cl	p
1	1	5	2.26957	0.454304	1.22195	3.31720	0.0011
2	1	5	1.70249	0.301995	1.00609	2.39889	0.0005
3	1	1	1.00000	0.000000	1.00000	1.00000	fixed

For 50,90,95,99% con. lim. using [parameter value +/- t(alpha/2)*std.err.]
t(.25) = 0.706, t(.05) = 1.860, t(.025) = 2.306, t(.005) = 3.355

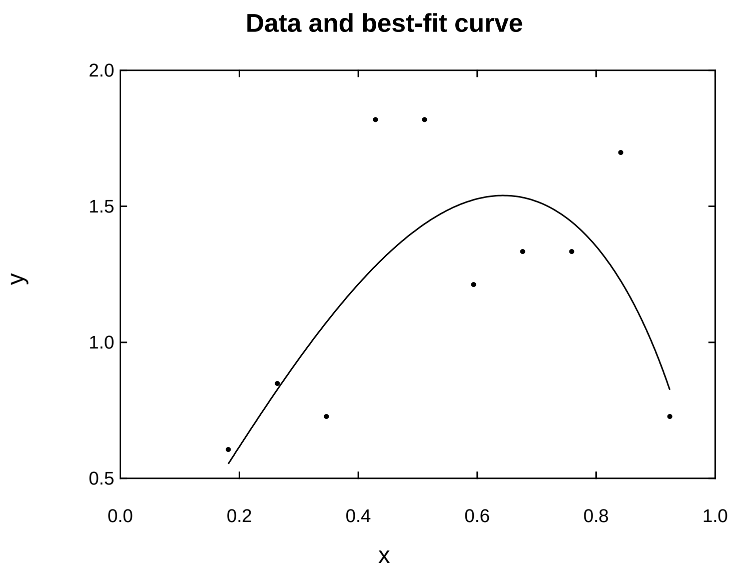
Note that the model used by SIMFIT has the following parameter definitions.

$$p(1) = \alpha$$
$$p(2) = \beta$$
$$p(3) = \Delta$$

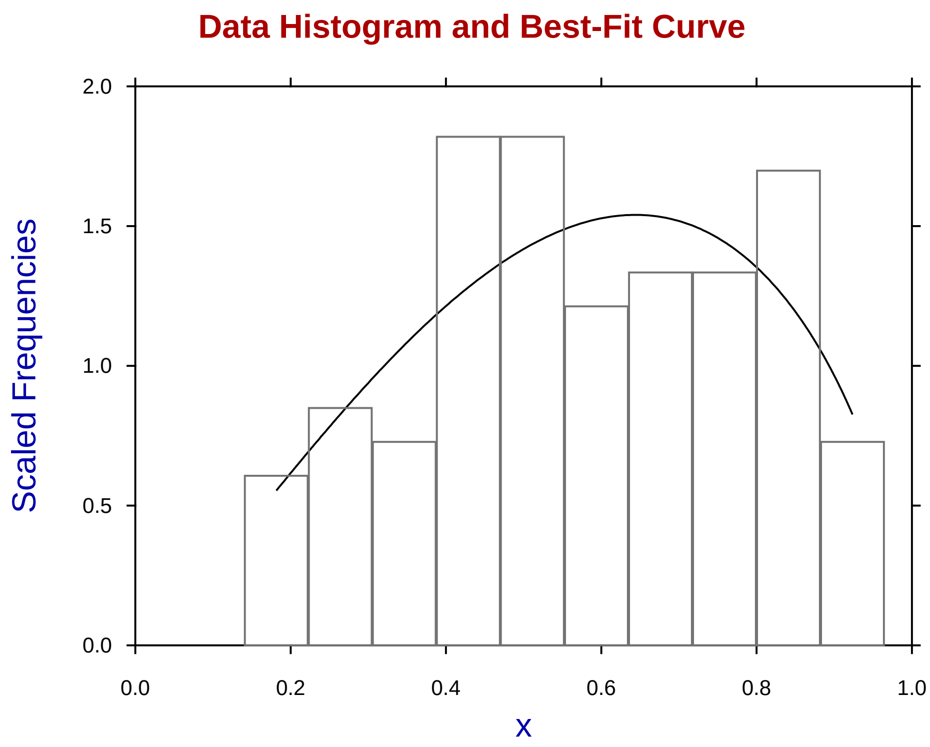
where Δ is a scaling factor than can be used if the area under the histogram is not one. For this fit $p(3)$ was not varied but was fixed, i.e., $\Delta = 1$.

After listing all the goodness of fit results program **qnfit** first shows a default graph where the tops of the histogram bins are shown as dots and the best–fit curve is displayed as a smooth curve ranging between the the centers of the first and last histogram bins.

This default graph from program **qnfit** is shown next.



Here is the default graph after editing to replace the dots by outline type histogram bars width 1.47, and other obvious changes, to give the next graph.



It is possible to create such graphs with many more possible options by saving the best–fit curve parameters, then reading the data into the Data Exploration option of program **simstat** to create the histogram overlayed by the pdf for a beta distribution with the best-fit parameters over the full range, etc.

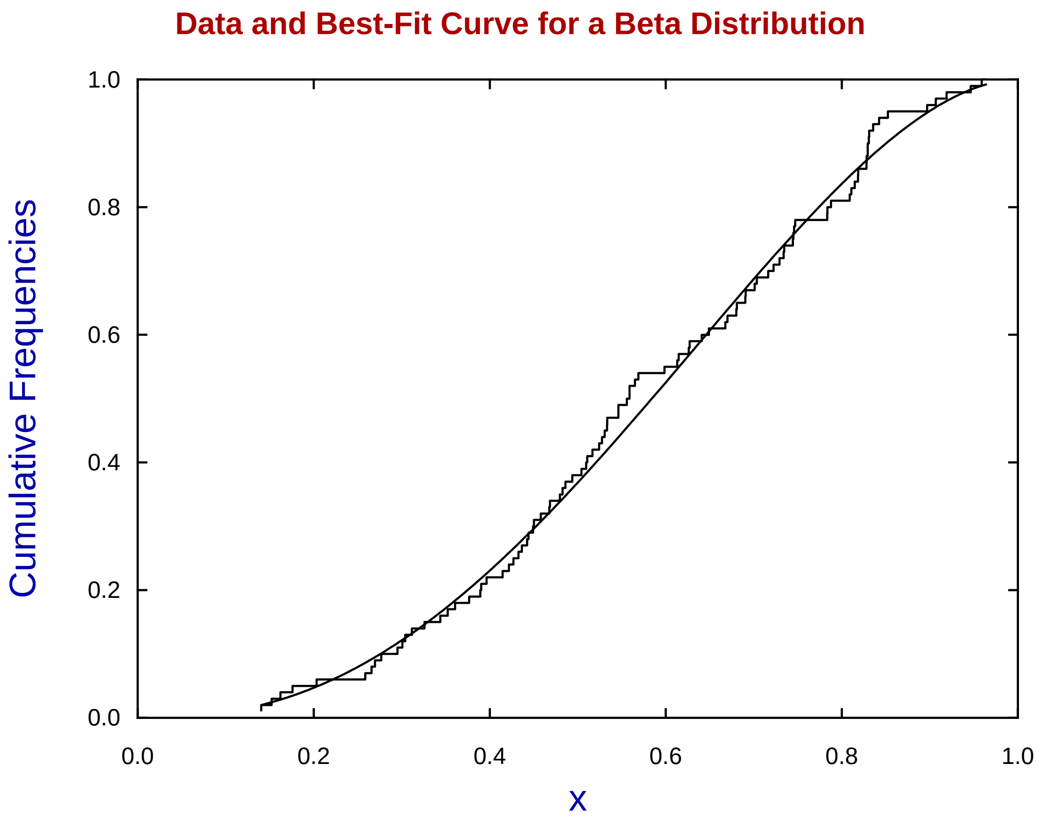
7.3.5 Fitting a beta cdf

Proceeding as before then fitting a beta cdf to the data file `beta32_cdf.tf1` using program **qnfit** yields these parameter estimates.

Number	Low-Limit	High-Limit	Value	Std.Error	Lower95%cl	Upper95%cl	p
1	1	5	2.52779	0.0557879	2.41708	2.63850	0.0000
2	1	5	1.88851	0.0403438	1.80845	1.96857	0.0000
3	1	1	1.0	0.0	1.0	1.0	fixed

For 50,90,95,99% con. lim. using [parameter value +/- t(alpha/2)*std.err.]
t(.25) = 0.677, t(.05) = 1.661, t(.025) = 1.984, t(.005) = 2.627

The following best-fit curve was edited by simply replacing the default plotting symbols (dots with no lines) for the data by no symbols but a cdf-type step curve.



It is clear that fitting such simple models with just two varied parameters gives well–defined parameter estimates ($p = 0$) but fitting the cdf using all 100 points gives better estimates than fitting to a histogram which only fits ten points.

To quantify this observation, the procedure of data generation by program **rannum** followed by fitting using program **qnfit** was repeated, and the Euclidean distance D between the estimates $(\hat{\alpha}, \hat{\beta})$ and the actual parameter values (α, β) was calculated, where D is defined as follows,

$$D = \sqrt{(\alpha - \hat{\alpha})^2 + (\beta - \hat{\beta})^2}.$$

$\hat{\alpha}$	$\hat{\beta}$	D	Type	Best Fit
2.38518	1.84594	0.633828	pdf: $\alpha = 3, \beta = 2$	
2.52170	2.06080	0.482149	cdf: $\alpha = 3, \beta = 2$	cdf
2.60811	1.65327	0.523259	pdf: $\alpha = 3, \beta = 2$	
2.63019	1.76255	0.439479	cdf: $\alpha = 3, \beta = 2$	cdf
2.26957	1.70249	0.788695	pdf: $\alpha = 3, \beta = 2$	
2.52799	1.88851	0.484998	cdf: $\alpha = 3, \beta = 2$	cdf
1.94617	3.97530	0.059226	pdf: $\alpha = 2, \beta = 4$	
1.85620	3.86457	0.197534	cdf: $\alpha = 2, \beta = 4$	pdf
1.64910	3.79293	0.407442	pdf: $\alpha = 2, \beta = 4$	
1.67319	4.03689	0.328885	cdf: $\alpha = 2, \beta = 4$	cdf
2.45517	4.91705	1.023797	pdf: $\alpha = 2, \beta = 4$	
2.37186	4.82599	0.905836	cdf: $\alpha = 2, \beta = 4$	cdf
2.13494	7.54346	0.476065	pdf: $\alpha = 2, \beta = 8$	
2.12449	8.17192	0.212260	cdf: $\alpha = 2, \beta = 8$	cdf

From this table, where both the pdf and cdf were fitted to the same data set as both histograms (observations pooled into 10 bins) and cumulative frequencies (all 100 observations) for a total of seven separate simulations, a number of tentative conclusions can be drawn.

- The parameters were estimated rather better using the data in cumulative distribution format.
- There is a tendency to underestimate the parameters.

Although not shown, there is an improvement in parameter estimates when the additional normalizing parameter is allowed to vary, more so with histograms of course. However there are other ways to decide which technique to use.

7.3.6 Plotting a combined graph

Often beta distributions are plotted simply to estimate the extent to which the mode is skewed away from the central position, and this is most convincingly seen in histograms as long as the number of bins is not too large.

So, as there are only two, or rarely three parameters to be fitted and the beta distribution is robust as an empirical model and easy to fit to a sample of observations, there seems no reason why both should not be fitted at the same time.

For a combined graph from such a fitting procedure there are two considerations.

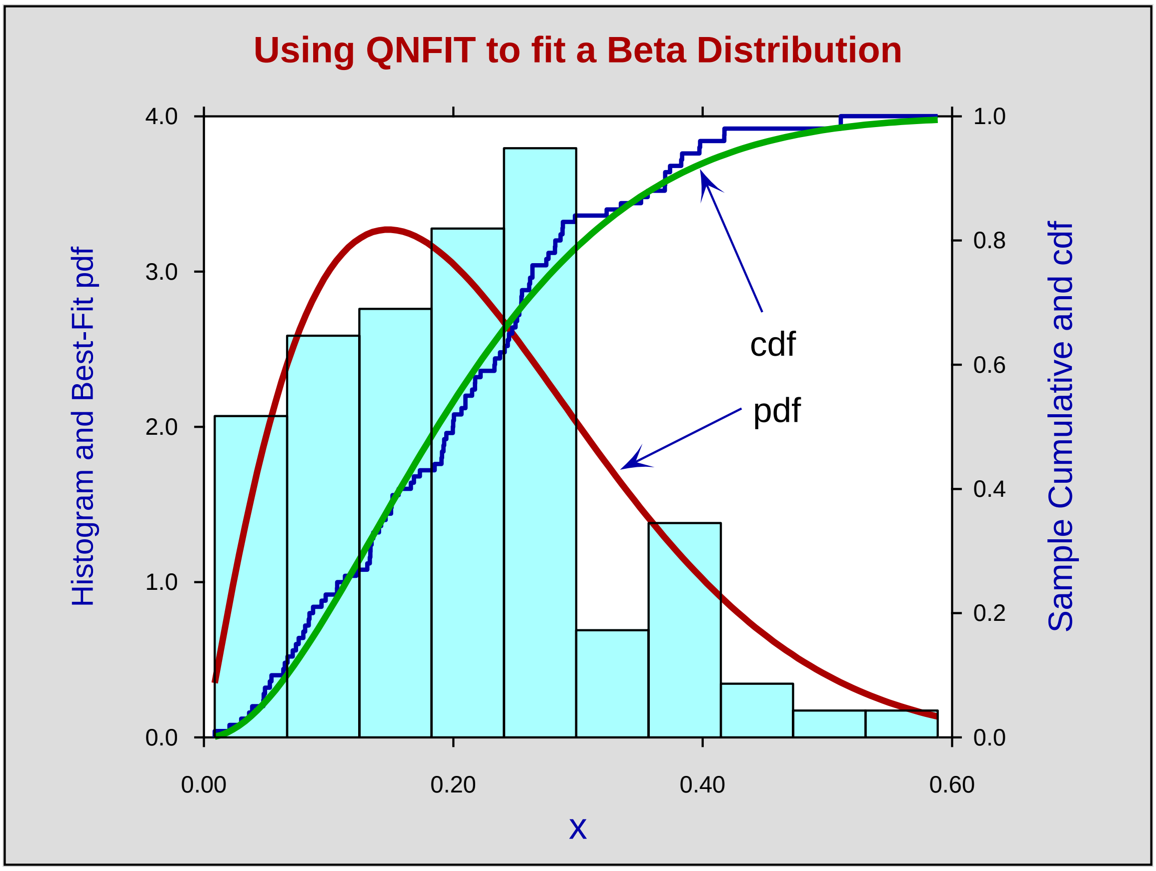
1. Four files of coordinates are required, that is:
   - File 1: coordinates for the histogram;
   - File 2: coordinates for the best–fit pdf;
   - File 3: coordinates for the sample cumulatives; and
   - File 4: coordinates for the best-fit cdf.
2. Two scales are required for the vertical axes, such as:
   - plotting the histogram and pdf using a left–hand scale; and
   - plotting the sample cumulative and cdf using a right–hand scale.

There are several methods by which this process can be done. Perhaps the most obvious is to save the coordinate files from the graphs of data and best fit graphs displayed by program **qnfit**, but this is not necessarily the best way.

Probably the best and easiest SimFIT technique do this, for instance, for a beta distribution like the one from the previous table with $\alpha = 2, \beta = 8$, is as follows.

1. Open the option in the SimFIT program **simstat** to compare a sample with an assumed distribution.
2. Read in the data and construct a histogram plotted against a beta distribution with best–fit parameters $\hat{\alpha} = 2.13494, \hat{\beta} = 7.54346$.
3. From this save the coordinates to File 1 and File 2.
4. Now construct a cumulative distribution stair-step type plot with added cdf with best–fit parameters $\hat{\alpha} = 2.12449, \hat{\beta} = 8.17192$.
5. From this save the coordinates to File 3 and File 4.
6. Open program **simplot** then choose two create a double axis plot and read in File 1 and File 2 to plot against the left–hand Y-axis, then File 3 and File 4 for the right–hand axis.

All that remains is fine tuning to create the following plot.



It should be noted that plotting symbols can be replaced by filled polygons, but if these are filled with color the histogram bin outlines will be lost. This can be overcome by using the same file (File 1) added interactively as an additional file (File 5) used to outline the resulting filled polygons. Alternatively, if this situation is anticipated, an additional copy of File 1 containing the histogram outlines can be added right from the start.

7.3.7 Practical issues

As the shape of the data will be evident before any computation of best–fit parameters, then visual inspection helps in the choice of starting estimates and limits.

Writing the beta probability density function, i.e. the PDF, in the following form

$$f_x(x:\alpha,\beta)=\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$

emphasizes that, as the complete beta function itself, i.e., $B(\alpha,\beta)$ is a constant and not dependent on x, the graphical behaviour of this density function for $0 \le x \le 1$ depends only on the expression

$$x^{\alpha-1}(1-x)^{\beta-1}$$

so there is a single turning point for non-degenerate cases at the mode M where

$$M=\frac{\alpha-1}{\alpha+\beta-2}.$$

As $M = 0.5$ when $\alpha = \beta$, while $M > 0.5$ if $\alpha > \beta$, and $M < 0.5$ if $\alpha < \beta$, the displacement of the mode from 0.5 indicates the relative magnitude of α and β. Of course the degenerate case when $\alpha = \beta = 1$ corresponding to a uniform distribution, the complications due to vertical asymptotes when $x = 0$ for $\alpha < 1$ and $x = 1$ for $\beta < 1$, along with the general inversion of shape when $\alpha < 1$ and $\beta < 1$ must be considered. As the general shape would be indicated by the data then this means it is easy to decide on the lower limits of 1 when $\alpha > 1$ and $\beta > 1$ and upper limits of 1 when $\alpha < 1$ and $\beta < 1$.

There are two other practical issues to consider when fitting the beta distribution to observations.

1. **The range of x values**

   In the cases where $\alpha > 1$ and $\beta > 1$ then there is no restriction of range and observations can be anywhere between $x = 0$ and $x = 1$. However, if vertical asymptotes are anticipated, then values must be restricted near potential asymptotes so that computation does not lead to overflow.

2. **The parameter limits**

   As computation of best–fit parameters proceeds then, at every fixed value of x, the values of the internal estimates $\hat{\alpha}$ and $\hat{\beta}$ are perturbed by factors of the order of machine precision. So the upper and lower limits should normally be chosen such that singular cases are avoided.

   Another issue concerns the evaluation of the complete beta function for non–integer arguments. As α and β become larger then the time taken to evaluate the complete beta function increases very rapidly. Of course the computer code doing this is is optimized, but is still faced with such limitations. So it is recommended that the upper limits requested for parameter estimates should be selected conservatively with this in mind to avoid lengthy computations.

7.4 Plotting the objective function using qnfit

SIMFIT tries to minimize $WSSQ/NDOF$ which has expectation unity at solution points, and it is useful to view this as a function of the parameters close to a minimum. Figure 7.3 shows the objective function from fitting a single exponential model

$$f(t) = A\exp(-kt)$$

to the test file `exfit.tf2`.

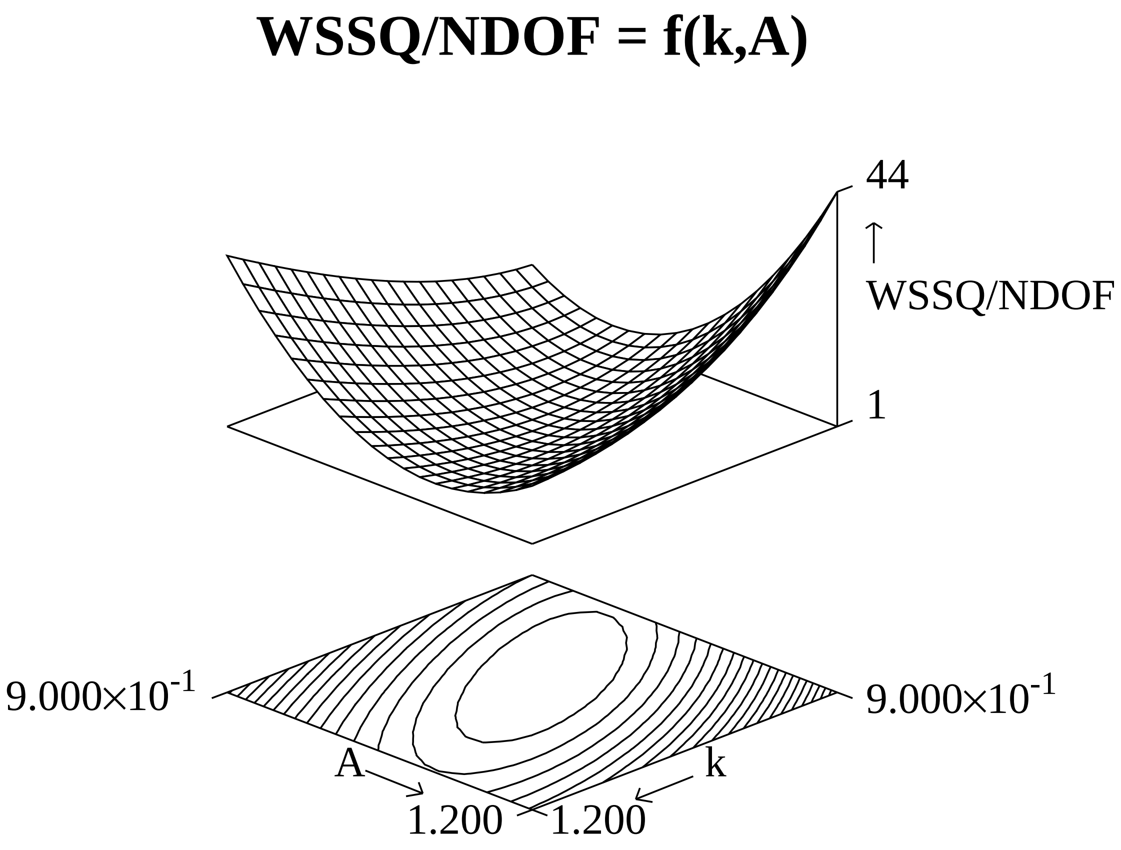


Figure 7.3: Objective function from an exponential model

Figure 7.4 shows details from fitting a simple Michaelis-Menten steady state model

$$v(S) = \frac{V_{max}S}{K_m + S}$$

with two parameters to data in the test file `mmfit.tf2`, and a double Michaelis-Menten model

$$v(S) = \frac{V_{max(1)}S}{K_{m(1)} + S} + \frac{V_{max(2)}S}{K_{m(2)} + S}$$

with four parameters, of which only two are varied for the plot, to data in test file `mmfit.tf4`.

Such plots are created by **qnfit** after fitting, by selecting any two parameters and the ranges of variation required. Information about the eccentricity is also available from the parameter covariance matrix, and the eigenvalues and condition number of the Hessian matrix in internal coordinates. Some contour diagrams show long valleys at solution points, sometimes deformed considerably from ellipses, illustrating the increased difficulty encountered with such ill-conditioned problems.

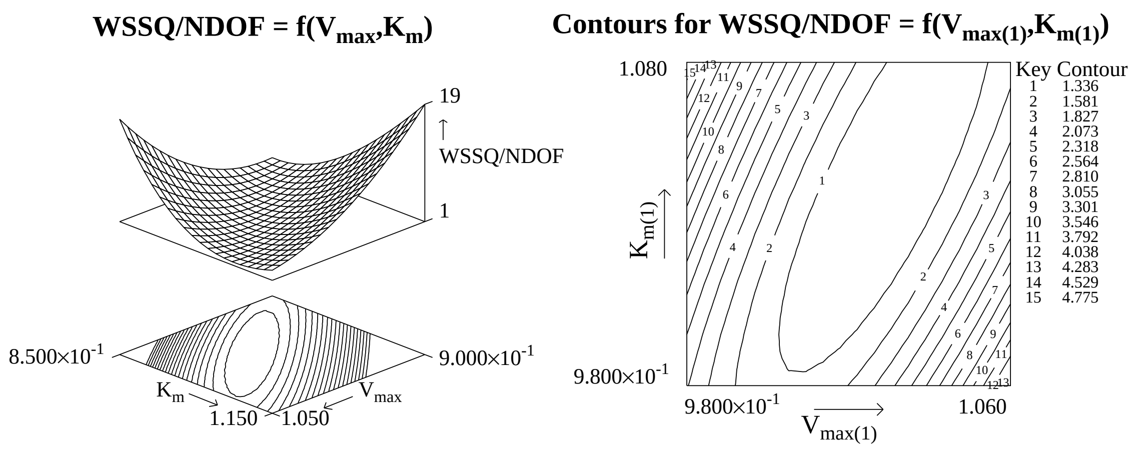


Figure 7.4: Objective function from Michaelis-Menten models

7.5 Plotting best-fit surfaces using qnfit

When fitting functions of two variables it is often useful to plot best-fit models as a three dimensional surface, or as a contour diagram with the two variables plotted over a range. For instance, figure 7.5 shows the best fit surface after using **qnfit** to fit the two-variable inhibition kinetics model

$$v(S, I) = \frac{V_{max}[S]}{K_m + [S](1 + [I]/K_i)}.$$

Note that such surfaces can also be created for functions of two variables using **makdat**.

It is also valuable to be able to plot sequential slices across best-fit surfaces to assess goodness of fit visually when functions of two variables have been fitted. So, as in figure 7.6, **qnfit** allows slices to be cut through a best fit surface for fixed values of either variable. Such composite plots show successive sections through a best fit surface, and this is probably the best way to visualize goodness of fit of a surface to functions of two variables.

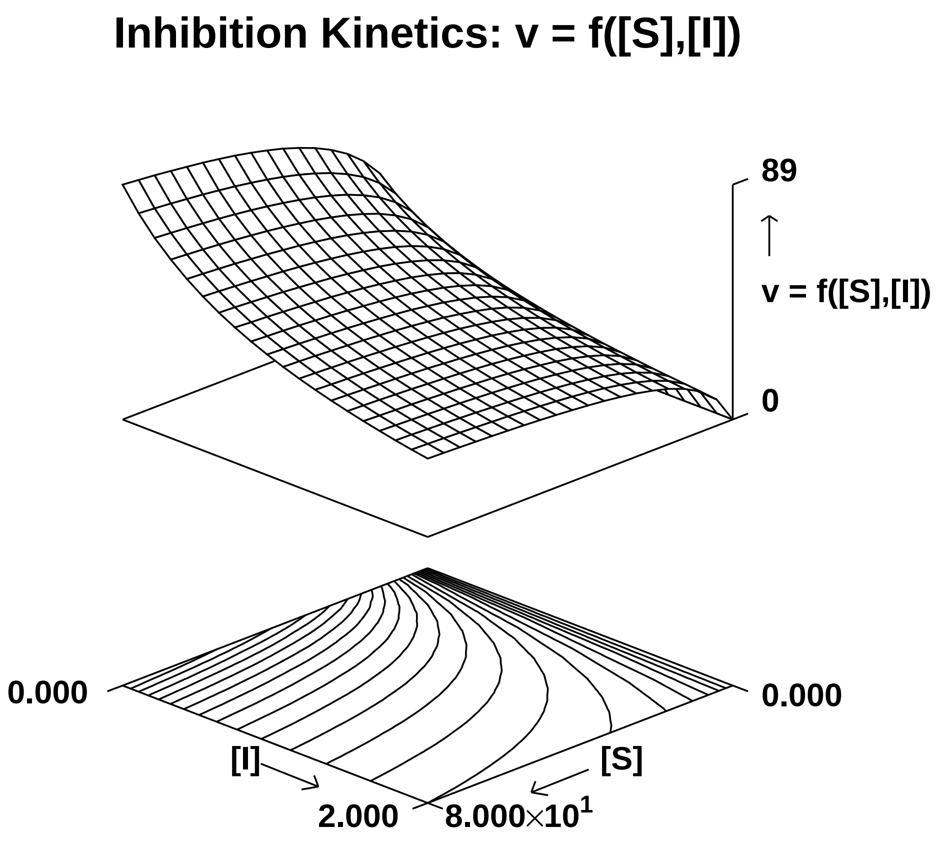


Figure 7.5: Best-fit two variable model surface

7.6 Fitting functions of several variables using qnfit

The only feature that differs from functions of one variable is that overlaying the current best-fit curve on the data before fitting and several other graphical techniques are not possible. Table 7.2 illustrates the outcome

```
Number Lower-Limit Upper-Limit      Value Std. Err. ..95% Conf. Lim..   p
  1           0.0         5.0  0.082438   0.01237  0.05548  0.1094  0.0000
  2           0.0         5.0     1.134    0.3082   0.4626   1.805  0.0031
  3           0.0         5.0    2.3428    0.2965    1.697   2.989  0.0000

 Parameter correlation matrix  0.753
                             -0.725 -0.997

Eigenvalues of Hessian 8.500E-04, 3.364E-01, 1.456E+00,  Cond. no. =  1.713E+03
```

Table 7.2: Results from fitting a function of three variables

from fitting the NAG library E04FYF/E04YCF example

$$f(x, y, z) = p_1 + \frac{x}{p_2 y + p_3 z}.$$

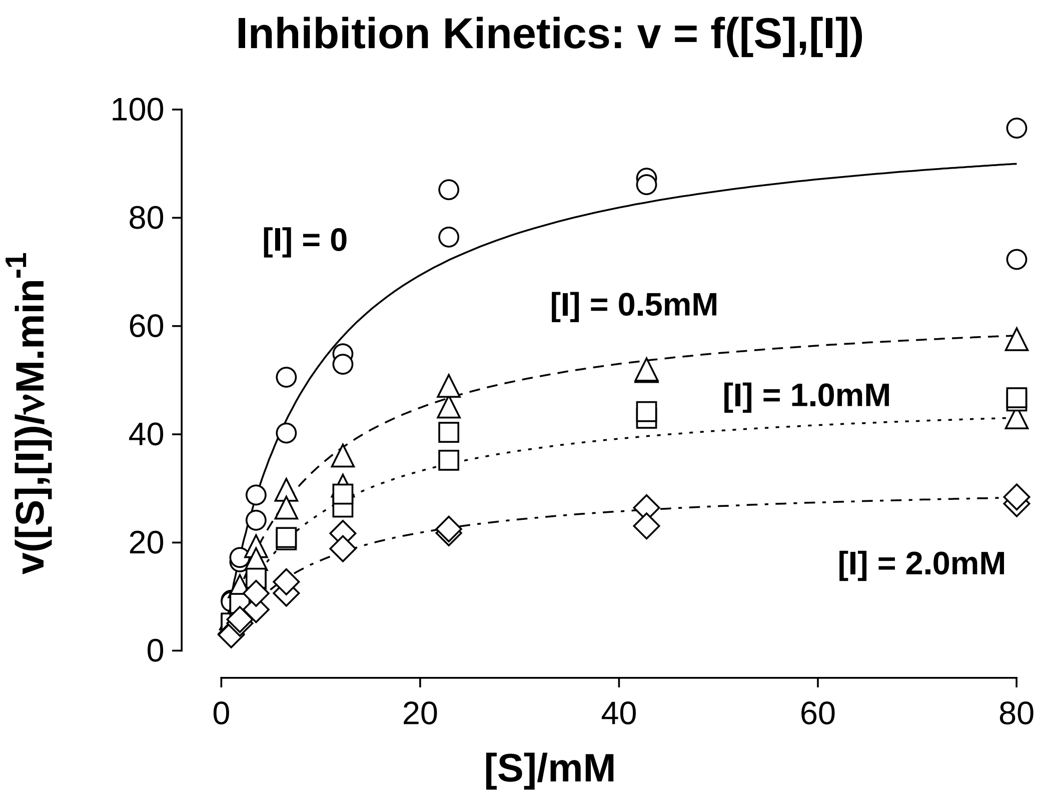


Figure 7.6: Sequential sections across a best-fit surface

7.7 Fitting multi-function models using qnfit

Often m distinct data sets are obtained from one experiment, or independently from several experiments, so that several model equations must be fitted at the same time. In this case the objective function is the normalized sum of the weighted sums of squares from each data set, i.e.

$$\frac{WSSQ}{NDOF} = \sum_{j=1}^{m}\sum_{i=1}^{n_j}\left(\frac{y_{ij} - f_j(x_{ij}, \Theta)}{s_{ij}}\right)^2 \bigg/ \left(\sum_{j=1}^{m} n_j - k\right)$$

and there are n_j observations in data set j for $j = 1, 2, \ldots, m$, and k parameters have been estimated for the m models f_j. Usually the models would not be disjoint but would have common parameters, otherwise the data sets could be fitted independently.

This can be best illustrated using a very simple model, three distinct straight lines. Open program **qnfit** and select to fit n functions of one variable. Specify that three equations are required, then read in the library file `line3.tfl` containing three data sets and select the model file `line3.mod` (page 457) which defines three independent straight lines. Choose expert mode and after fitting figure 7.7 will be obtained. Since, in this case, the three component sub-models are uncoupled, the off-diagonal covariance matrix elements will be seen to be zero. This is because the example has been selected to be the simplest conceivable example to illustrate how to perform multi-function fitting.

The main points to note when fitting multiple functions are as follows.

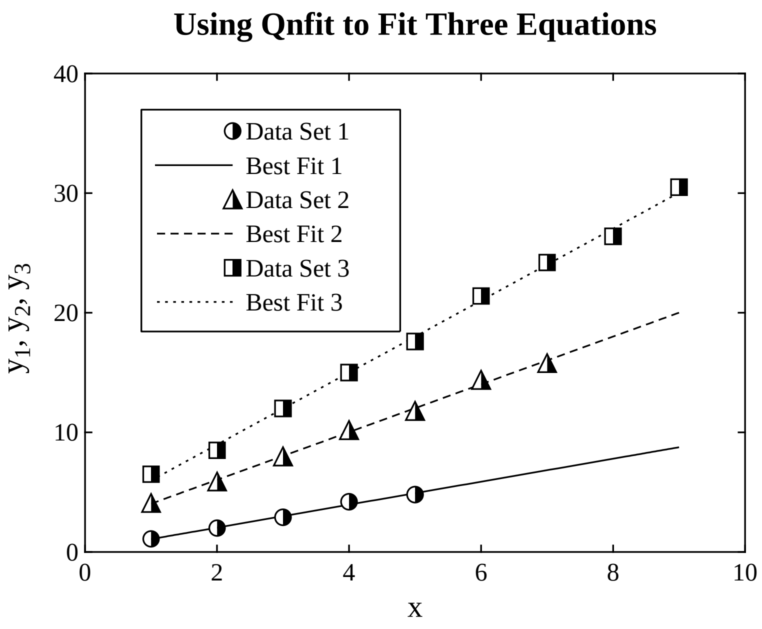


Figure 7.7: Fitting three equations simultaneously

- You must provide the data sets as a library file.
- Missing data sets in the library file can be denoted using a percentage sign %.
- Expert mode starting parameters, if required, must be added to the first non-missing data file specified in the library file.
- You must provide the model as a user-defined model file.

A more instructive example is using data in test file `consec3.tfl` together with the model file `consec3.mod` (page 460) to fit two irreversible consecutive chemical reactions involving three species $A \longrightarrow B \longrightarrow C$ with $A(0) = A_0 > 0$, $B(0) = C(0) = 0$ when the equations are

$$\begin{aligned}
A(x) &= A_0 \exp(-k_1 x)\\
B(x) &= \frac{k_1}{k_2 - k_1} A_0(\exp(-k_1 x) - \exp(-k_2 x)), \text{ when } k_1 \neq k_2\\
&= A_0 x \exp(-kx), \text{ when } k_1 = k_2 = k\\
C(x) &= A_0 - A(x) - B(x).
\end{aligned}$$

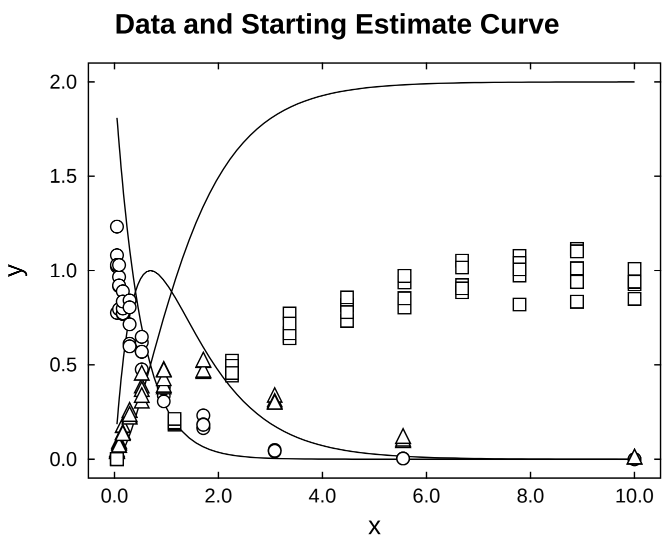
File `consec3.mod` can be browsed to see how use the SIMFIT *if . . . then . . . else . . .* technique (page 440) in a user-defined model to evaluate the required expression for $B(x)$, and the results are illustrated in figure 7.8.

7.8 Fitting a convolution integral using qnfit

Fitting convolution integrals (page 439) involves parameter estimation in $f(t)$, $g(t)$, and $(f * g)(t)$, where

$$(f * g)(t) = \int_0^t f(u)g(t-u)\,du,$$

and such integrals occur as output functions from the response of a device to an input function. Sometimes the input function can be controlled independently so that, from sampling the output function, parameters of



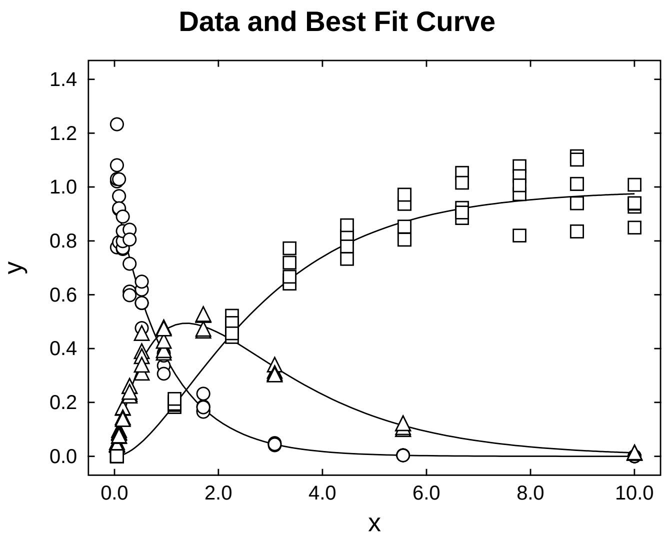


Figure 7.8: Fitting consecutive chemical reactions

the response function can be estimated, and frequently the functions may be normalized, e.g. the response function may be modeled as a function integrating to unity as a result of a unit impulse at zero time. However, any one, two or even all three of the functions may have to be fitted. Figure 7.9 shows the graphical display following the fitting of a convolution integral using **qnfit** where, to demonstrate the procedure to be followed for maximum generality, replicates of the output function at unequally spaced time points have been assumed. The model is `convolv3.mod` (page 462) and the data file is `convolv3.tfl`, which just specifies replicates for the output function resulting from

$$\begin{aligned} f(t) &= \exp(-\alpha t) \\ g(t) &= \beta^2 t \exp(-\beta t). \end{aligned}$$

Note how missing data for $f(t)$ and $g(t)$ are indicated by percentage symbols in the library file so, in this case, the model `convolve.mod` could have been fitted as a single function. However, by fitting as three functions of one variables but with data for only one function, a visual display of all components of the convolution integral evaluated at the best-fit parameters can be achieved.

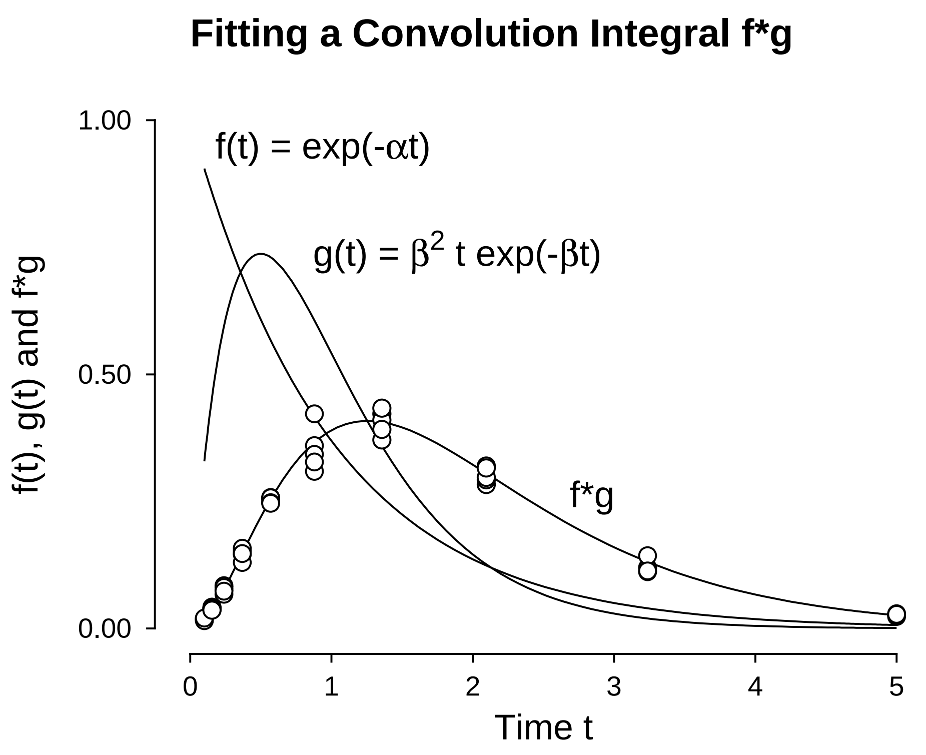


Figure 7.9: Fitting a convolution integral

Part 8

Differential equations

8.1 Introduction

Program **makdat** can simulate single differential equations, program **qnfit** can fit single differential equations, and program **deqsol** can simulate and fit systems of differential equations. To do this the equations must be either used from the library or supplied as user-defined models. Systems of n equations must be formulated as a set of n simple expressions and a Jacobian can be supplied or omitted. If a Jacobian is to be supplied the solution must be compared using the BDF technique both with and without the supplied Jacobian to make sure the supplied jacobian has been coded correctly.

8.2 Phase portraits of plane autonomous systems

When studying plane autonomous systems of differential equations it is useful to plot phase portraits. Consider, for instance, a simplified version of the Lotka-Volterra predator-prey equations given by

$$\frac{dy(1)}{dx} = y(1)(1 - y(2))$$
$$\frac{dy(2)}{dx} = y(2)(y(1) - 1)$$

which has singular points at (0,0) and (1,1). Figure 8.1 was generated using **deqsol**, then requesting a phase portrait. You can choose the range of independent variables, the number of grid points, and the precision required to identify singularities. At each grid point the direction is defined by the right hand sides of the defining equations and singular points are emphasized by automatic change of plotting symbol. Note that arrows can have fixed length or have length proportional to magnitude.

8.3 Orbits of differential equations

To obtain orbits with $y(i)$ parameterized by time as in figure 8.2, trajectories have to be integrated and collected together. For instance the simple system

$$\frac{dy(1)}{dx} = y(2)$$
$$\frac{dy(2)}{dx} = -(y(1) + y(2))$$

which is equivalent to

$$y'' + y' + y = 0$$

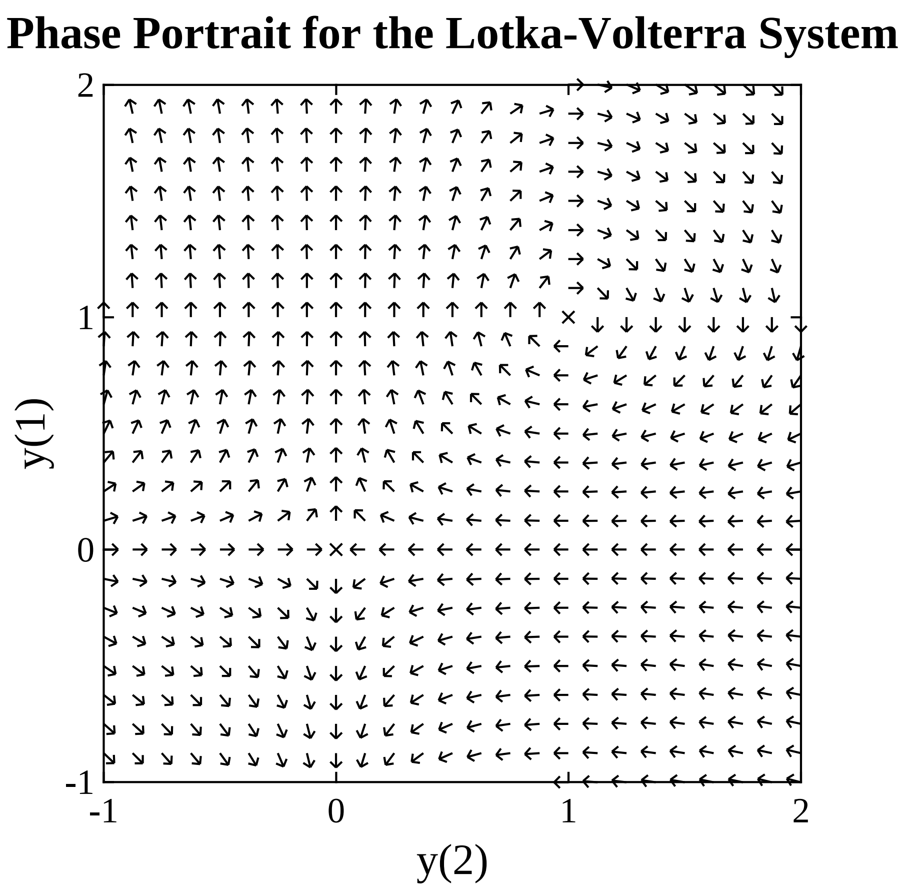


Figure 8.1: Phase portraits of plane autonomous systems

was integrated by **deqsol** for the initial conditions illustrated, then the orbits were used to create the orbit diagram. The way to create such diagrams is to integrate repeatedly for different initial conditions, then store the required orbits, which is a facility available in **deqsol**. The twelve temporary stored orbits are `f$orbits.001` to `f$orbits.012` and orbits generated in this way can also be plotted as an overlays on a phase portrait in order to emphasize particular trajectories. Just create a library file with the portrait and orbits together and choose the vector field option in program **simplot**. With this option, files with four columns are interpreted as arrow diagrams while files with two columns are interpreted in the usual way as coordinates to be joined up to form a continuous curve.

8.4 Fitting differential equations

SIMFIT can simulate systems of differential equations and fit them to experimental data just like any other model. If there are n differential equations with m parameters then there must be n additional parameters for the initial conditions. That is, parameters $p_{m+i} = y_i(0)$ for $i = 1, 2, \ldots, n$. When fitting differential equations the data must be extensive with a high signal to noise ratio and, if possible, the initial conditions should be treated as fixed parameters, or at least constrained between narrow limits. Although the main SIMFIT program for simulating and fitting differential equations is **deqsol**, it is probably more convenient to fit single equations using **qnfit**, so such examples will be described next.

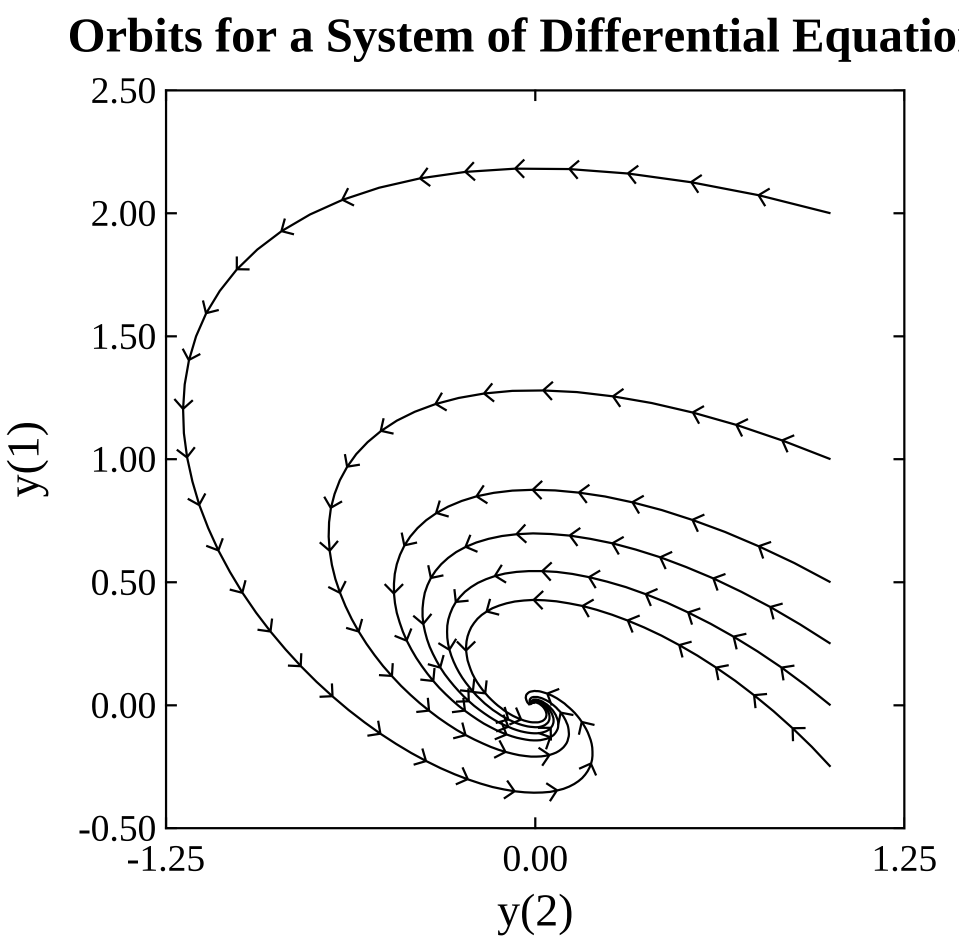


Figure 8.2: Orbits of differential equations

8.4.1 Fitting a single differential equation using qnfit

To illustrate fitting a single differential equation we will consider three examples in increasing order of difficulty for fitting.

1. Michaelis-Menten irreversible substrate depletion.
2. Von Bertalanffy allometric growth model.
3. Von Bertalanffy allometric growth/decay model.

8.4.1.1 Michaelis-Menten irreversible substrate depletion

The model equation is

$$\frac{dS}{dt} = \frac{-V_{max}S}{K_m + S}$$

which, in terms of parameters, is expressed as

$$\frac{dy}{dx} = \frac{-p_2 y}{p_1 + y}, \text{ with } p_3 = y(0).$$

With only three parameters to be estimated this model is extremely easy to fit and, in this case, can be formally integrated anyway (page 469). Note that, if the initial substrate concentration was known exactly, then $y(0)$ could have been fixed, but in this case replicates representing experimental error are included in test file `qnfit_ode.tf1`, so $y(0)$ was estimated as shown in table 8.1.

Number	Lower-Limit	Upper-Limit	Value	Std.Err.	95% Con.Lim.		p
1	0.5	1.5	1.0965	0.07901	0.9371	1.256	0.0000
2	0.5	1.5	1.0927	0.07358	0.9442	1.241	0.0000
3	0.5	1.5	1.0466	0.2232	1.002	1.092	0.0000

Table 8.1: Parameters for Michaelis-Menten irreversible substrate depletion

8.4.1.2 Von Bertalanffy allometric growth model

The data in test file `qnfit_ode.tf2` were read into **qnfit** and the Von Bertalanffy differential equation

$$\begin{aligned}\frac{dy}{dx} &= Ay^m - By^n\\ &= p_1y^{p_2} - p_3y^{p_4}, \text{ with } p_5 = y(0)\end{aligned}$$

was fitted giving the results in figure 8.3.

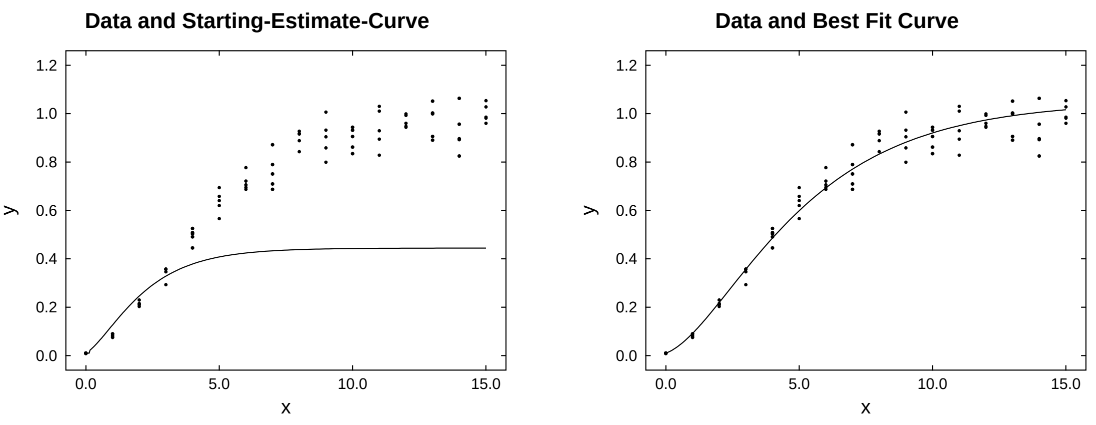


Figure 8.3: Fitting the Von Bertalanffy growth differential equation

Note that this differential equation must have $A > 0$, $B > 0$, $n > m > 0$ and $y(0) > 0$ in order to model growth curves, and it cannot model growth curves with turning points since it is autonomous.

It should also be emphasized that this model is over-parameterized so that the parameter estimates are highly correlated. This means that widely varying sets of parameter estimates can give more or less equally good fit. Neverthless, despite this limitation, the best-fit curve is still useful as an empirical data smoothing curve as explained next.

Often, after fitting equations to data, users wish to estimate derivatives, or areas, or use the best-fit curve as a standard curve in calibration. So, to illustrate another feature of **qnfit** the ability to inverse predict X given Y after fitting was exploited with results as in table 8.2. Another feature that is often required after fitting is to estimate the minimum and maximum slope of the best-fit curve. This is particularly useful when equations are used to model growth curves, and the results from such an analysis are shown in table 8.3. This example illustrates that often, where parameters cannot be determined precisely, model fitting to data can

y input	x predicted
0.2	1.859
0.4	3.323
0.6	5.011
0.8	7.445

Table 8.2: Inverse prediction with the Von Bertalanffy differential equation

```
Maximum dy/dx = 0.139     at x = 2.39
Initial dy/dx = 0.0       at x = 0.0
Final   dy/dx = 0.009417  at x = 15.0
Minimum dy/dx = 0.0,      at x = 0.0
Maximum dy/dx = 0.1385    at x = 2.395
```

Table 8.3: Estimating derivatives with the Von Bertalanffy equation

nevertheless be justified by demonstrating that a selected model can actually reproduce the features of a data set and estimate derived results from the best-fit curve. Where, as in this example, data are asymptotic to a constant value, such a deterministic model may predict with less bias than using a polynomial or spline curve.

Figure 8.4 is a graphical illustration of the results from table 8.2 and table 8.3. The left hand figure shows

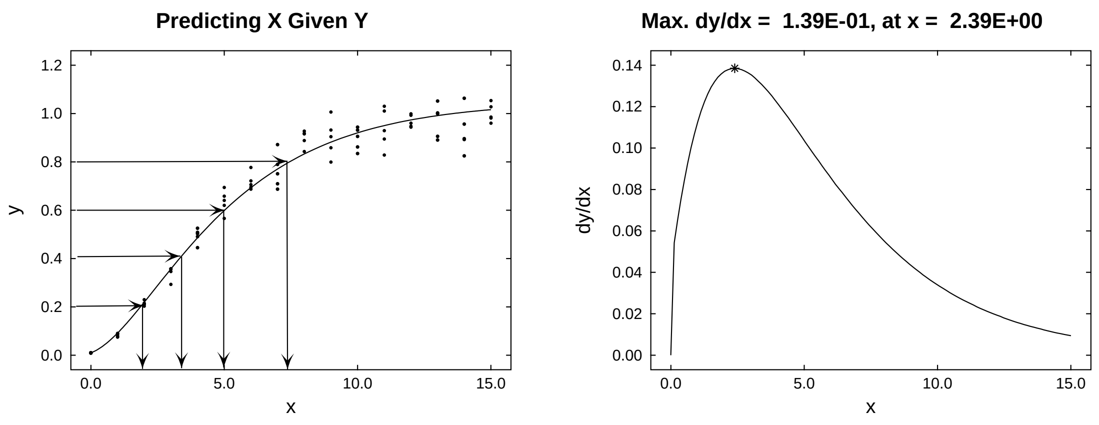


Figure 8.4: Inverse prediction with the Von Bertalanffy differential equation

what is involved in predicting X given Y by solving the appropriate equation numerically, while the right hand figure illustrates the derivative as a function of the independent variable. This can be done for all equations, not just differential equations as, again, the calculation is done numerically.

8.4.1.3 Von Bertalanffy allometric growth and decay model

An advantage of the Von Bertalanffy growth model is that it is very flexible and can fit a wide variety of typically sigmoid growth curves. However, as it is autonomous, extra parameters have to added to model other growth curve features, such as growth followed by decay. A simple modification gives

$$\frac{dy}{dx} = \exp(-p_5 x)p_1 y^{p_2} - p_3 y^{p_4}, \text{ with } p_6 = y(0)$$

where the anabolic term decreases exponentially so that, eventually, the catabolic term dominates and $y(x)$ has a turning point. The results from fitting data in the test file `qnfit_ode.tf3` are shown in figure 8.5.

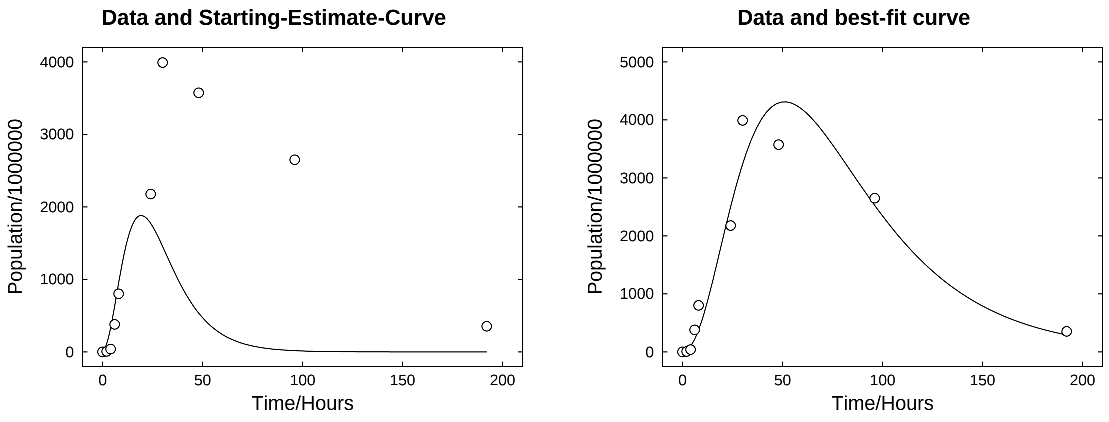


Figure 8.5: Fitting the Von Bertalanffy growth and decay differential equation

Before leaving this example we must observe table 8.4. The data in this table were obtained experimentally

Number	Lower-Limit	Upper-Limit	Value	Std.Err.	95% Con.Lim.		p
1	1.0	10.0	2.7143	0.08793	2.499	2.929	0.0000
2	0.6667	0.6667	0.6667				fixed
3	0.0001	2.0	0.1104	0.005081	0.09797	0.1228	0.0000
4	1.0	1.0	1.0				fixed
5	0.0001	0.1	0.0080919	0.001249	0.005036	0.01115	0.0006
6	1.0	50.0	3.9033	0.1058	3.644	4.162	0.0000

Table 8.4: Parameters for growth and decay

for a bacterial population with a finite food resource and, in order to obtain a good fit, it is necessary to reduce the number of parameters to be fitted. In this case the usual allometric values for the anabolic exponent p_2 and catabolic exponent p_4 were used, and these are indicated as being fixed in table 8.4. Further, the observations were weighted by the square root of the counts, to avoid overflow and other numerical difficulties caused by the large values used in this particular data set.

8.4.2 Fitting systems of differential equations using deqsol

Figure 8.6 illustrates how to use **deqsol** to fit systems of differential equations for the simple three component epidemic model

$$\frac{dy(1)}{dt} = -p_1 y(1)y(2)$$
$$\frac{dy(2)}{dt} = p_1 y(1)y(2) - p_2 y(2)$$
$$\frac{dy(3)}{dt} = p_2 y(2)$$

where $y(1)$ are susceptible individuals, $y(2)$ are infected, and $y(3)$ are resistant members of the population. Fitting differential equations is a very specialized procedure and should only be undertaken by those who understand the issues involved. For example, there is a very important point to remember when using **deqsol**: if a system of n equations involves m parameters p_i and n initial conditions $y0(j)$ for purposes of simulation, there will actually be $m + n$ parameters as far as curve fitting is concerned, as the last n parameters p_i for $i = m+1, m+2, \ldots, m+n$, will be used for the initial conditions, which can be varied or (preferably) fixed. To

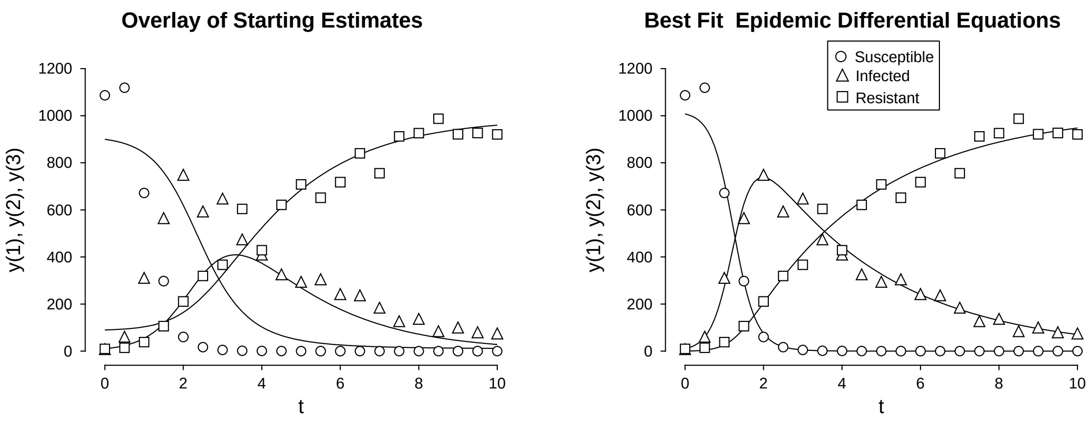


Figure 8.6: Fitting the epidemic differential equations

show you how to practise simulating and fitting differential equations, the steps followed to create figure 8.6, and also some hints, are now given.

- Program **deqsol** was opened, then the epidemic model was selected from the library of three component models and simulated for the default parameters p_1, p_2 and initial conditions $y0(1)$, $y0(2)$, $y0(3)$ (i.e. parameters p_3, p_4, and p_5).

- The data referenced in the library file `epidemic.tfl` were generated using parameter values 0.004, 0.3, 980, 10 and 10, by first writing the simulated (i.e. exact) data to files `y1.dat`, `y2.dat`, and `y3.dat`, then adding 10% relative error using **adderr** to save perturbed data as `y1.err`, `y2.err`, and `y3.err`. Program **maklib** was then used to create the library file `epidemic.tfl`, which just has a title followed by the three file names.

- Curve fitting was then selected in **deqsol**, the default equations were integrated, the library file was input and the current default differential equations were overlayed on the data to create the left hand plot. The first file referenced has a
  `begin{limits} ... end{limits}`
  section to initialize starting estimates and limits, and you should always overlay the starting solution over the data before fitting to make sure that good starting estimates are being used.

- By choosing direct curve fitting, the best fit parameters and initial conditions were estimated. It is also possible to request random starts, when random starting estimates are selected in sequence and the results are logged for comparison, but this facility is only provided for expert users, or for systems where small alterations in starting values can lead to large changes in the solutions.

- After curve fitting, the best fit curves shown in the right hand plot were obtained. In this extremely simple example the initial conditions were also estimated along with the two kinetic parameters. However, if at all possible, the initial conditions should be input as fixed parameters (by setting the lower limit, starting value and upper limit to the same value) or as parameters constrained within a narrow range, as solutions always advance from the starting estimates so generating a sort of autocorrelation error. Note that, in this example, the differential equations add to zero, indicating the conservation equation

  $$y(1) + y(2) + y(3) = k$$

  for some constant k. This could have been used to eliminate one of the $y(i)$, leading to a reduced set of equations. However, the fact that the differentials add to zero, guarantees conservation when the full set is used, so the system is properly specified and not overdetermined, and it is immaterial whether the full set or reduced set is used.

❍ Note that, when the default graphs are transferred to **simplot** for advanced editing, such as changing line and symbol types, the data and best fit curves are transferred alternately as data/best-fit pairs, not all data then all best-fit, or vice versa.

❍ For situations where there is no experimental data set for one or more of the components, a percentage sign % can be used in the library file to indicate a missing component. The curve fitting procedure will then just ignore this particular component when calculating the objective function.

❍ Where components measured experimentally are linear combinations of components of the system of differential equations, a transformation matrix can be supplied as described in the readme files.

❍ As the covariance matrix has to be estimated iteratively, the default setting which is to calculate parameter estimates with standard errors may prove inconvenient. The extra, time-consuming, step of calculating the variance covariance matrix can be switched off where this is preferred.

❍ When requesting residuals and goodness of fit analysis for any given component $y(i)$ you must provide the number of parameters estimated for that particular component, to correct for reduced degrees of freedom for an individual fit, as opposed to a total residuals analysis which requires the overall degrees of freedom

Part 9

Calibration and Bioassay

9.1 Introduction

Multivariate calibration requires the PLS technique (page 117), but univariate calibration and bioassay using SimFIT employs the following definitions.

- **Calibration**
  This requires fitting a curve $y = f(x)$ to a (x, y) training data set with x known exactly and y measured with limited error, so that the best fit model $\hat{f}(x)$ can then be used to predict x_i given arbitrary y_i. Usually the model is of no significance and steps are taken to use a data range over which the model is approximately linear, or at worst a shallow smooth curve. It is assumed that experimental errors arising when constructing the best fit curve are uncorrelated and normally distributed with zero mean, so that the standard curve is a good approximation to the maximum likelihood estimate.

- **Bioassay**
  This is a special type of calibration, where the data are obtained over as wide a range as possible, nonlinearity is accepted (e.g. a sigmoid curve), and specific parameters of the underlying response, such as the time to half-maximum response, final size, maximum rate, area AUC, $EC50$, $LD50$, or $IC50$ are to be estimated. With bioassay, a known deterministic model may be required, and assuming normally distributed errors may sometimes be a reasonable assumption, but alternatively the data may consist of proportions in one of two categories (e.g. alive or dead) as a function of some treatment, so that binomial error is more appropriate and probit analysis, or similar, is called for.

9.2 Calibration curves

Creating and using a standard calibration curve involves:

1. Measuring responses y_i at fixed values of x_i, and using replicates to estimate s_i, the sample standard deviation of y_i if possible.
2. Preparing a curve fitting type file with x, y, and s using program **makfil**, and using **makmat** to prepare a vector type data file with x_i values to predict y_i.
3. Finding a best fit curve $y = f(x)$ to minimize $WSSQ$, the sum of weighted squared residuals.
4. Supplying y_i values and predicting x_i together with 95% confidence limits, i.e. inverse-prediction of $x_i = \hat{f}^{-1}(y_i)$. Sometimes you may also need to evaluate $y_i = \hat{f}(x_i)$.

It may be that the s_i are known independently, but often they are supposed constant and unweighted regression, i.e. all $s_i = 1$, is unjustifiably used. Any deterministic model can be used for $f(x)$, e.g., a sum of logistics or Michaelis-Menten functions using program **qnfit**, but this could be unwise. Calibration curves arise from the operation of numerous effects and cannot usually be described by one simple equation. Use of such equations can lead to biased predictions and is not always recommended. Polynomials are useful for gentle curves as long as the degree is reasonably low (≤ 3 ?) but, for many purposes, a weighted least squares data smoothing cubic spline is the best choice. Unfortunately polynomials and splines are too flexible and follow outliers, leading to oscillating curves, rather than the data smoothing that is really required. Also they cannot fit horizontal asymptotes. You can help in several ways.

a) Get good data with more distinct x-values rather than extra replicates.
b) If the data approach horizontal asymptotes, either leave some data out as they are no use for prediction anyway, or try using $\log(x)$ rather than x, which can be done automatically by program **calcurve**.
c) Experiment with the weighting schemes, polynomial degrees, spline knots or constraints to find the optimum combinations for your problem.
d) Remember that predicted confidence limits also depend on the s values you supply, so either get the weighting scheme right, or set all all $s_i = 1$.

9.2.1 Turning points in calibration curves

You will be warned if $f(x)$ has a turning point, since this can make inverse prediction ambiguous. You can then re-fit to get a new curve, eliminate bad data points, get new data, etc., or carry on if the feature seems to be harmless. You will be given the option of searching upwards or downwards for prediction in such ambiguous cases. It should be obvious from the graph, nature of the mathematical function fitted, or position of the turning point in which direction the search should proceed.

9.2.2 Calibration using polnom

For linear or almost linear data you can use program **linfit** (page 40) which just fits straight lines of the form

$$f(x) = p_0 + p_1 x,$$

as shown in figure 9.1. However, for smooth gentle curves, program **polnom** (page 42) is preferred because

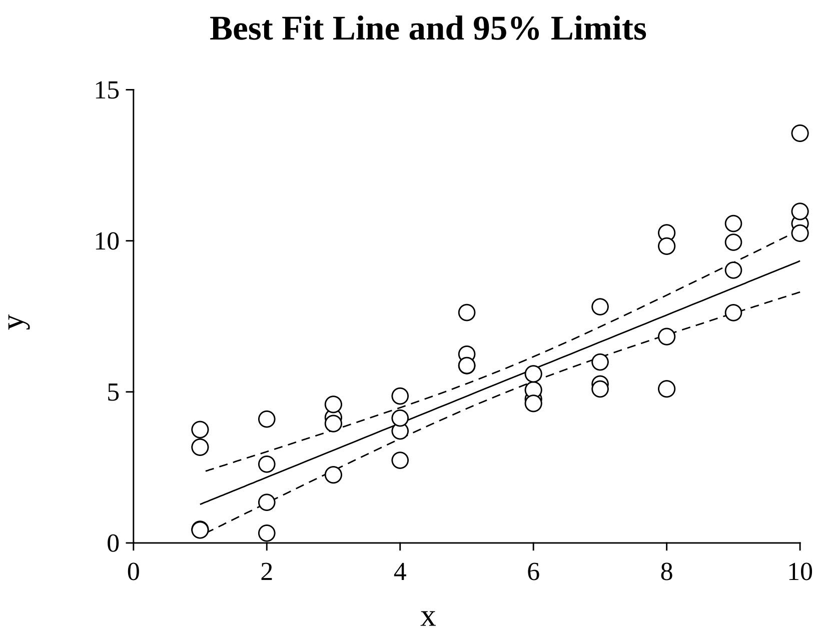


Figure 9.1: A linear calibration curve

it can also fit a polynomial

$$f(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n,$$

where the degree n is chosen according to statistical principles. What happens is that **polnom** fits all polynomials from degree 0 up to degree 6 and gives statistics necessary to choose the statistically justified best fit n. However, in the case of calibration curves, it is not advisable to use a value of n greater than 2 or at most 3, and warnings are issued if the best fit standard curve has any turning points that could make inverse prediction non-unique.

To practise, read test file `line.tf1` for a line, or `polnom.tf1` for a gentle curve, into **polnom** and create a linear calibration curve with confidence envelope as shown in figure 9.1. Now predict x from y values, say for instance using `polnom.tf3`, or the values 2,4,6,8,10.

9.2.3 Calibration using calcurve

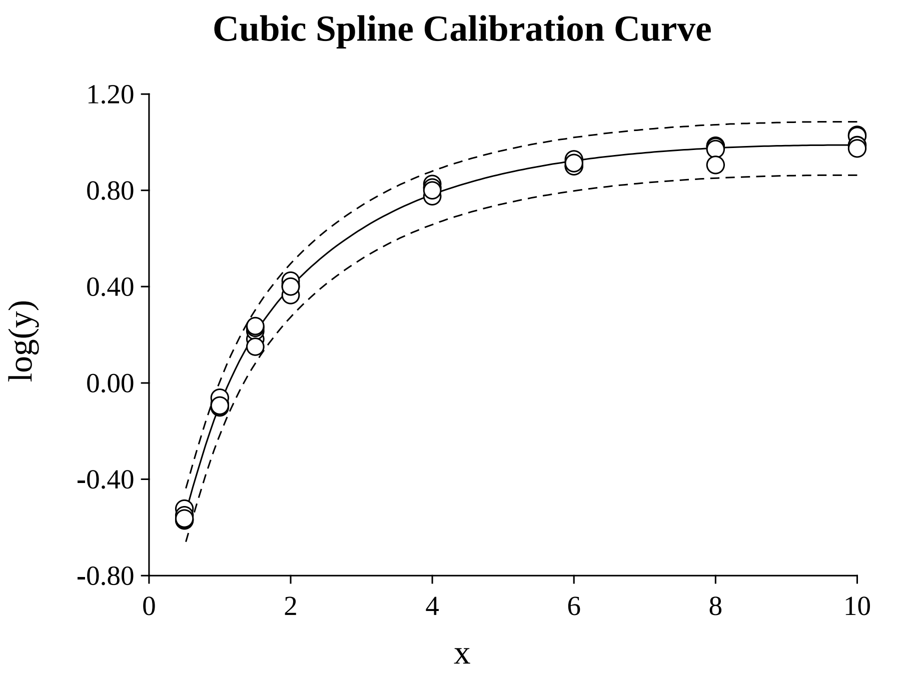


Figure 9.2: A cubic spline calibration curve

If a polynomial of degree 2 or at most 3 is not adequate, a cubic spline calibration curve could be considered. It does not matter how nonlinear your data are, **calcurve** can fit them with splines with user-defined fixed knots as described on page 270. The program has such a vast number of options that a special mode of operation is allowed, called the expert mode, where all decisions as to weighting, spline knots, transformations, etc. are added to the data file. The advantage of this is that, once a standard curve has been created, it can be reproduced exactly by reading in the standard curve data file.

To practise, read in `calcurve.tf1` and use expert mode to get figure 9.2. Now do inverse prediction using `calcurve.tf3` and browse `calcurve.tf1` to understand expert mode.

9.2.4 Calibration using qnfit

Sometimes you would want to use a specific mathematical model for calibration. For instance, a mixture of two High/Low affinity binding sites or a cooperative binding model might be required for a saturation curve, or a mixture of two logistics might adequately fit growth data. If you know an appropriate model for the standard curve, use **qnfit** for inverse prediction because, after fitting, the best-fit curve can be used for calibration, or for estimating derivatives or areas under curves AUC if appropriate.

9.3 Dose response curves, EC50, IC50, ED50, and LD50

A special type of inverse prediction is required when equations are fitted to dose response data in order to estimate some characteristic parameter, such as the half time $t_{1/2}$, the area under the curve AUC, or median effective dose in bioassay (e.g. $ED50$, $EC50$, $IC50$, $LD50$, etc.), along with standard errors and 95% confidence limits. The model equations used in this sort of analysis are not supposed to be exact models constructed according to scientific laws, rather they are empirical equations, selected to have a shape that is close to the shape expected of such data sets. So, while it is is pedantic to insist on using a model based on scientific model building, it is important to select a model that fits closely over a wide variety of conditions.

Older techniques, such as using data subjected to a logarithmic transform in order to fit a linear model, are no longer called for as they are very unreliable, leading to biased parameter estimates. Hence, in what follows, it is assumed that data are to be analyzed in standard, not logarithmically transformed coordinates, but there is nothing to prevent data being plotted in transformed space after analysis, as is frequently done when the independent variable is a concentration, i.e., it is desired to have an the independent variable proportional to chemical potential. The type of analysis called for depends very much on the nature of the data, the error distribution involved, and the goodness of fit of the assumed model. It is essential that data are obtained over a wide range, and that the best fit curves are plotted and seen to be free from bias which could seriously degrade routine estimates of percentiles, say. The only way to decide which of the following procedures should be selected for your data, is to analyze the data using those candidate models that are possibilities, and then to adopt the model that seems to perform best, i.e., gives the closest best fit curves and most sensible inverse predictions.

❏ **Exponential models**
If the data are in the form of a simple or multiphasic exponential decline from a finite value at $t = 0$ to zero as $t \longrightarrow \infty$, and half times $t_{1/2}$, or areas AUC are required, use **exfit** (page 60) to fit one or a sum of two exponentials with no constant term. Practise with **exfit** and test file `exfit.tf4`. With the simple model

$$f(t) = A\exp(-kt)$$

of order 1, then the $AUC = A/k$ and $t_{1/2} = -\log(2)/k$ are given explicitly but, if this model does not fit and a higher model has to be used, then the corresponding parameters will be estimated numerically.

❏ **Trapezoidal estimation**
If no deterministic model can be used for the AUC it is usual to prefer the trapezoidal method with no data smoothing, where replicates are simply replaced by means values that are then joined up sequentially by sectional straight lines. The program **average** (page 269) is well suited to this sort of analysis.

❏ **The Hill equation**
This empirical equation is

$$f(x) = \frac{Ax^n}{B^n + x^n},$$

which can be fitted using program **inrate** (page 264), with either n estimated or n fixed, and it is often used in sigmoidal form (i.e. $n > 1$) to estimate the maximum value A and half saturation point B, with sigmoidal data (not data that are only sigmoidal when x-semilog transformed, as all binding isotherms are sigmoidal in x-semilog space).

- **Ligand binding and enzyme kinetic models.**
  There are three cases:
  a) data are increasing as a function of an effector, i.e., ligand or substrate, and the median effective ligand concentration $ED50$ or apparent $K_m = EC50 = ED50$ is required,
  b) data are a decreasing function of an inhibitor $[I]$ at fixed substrate concentration $[S]$ and $IC50$, the concentration of inhibitor giving half maximal inhibition, is required, or
  c) the flux of labeled substrate $[Hot]$, say, is measured as a decreasing function of unlabeled isotope $[Cold]$, say, with $[Hot]$ held fixed.
  If the data are for an increasing saturation curve and ligand binding models are required, then **hlfit** (page 65) or, if cooperative effects are present, **sffit** (page 66) can be used to fit one or two binding site models. Practise with **sffit** and `sffit.tf4`.

  More often, however, an enzyme kinetic model, such as the Michaelis-Menten equation will be used as now described. To estimate the maximum rate and apparent K_m, i.e., $EC50$ the equation fitted by **mmfit** in substrate mode would be

$$v([S]) = \frac{V_{max}[S]}{K_m + [S]}$$

  while the interpretation of $IC50$ for a reversible inhibitor at concentration $[i]$ with substrate fixed at concentration S would depend on the model assumed as follows.

$$\begin{aligned}
\text{Competitive inhibition } v([I]) &= \frac{V_{max}[S]}{K_m(1 + I/K_i) + [S]}\\
IC50 &= \frac{K_i(K_m + [S])}{K_m}\\
\text{Uncompetitive inhibition } v([I]) &= \frac{V_{max}[S]}{K_m + [S](1 + [I]/K_i]}\\
IC50 &= \frac{K_i(K_m + [S])}{[S]}\\
\text{Noncompetitive inhibition } v([I]) &= \frac{V_{max}[S]}{(1 + [I]/K_i)(K_m + [S])}\\
IC50 &= K_i\\
\text{Mixed inhibition } v([I]) &= \frac{V_{max}[S]}{K(1 + [I]/K_{i1}) + [S](1 + [I]/K_{i2})}\\
IC50 &= \frac{K_{i1}K_{i2}(K_m + [S])}{(K_m K_{i2} + [S]K_{i1})}\\
\text{Isotope displacement } v([Cold]) &= \frac{V_{max}[Hot]}{K_m + [Hot] + [Cold]}\\
IC50 &= K_m + [Hot]
\end{aligned}$$

  Of course, only two independent parameters can be estimated with these models, and, if higher order models are required and justified by statistics and graphical deconvolution, the apparent V_{max} and apparent K_m are then estimated numerically.

- **Growth curves.**
  If the data are in the form of sigmoidal increase, and maximum size, maximum growth rate, minimum growth rate, $t_{1/2}$ time to half maximum size, etc. are required, then use **gcfit** in growth curve mode 1 (page 74). Practise with test file `gcfit.tf2` to see how a best-fit model is selected. For instance, with the logistic model

$$\begin{aligned}
f(t) &= \frac{A}{1 + B\exp(-kt)}\\
t_{1/2} &= \frac{\log(B)}{k}
\end{aligned}$$

the maximum size A and time to reach half maximal size $t_{/2}$ are estimated.

❏ **Survival curves.**
If the data are independent estimates of fractions remaining as a function of time or some effector, i.e. sigmoidally decreasing profiles fitted by **gcfit** in mode 2, and $t_{1/2}$ is required, then normalize the data to proportions of time zero values and use **gcfit** in survival curve mode 2 (page 77). Practise with `Weibull.tf1`, which has the model equation

$$S(t) = 1 - \exp(-(At)^B)$$
$$t_{1/2} = \frac{\log(2)}{AB}.$$

❏ **Survival time models.**
If the data are in the form of times to failure, possibly censored, then **gcfit** should be used in survival time mode 3 (page 253). Practise with test file `survive.tf2`. With the previous survival curve and with survival time models the median survival time $t_{1/2}$ is estimated, where

$$\int_0^{t_{1/2}} f_T(t)\, dt = \frac{1}{2},$$

and $f_T(t)$ is the survival probability density function.

❏ **Models for proportions.**
If the data are in the form of numbers of successes (or failures) in groups of known size as a function of some control variable and you wish to estimate percentiles, e.g., $EC50$, $IC50$, or maybe $LD50$ (the median dose for survival in toxicity tests), use **gcfit** in GLM dose response mode. This is because the error distribution is binomial, so generalized linear models, as discussed on page 49, should be used. You should practise fitting the test file `ld50.tf1` with the logistic, probit and log-log models, observing the goodness of fit options and the ability to change the percentile level interactively. An example of how to use this technique follows.

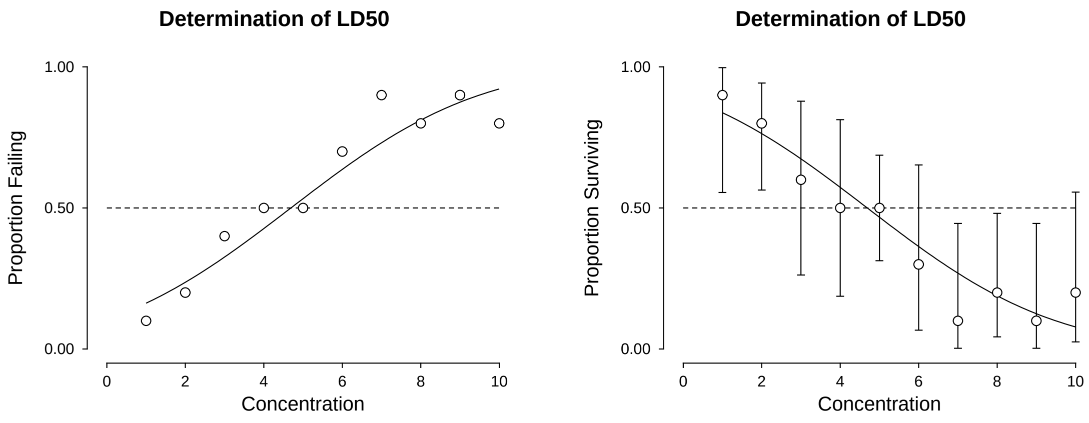


Figure 9.3: Plotting LD50 data with error bars

Figure 9.3 illustrates the determination of $LD50$ using GLM. The left hand figure shows the results from using the probit model to determine $LD50$ using test file `ld50.tf2`. The right hand figure shows exactly the same analysis but carried out using the proportion surviving, i.e., the complement of the numbers in test file `ld50.tf2`, replacing y, the number failing (dying) in a sample of size N, by $N - y$, the number succeeding (surviving) in a sample of size N. Of course the value of the $LD50$ estimate and the associated standard error are identical for both data sets. Note that, in GLM analysis, the percentile can be changed interactively, e.g., if you need to estimate $LD25$ or $LD75$, etc.

The point about about using the analysis of proportions routines in this way for the error bars in the right hand figure is that exact, unsymmetrical 95% confidence limits can be generated from the sample sizes and numbers of successes in this way.

9.4 95% confidence regions in inverse prediction

polnom estimates non-symmetrical confidence limits assuming that the N values of y for inverse prediction and weights supplied for weighting are exact, and that the model fitted has n parameters that are justified statistically. **calcurve** uses the weights supplied, or the estimated coefficient of variation, to fit confidence envelope splines either side of the best fit spline, by employing an empirical technique developed by simulation studies. Root finding is employed to locate the intersection of the y_i supplied with the envelopes. The AUC, LD50, half-saturation, asymptote and other inverse predictions in SimFIT use a t distribution with $N - n$ degrees of freedom, and the variance-covariance matrix estimated from the regression. That is, assuming a prediction parameter defined by $p = f(\theta_1, \theta_2, \ldots, \theta_n)$, a central 95% confidence region is constructed using the prediction parameter variance estimated by the propagation of errors formula

$$\hat{V}(p) = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial \theta_i}\right)^2 \hat{V}(\theta_i) + 2\sum_{i=2}^{n}\sum_{j=1}^{i-1} \frac{\partial f}{\partial \theta_i}\frac{\partial f}{\partial \theta_j}\hat{CV}(\theta_i, \theta_j).$$

Note that this formula for the propagation of errors can be used to calculate parameter standard errors for parameters that are calculated as functions of parameters that have been estimated by fitting, such as apparent maximal velocity when fitting sums of Michaelis-Menten functions. However, such estimated standard errors will only be very approximate.

9.5 Partial Least Squares (PLS)

This technique is also known as regression by projection to latent structures, and it is sometimes useful when a n by r matrix of responses Y, with $r \geq 1$, is observed with a n by m matrix of predictor variables X, with $m > 1$, and one or more of the following conditions may apply:

- There is no deterministic model to express the r columns of Y as functions of the m columns of the matrix X.
- The number of columns of X is too large for convenient analysis, or the number of observations n is not significantly greater than the number of predictor variables m.
- The X variables may be correlated and/or the Y variables may be correlated.

The idea behind PLS is to express the X and Y matrices in terms of sets of k factors, with $k \leq m$, derived from the matrices by projection and regression techniques. The X scores would have maximum covariance with the Y scores, and the principal problem is to decide on a sufficiently small dimension l, with $l \leq k$, that would be needed to represent the relationship between Y and X adequately. Having obtained satisfactory expressions for approximating X and Y using these factors, they can then be used to treat X as a training matrix, then predict what new Y would result from a new n by m matrix Z that is expressed in the same variables as the training matrix X. Hence the use of this technique in multivariate calibration, or quantitative structure activity relationships (QSAR).

If X_1 is the centered matrix obtained from X by subtracting the X column means, and Y_1 is obtained from Y by subtracting the Y column means, then the first factor is obtained by regressing on a column vector of n normalized scores t_1, as in

$$\begin{aligned} \hat{X}_1 &= t_1 p_1^T \\ \hat{Y}_1 &= t_1 c_1^T \\ t_1^T t_1 &= 1, \end{aligned}$$

where the column vectors of m x-loadings p_1 and r y-loadings c_1 are calculated by least squares, i.e.

$$\begin{aligned} p_1^T &= t_1^T X_1 \\ c_1^T &= t_1^T Y_1. \end{aligned}$$

The x-score vector $t_1 = X_1 w_1$ is the linear combination of X_1 that has maximum covariance with the y-scores $u_1 = Y_1 c_1$, where the x-weights vector w_1 is the normalized first left singular vector of $X_1^T Y_1$. The further $k - 1$ orthogonal factors are then calculated successively using

$$\begin{aligned} X_i &= X_{i-1} - \hat{X}_{i-1} \\ Y_i &= Y_{i-1} - \hat{Y}_{i-1},\ i = 2, 3, \ldots, k \\ t_i^T t_j &= 0,\ \ j = 1, 2, \ldots, i - 1. \end{aligned}$$

Once a set of k factors has been calculated, these can be used to generate the parameter estimates necessary to predict a new Y matrix from a Z matrix, given the original training matrix X. Usually k would be an upper limit on the number of factors to consider, and the m by r parameter estimates matrix B required for l factors, where $l \leq k$, would be given by

$$B = W(P^T W)^{-1} C^T.$$

Here W is the m by k matrix of x-weights, P is the m by k matrix of x-loadings, and C is the r by k matrix of y-loadings. Note that B calculated in this way is for the centered matrices X_1 and Y_1, but parameter estimates appropriate for the original data are also calculated.

Before proceeding to discuss a worked example, it is important to emphasize a complication which can arise when predicting a new Y matrix using the parameter estimates. In most multivariate techniques it is immaterial whether the data are scaled and centered before submitting a sample for analysis, or whether the data are scaled and centered internally by the software. In the case of PLS, the Y predicted will be incorrect if the data are centered and scaled independently before analysis, but then the Z matrix for prediction is centered and scaled using its own column means and variances.

So there are just two ways to make sure PLS predicts correctly.

1. You can submit X and Y matrices that are already centered and scaled, but then you must submit a Z matrix that has not been centered and scaled using its own column means and standard deviations, but one that has been processed by subtracting the original X column means and scaled using the original X column standard deviations.
2. Do not center or scale any data. Just submit the original data for analysis, request automatic centering and scaling if necessary, but allow the software to then center and scale internally.

As the first method is error prone and will predict scaled and centered predictions, which could be confusing, the advice to PLS users would be:

Do not center or scale any training sets, or Z-data for predicting new Y, before PLS analysis.
Always submit raw data and allow the software to perform centering and scaling.
That way predictions will be in coordinates corresponding to the original Y-coordinates.

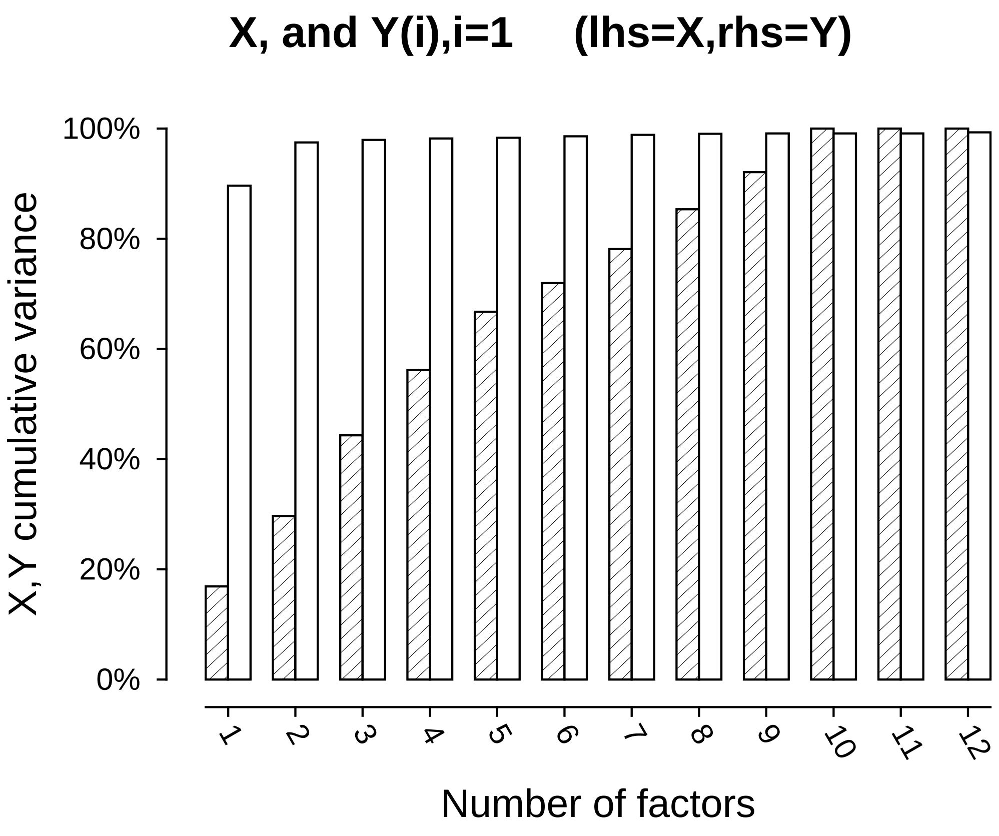


Figure 9.4: PLS: selecting l factors from k=12 by using the variance explained in X and Y

Several techniques are available to decide how many factors l out of the maximum calculated k should be selected when using a training set for prediction. For instance, figure 9.4 was obtained by using test file `g02laf.tf1` with 15 rows and 15 columns as the source of X prediction data, and test file `g02laf.tf2` with

15 rows and just 1 column as the source of Y response data, then fitting a PLS model with up to a maximum of $k = 12$ factors. It illustrates how the cumulative percentage of variance in X and a column of Y is accounted for the factor model as the number of factors is steadily increased. It is clear that two factors are sufficient to account for the variance of the single column of Y in this case but more, probably about 6, are required to account for the variance in the X matrix, i.e. we should choose $l = 6$.

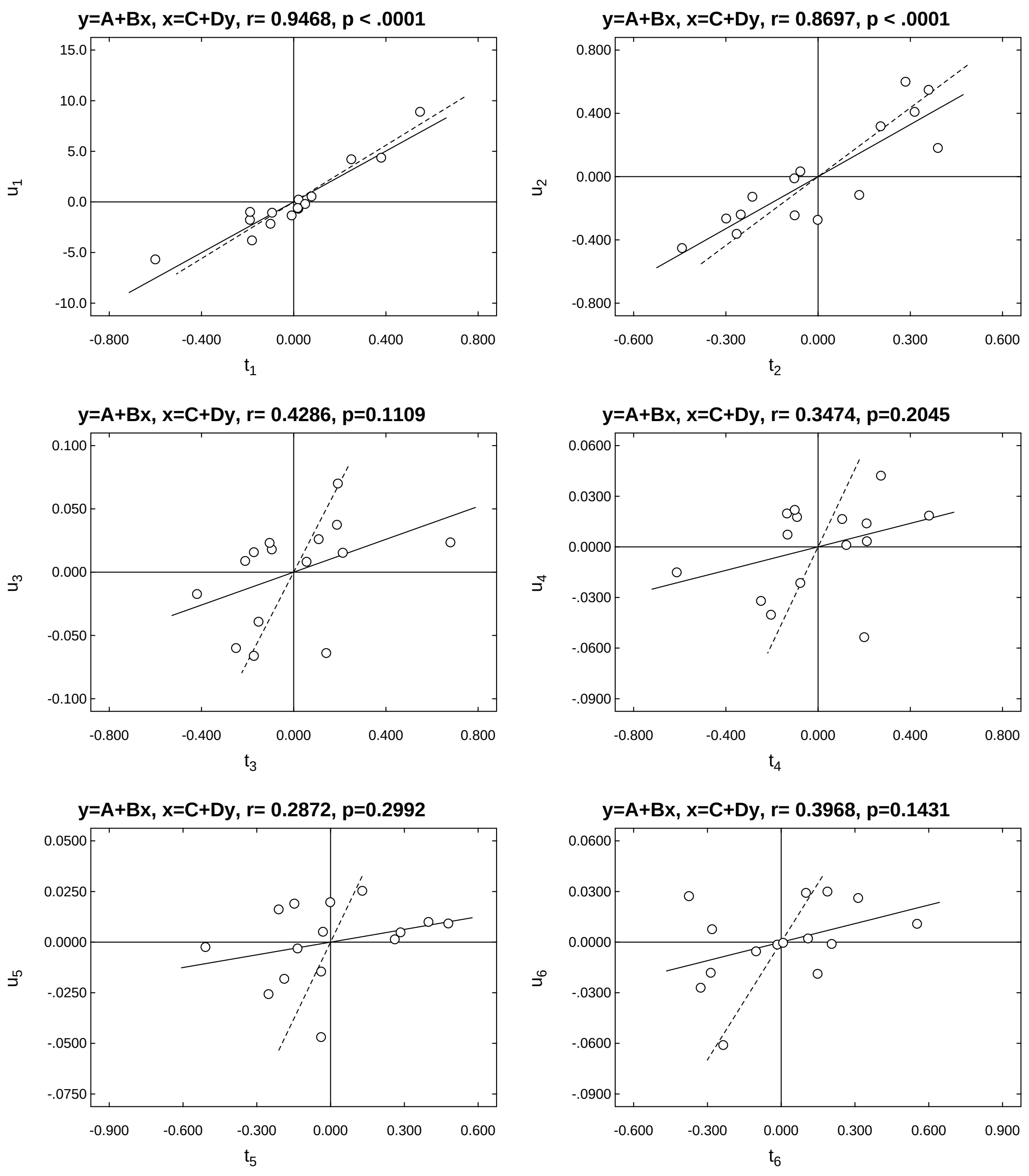


Figure 9.5: PLS correlation between scores

Alternatively, figure 9.5 plots the successive correlations between the X and Y scores. Each plot shows the best fit linear regression for the u_i i.e. Y scores on the t_i i.e. X scores, and also the best fit linear regression of the X scores on the Y scores, as explained elsewhere (page 190), together with the correlation coefficients r and and significance levels p. Clearly the scores corresponding to the first two factors are highly correlated, but thereafter the correlation is very weak.

Note that the PLS model can be summarized as follows

$$\begin{aligned} X &= \bar{X} + TP^T + E \\ Y &= \bar{Y} + UC^T + F \\ U &= T + H \end{aligned}$$

where E, F, and H are matrices of residuals. So the SIMFIT PLS routines also allow users to study such residuals, to see how closely the fitted model predicts the original Y data for increasing numbers of factors before the number of factors to be used routinely is decided. Various tests for goodness of fit can be derived from these residuals and, in addition, variable influence on projection (VIP) statistics can also be calculated.

Table 9.1 gives the VIP values for fitting a PLS model to the X data in test file `g02laf.tf1`, which are available after parameters have been calculated.

```
Variable     VIP
    1      0.611072
    2      0.318216
    3      0.751272
    4       0.50482
    5      0.271225
    6      0.359276
    7       1.57773  *
    8       2.43477  *
    9        1.1322  *
   10       1.22255  *
   11       1.17993  *
   12      0.883977
   13      0.212875
   14      0.212875
   15      0.212875
```

Table 9.1: PLS: variables influence on projection

Now the sum of squared VIP values equals the number of X variables, and a large VIP(i) value indicates that X variable i has an important influence on projection. So, we see that only variables 7, 8, 9, 10, and 11 have VIP values greater than 1, and these are therefore the more important predictor variables in the X predictor matrix.

Part 10

Statistical analysis

10.1 Introduction

The main part of the SimFIT statistics functions are to be found in the program **simstat**, which is in many ways like a small scale statistics package. This provides options for data exploration, statistical tests, analysis of variance, multivariate analysis, regression, time series, power calculations, etc., as well as a number of calculations, like finding zeros of polynomials, or values of determinants, inverses, eigenvalues or eigenvalues of matrices. In addition to **simstat** there are also several specialized programs that can be used for more detailed work and to obtain information about dedicated statistical distributions and related tests but, before describing the **simstat** procedures with worked examples, a few comments about tests may be helpful.

10.1.1 Statistical tests

A test statistic is a function evaluated on a data set, and the significance level of a test is the probability of obtaining a test statistic as extreme, or more extreme, from a random sample, given a null hypothesis H_0, which usually specifies a distribution for that test statistic. If the error rate, i.e. significance level p is less than some critical level, say $\alpha = 0.05$ or $\alpha = 0.01$, it is reasonable to consider whether the null hypothesis should be rejected. The correct procedure is to choose a test, decide whether to use the upper, lower, or two-tail test statistic as appropriate, select the critical significance level, do the test, then accept the outcome. What is not valid is to try several tests until you find one that gives you the result you want. That is because the probability of a Type 1 error increases monotonically as the number of tests increases, particularly if the tests are on the same data set, or some subsets of a larger data set. This multiple testing should never be done, but everybody seems to do it. Of course, all bets are off anyway if the sample does not conform to the assumptions implied by H_0, for instance, doing a t test with two samples that are known not to be normally distributed with the same variance.

10.1.2 Multiple tests

Statistical packages are designed to be used in the rather pedantic but correct manner just described, which makes them rather inconvenient for data exploration. SimFIT, on the other hand, is biased towards data exploration, so that various types of multiple testing can be done. However, once the phase of data exploration is completed, there is nothing to stop you making the necessary decisions and only using the subset of results calculated by the SimFIT statistical programs, as in the classical (correct) manner. Take, for example, the t test. SimFIT does a test for normality and variance equality on the two samples supplied, it reports lower, upper and two tail test statistics and p values simultaneously, it performs a corrected test for the case of unequal variances at the same time, it allows you to follow the t test by a paired t test if the sample sizes are equal and, after doing the t test, it saves the data for a Mann-Whitney U or Kolmogorov-Smirnov 2-sample test on request. An even more extreme example is the all possible pairwise comparisons option, which does all possible t, Mann-Whitney U and Kolmogorov-Smirnov 2-sample tests on a library file of column vectors.

In fact there are two ways to view this type of multiple testing. If you are just doing data exploration to identify possible differences between samples, you can just regard the p values as a measure of the differences between pairs of samples, in that small p values indicate samples which seem to have different distributions. In this case you would attach no importance as to whether the p values are less than any supposed critical α values. On the other hand, if you are trying to identify samples that differ significantly, then some technique is required to structure the multiple testing procedure and/or alter the significance level, as in the Tukey Q test. If the experimentwise error rate is α_e while the comparisonwise error rate is α_c and there are k comparisons then, from equating the probability of k tests with no Type 1 errors it follows that

$$1 - \alpha_e = (1 - \alpha_c)^k.$$

This is known as the Dunn-Sidak correction, but, alternatively, the Bonferroni correction is based on the recommendation that, for k tests, the error rate should be decreased from α to α/k, which gives a similar value to use for α_c in the multiple test, given α_e.

10.2 Data exploration

SimFIT has a number of techniques that are appropriate for exploration of data and data mining. Such techniques do not always lead to meaningful hypothesis tests, but are best used for preliminary investigation of data sets prior to more specific model building.

10.2.1 Exhaustive analysis: arbitrary vector

This procedure is used when you have a single sample (column vector) and wish to explore the overall statistical properties of the data. For example, read in the vector test file `normal.tf1` and you will see that all the usual summary statistics are calculated as in Table 10.1, including the range, hinges (i.e. quartiles), mean

```
Data: 50 numbers from a normal distribution mu = 0 and sigma = 1
Sample size                    50
Minimum, Maximum values      -2.208,  1.617
Lower and Upper Hinges       -0.855,  0.786
Coefficient of skewness      -001.669
Coefficient of kurtosis      -0.70684
Median value                 -0.09736
Sample mean                  -0.02579
Sample standard deviation     1.006: CV% > 100%
Standard error of the mean    0.1422
Upper 2.5% t-value            2.01
Lower 95% con lim for mean   -0.3116
Upper 95% con lim for mean    0.26
Variance of the sample        1.011
Lower 95% con lim for var.    0.7055
Upper 95% con lim for var.    1.570
Shapiro-Wilks W statistic     0.9627
Significance level for W      0.1153     Tentatively accept normality
```

Table 10.1: Exhaustive analysis of an arbitrary vector

$\bar{x}$, standard deviation s, and the normalized sample moments s_3 (coefficient of skewness), and s_4 (coefficient of kurtosis), defined in a sample of size n by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$s_3 = \frac{n}{(n-1)(n-2)} \frac{\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}$$

$$s_4 = \frac{(n+1)n}{(n-1)(n-2)(n-3)} \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{s^4} - \frac{3(n-1)^2}{(n-2)(n-3)}.$$

You can then do a Shapiro-Wilks test for normality (which will, of course, not always be appropriate) or create a histogram, pie chart, cumulative distribution plot or appropriate curve-fitting files. This option is a very valuable way to explore any single sample before considering other tests. If you created files `vector.1st` and `vector.2nd` as recommended earlier you can now examine these. Note that once a sample has been read into program **simstat** it is saved as the current sample for editing, transforming or re-testing. Since vectors have only one coordinate, graphical display requires a further coordinate. In the case of histograms the extra coordinate is provided by the choice of bins, which dictates the shape, but in the case of cumulative distributions it is automatically created as steps and therefore of unique shape. Pie chart segments are calculated in proportion to the sample values, which means that this is only appropriate for positive samples, e.g., counts. The other techniques illustrated in figure 10.1 require further explanation. If the sample values

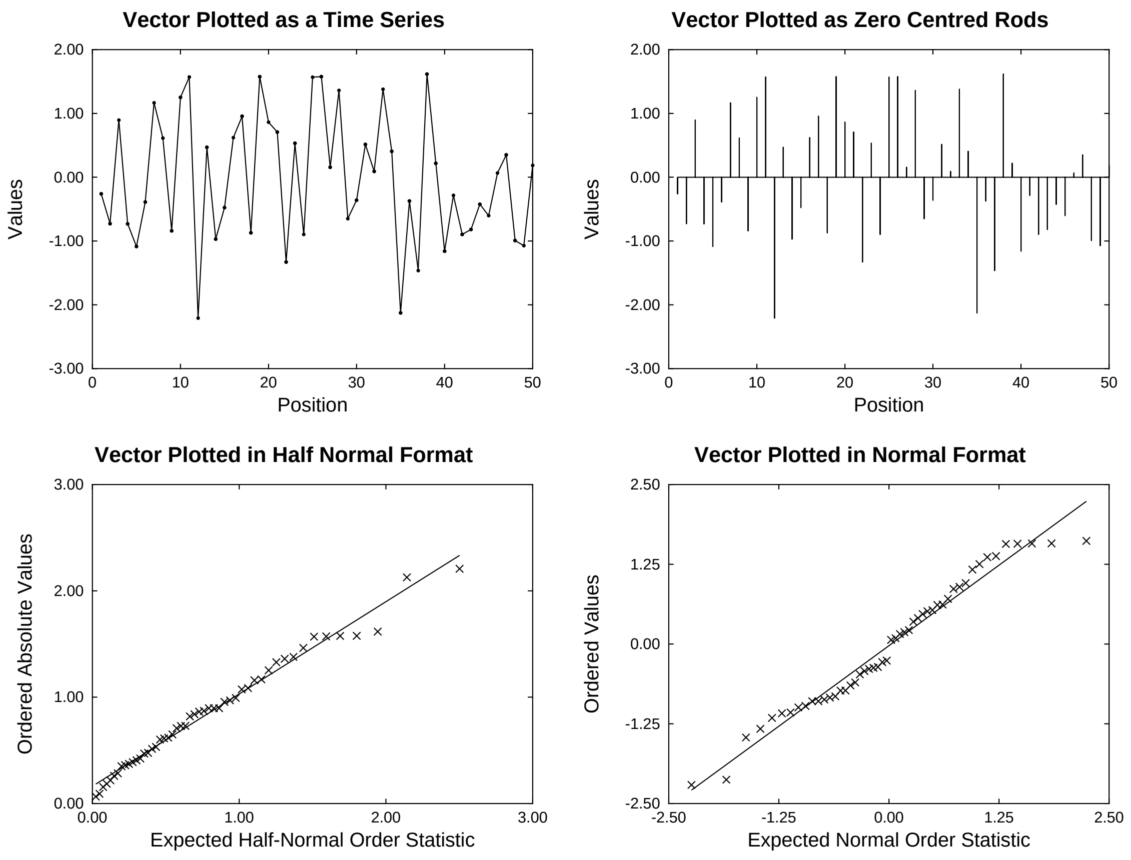


Figure 10.1: Plotting vectors

have been measured in some sequence of time or space, then the y values could be the sample values while the x values would be successive integers, as in the time series plot. Sometimes it is useful to see the variation in the sample with respect to some fixed reference value, as in the zero centered rods plot. The data can be centered automatically about zero by subtracting the sample mean if this is required. The half normal and normal plots are particularly useful when testing for a normal distribution with residuals, which should be approximately normally distributed if the correct model is fitted. In the half normal plot, the absolute values of a sample of size n are first ordered then plotted as $y_i, i = 1, \ldots, n$, while the half normal order statistics are approximated by

$$x_i = \Phi^{-1}\left(\frac{n + i + \frac{1}{2}}{2n + \frac{9}{8}}\right), \ i = 1, \ldots, n$$

which is valuable for detecting outliers in regression. The normal scores plot simply uses the ordered sample as y and the normal order statistics are approximated by

$$x_i = \Phi^{-1}\left(\frac{i - \frac{3}{8}}{n + \frac{1}{4}}\right), \ i = 1, \ldots, n$$

which makes it easy to visualize departures from normality. Best fit lines, correlation coefficients, and significance values are also calculated for half normal and normal plots. Note that a more accurate calculation for expected values of normal order statistics is employed when the Shapiro-Wilks test for normality (page 133) is used and a normal scores plot is required.

10.2.2 Exhaustive analysis: arbitrary matrix

This procedure is provided for when you have recorded several variables (columns) with multiple cases (rows) and therefore have data in the form of a rectangular matrix, as with Table 10.2 resulting from analyzing `matrix.tf2`. The option is used when you want summary statistics for a numerical matrix with no missing

```
Data: Matrix of order 7 by 5
  Row       Mean   Variance    St.Dev.     Coeff.Var.
    1      3.680      8.197      2.863         77.80%
    2      6.804     9.5905     3.0969         45.52%
    3      6.246     3.5253     1.8776         30.06%
    4      4.546     7.1105     2.6666         58.66%
    5      5.784     5.7305     2.3939         41.39%
    6      4.822     6.7613     2.6003         53.92%
    7      5.940     1.7436     1.3205         22.23%
```

Table 10.2: Exhaustive analysis of an arbitrary matrix

values. It analyzes every row and column in the matrix then, on request, exhaustive analysis of any chosen row or column can be performed, as in exhaustive analysis of a vector.

Often the rows or columns of a data matrix have pairwise meaning. For instance, two columns may be measurements from two populations where it is of interest if the populations have the same means. If the populations are normally distributed with the same variance, then an unpaired t test might be appropriate (page 136), otherwise the corresponding nonparametric test (Mann-Whitney U, page 140), or possibly a Kolmogorov-Smirnov 2-sample test (page 139) might be better. Again, two columns might be paired, as with measurements before and after treatment on the same subjects. Here, if normality of differences is reasonable, a paired t test (page 138) might be called for, otherwise the corresponding nonparametric procedure (Wilcoxon signed rank test, page 142), or possibly a run test (page 153), or a sign test (page 152) might be useful for testing for absence of treatment effect. Table 10.3 illustrates the option to do statistics on paired rows or columns, in this case columns 1 and 2 of `matrix.tf2`. You identify two rows or columns from the matrix then simple plots, linear regression, correlation, and chosen statistical tests can be done. Note that all the p values calculated for this procedure are for two-tail tests, while the run, Wilcoxon sign rank, and sign test

```
Unpaired t test:
t = -3.094
p =  0.0093    *p =< 0.01
Paired t test:
t = -3.978
p =  0.0073    *p =< 0.01
Kolmogorov-Smirnov 2-sample test:
d =  0.7143
z =  0.3818
p =  0.0082    *p =< 0.01
Mann-Whitney U test:
u =  7.0
z = -2.172
p =  0.0262    *p =< 0.05
Wilcoxon signed rank test:
w =  1.0
z = -2.113
p =  0.0313    *p =< 0.05
Run test:
+ =  1 (number of x > y)
- =  6 (number of x < y)
p =  0.2857
Sign test:
N =  7 (non-tied pairs)
- =  6 (number of x < y)
p =  0.1250
```

Table 10.3: Statistics on paired columns of a matrix

ignore values which are identical in the two columns. More detailed tests can be done on the selected column vectors by the comprehensive statistical test options to be discussed subsequently (page 130).

The comprehensive analysis of a matrix procedure also allows for the data matrix to be plotted as a 2-dimensional bar chart, assuming that the rows are cases and the columns are numbers in distinct categories, or as a 3-dimensional bar chart assuming that all cell entries are as in a contingency table, or similar. Alternatively, plots displaying the columns as scattergrams, box and whisker plots, or bar charts with error bars can be constructed.

10.2.3 Exhaustive analysis: multivariate normal matrix

This provides options that are useful before proceeding to more specific techniques that depend on multivariate normality (page 421), e.g., MANOVA and some types of ANOVA.

A graphical technique is provided for investigating if a data matrix with n rows and m columns, where $n >> m > 1$, is consistent with a multivariate normal distribution. For example, figure 10.2 shows plots for two random samples from a multivariate normal distribution. The plot uses the fact that, for a multivariate normal distribution with sample mean $\bar{x}$ and sample covariance matrix S,

$$(x - \bar{x})^T S^{-1} (x - \bar{x}) \sim \frac{m(n^2 - 1)}{n(n - m)} F_{m,n-m},$$

where x is a further independent observation from this population, so that the transforms plotted against the quantiles of an F distribution with m and $n - m$ degrees of freedom, i.e. according to the cumulative

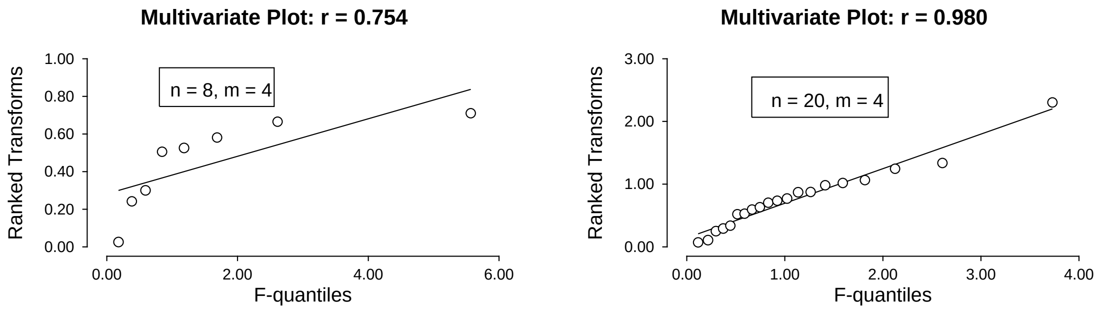


Figure 10.2: Plot to diagnose multivariate normality

probabilities for $(i - 0.5)/n$ for $i = 1, 2, \ldots, n$ should be a straight line. It can be seen from figure 10.2 that this plot is of little value for small values of n, say $n \approx 2m$ but becomes progressively more useful as the sample size increases, say $n > 5m$.

Again, there are procedures to calculate the column means $\bar{x}_j$, and m by m sample covariance matrix S, defined for a n by m data matrix x_{ij} with $n \geq 2, m \geq 2$ as

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$
$$s_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

and then exploit several techniques which use these estimates for the population mean vector and covariance matrix. The eigenvalues and determinants of the sample covariance matrix and its inverse are required for several MANOVA techniques, so these can also be estimated. It is possible to perform two variants of the Hotelling T^2 test, namely

- testing for equality of the mean vector with a specified reference vector of means, or
- testing for equality of all means without specifying a reference mean.

Dealing first with testing that a vector of sample means is consistent with a reference vector, table 10.4 resulted

```
Hotelling one sample T-square test

H0: Delta = (Mean - Expected) are all zero

Number of rows = 10, Number of columns = 4
Hotelling T-square          = 7.439
F Statistic (FTS)           = 1.240
Degrees of Freedom (d1,d2)  = 4, 6
p = P(F(d1,d2) >= FTS)      =  0.3869
Column     Mean   Std.Err. Expected  Delta       t     p
   1      -0.53     0.463       0.0  -0.53   -1.15  0.2815
   2      -0.03     0.386       0.0  -0.03 -0.0778  0.9397
   3      -0.59     0.491       0.0  -0.59    -1.2  0.2601
   4        3.1      1.95       0.0    3.1    1.59  0.1457
```

Table 10.4: Hotelling T^2 test for H_0: means = reference

when the test file hotel.tf1 was analyzed using the Hotelling one sample test procedure. This tests the null hypothesis $H_0 : \mu = \mu_0$ against the alternative $H_1 : \mu \neq \mu_0$, where μ_0 is a known mean vector and no assumptions are made about the covariance matrix Σ. Hotelling's T^2 is

$$T^2 = n(\bar{x} - \mu_0)^T S^{-1}(\bar{x} - \mu_0)$$

and, if H_0 is true, then an F test can be used since $(n - m)T^2/(m(n - 1))$ is distributed asymptotically as $F_{m,n-m}$. Users can input any reference mean vector μ_0 to test for equality of means but, when the data columns are all differences between two observations for the same subjects and the aim is to test for no significant differences, so that μ_0 is the zero vector, as with hotel.tf1, the test is a sort of higher dimensional analogue of the paired t test. Table 10.4 also shows the results when t tests are applied to the individual columns of differences between the sample means $\bar{x}$ and the reference means μ_0, which is suspect because of multiple testing but, in this case, the conclusion is the same as the Hotelling T^2 test: none of the column means are significantly different from zero.

Now, turning to a test that all means are equal, table 10.5 shows the results when the data in anova6.tf1 are

```
Hotelling one sample T-square test

H0: Column means are all equal

Number of rows = 5, Number of columns = 4
Hotelling T-square = 170.5
F Statistic (FTS)  = 28.41
Deg. Free. (d1,d2) = 3, 2
P(F(d1,d2) >= FTS) = 0.0342  Reject H0 at 5% sig.level
```

Table 10.5: Hotelling T^2 test for H_0: means are equal

analyzed, and the theoretical background to this test will be presented subsequently (page 175).

Options are provided for investigating the structure of the covariance matrix. The sample covariance matrix and its inverse can be displayed along with eigenvalues and determinants, and there are also options to check if the covariance matrix has a special form, namely

- testing for compound symmetry,
- testing for spherical symmetry, and
- testing for spherical symmetry of the covariance matrix of orthonormal contrasts.

For instance, using the test file hotel.tf1 produces the results of table 10.6 showing an application of a test for compound symmetry and a test for sphericity. Compound symmetry is when a covariance matrix Σ has a special form with constant nonnegative diagonals and equal nonnegative off-diagonal elements as follows.

$$\Sigma = \sigma^2 \begin{pmatrix} 1 & \rho & \dots & \rho \\ \rho & 1 & \dots & \rho \\ \dots & \dots & \dots & \dots \\ \rho & \rho & \dots & 1 \end{pmatrix}$$

This can be tested using estimates for the diagonal and off-diagonal elements σ^2 and $\sigma^2\rho$ as follows

$$s^2 = \frac{1}{m}\sum_{i=1}^{m} s_{ii}$$

$$s^2 r = \frac{2}{m(m-1)}\sum_{i=2}^{m}\sum_{j=1}^{i-1} s_{ij}.$$

```
Variance-Covariance matrix
  2.1401 -0.11878 -0.89411  3.5922
-0.11871   1.4868  0.79144  1.8811
-0.89411  0.79144   2.4099 -4.6011
  3.5922   1.8811  -4.6011  37.878
Pearson product-moment correlations
 1.0000 -0.0666 -0.3937  0.3990
-0.0666  1.0000  0.4181  0.2507
-0.3937  0.4181  1.0000 -0.4816
 0.3990  0.2507 -0.4816  1.0000

Compound symmetry test

H0: Covariance matrix has compound symmetry

Number of groups        = 1
Number of variables (m) = 4
Sample size (n)         = 10
Determinant of CV       = 98.14
Determinant of S_0      = 1452.0
LRTS (-2*log(lambda))   = 36.3
Degrees of Freedom      = 8
p = P(chi-sq. >= LRTS)  = 0.0000  Reject H0 at 1% sig.level

Likelihood ratio sphericity test

H0: Covariance matrix = k*Identity (for some k > 0)

Number of small eigenvalues = 0 (i.e. < 0.0000001)
Number of variables (m)     = 4
Sample size (n)             = 10
Determinant of CV           = 98.14
Trace of CV                 = 43.91
Mauchly W statistic         = 0.006756
LRTS (-2*log(lambda))       = 49.97
Degrees of Freedom          = 9
p = P(chi-sq. >= LRTS)      = 0.0000  Reject H0 at 1% sig.level
```

Table 10.6: Covariance matrix symmetry and sphericity tests

The Wilks generalized likelihood-ratio statistic is

$$L = \frac{|S|}{(s^2 - s^2 r)^{m-1}[s^2 + (m-1)s^2 r]},$$

where the numerator is the determinant of the covariance matrix estimated with ν degrees of freedom, while the denominator is the determinant of the matrix with average variance on the diagonals and average covariance as off-diagonal elements, and this is used to construct the test statistic

$$\chi^2 = -\left[\nu - \frac{m(m+1)^2(2m-3)}{6(m-1)(m^2+m-4)}\right]\log L$$

which, for large ν, has an approximate chi-squared distribution with $m(m+1)/2 - 2$ degrees of freedom.

The sphericity test, designed to test the null hypothesis $H_0 : \Sigma = kI$ against $H_1 : \Sigma \neq kI$. In other words, the population covariance matrix Σ is a simple multiple of the identity matrix, which is a central requirement for some analytical procedures. If the sample covariance matrix S has eigenvalues α_i for $i = 1, 2, \ldots, m$ then, defining the arithmetic mean A and geometric mean G of these eigenvalues as

$$A = (1/m) \sum_{i=1}^{m} \alpha_i$$
$$G = (\prod_{i=1}^{m} \alpha_i)^{1/m},$$

the likelihood ratio test statistic

$$-2 \log \lambda = nm \log(A/G)$$

is distributed asymptotically as χ^2 with $(m-1)(m+2)/2$ degrees of freedom. Using the fact that the determinant of a covariance matrix is the product of the eigenvalues while the trace is the sum, the Mauchly test statistic W can also be calculated from A and G since

$$\begin{aligned} W &= \frac{|S|}{\{Tr(S)/m\}^m} \\ &= \frac{\prod_{i=1}^{m} \alpha_i}{\{(\sum_{i=1}^{m} \alpha_i)/m\}^m} \\ \text{so that } -2 \log \lambda &= -n \log W. \end{aligned}$$

Clearly, the test rejects the assumption that the covariance matrix is a multiple of the identity matrix in this case, a conclusion which is obvious from inspecting the sample covariance and correlation matrices. Since the calculation of small eigenvalues is very inaccurate when the condition number of the covariance matrix is appreciable, any eigenvalues less than the minimal threshold indicated are treated as equal to that threshold when calculating the test statistic.

10.2.4 t tests on groups across rows of a matrix

Sometimes a matrix with n rows and m columns holds data where groups are defined across rows by membership according to columns, and it is wished to do tests based on groups down through all rows. For instance, test file `ttest.tf6` has 5 rows, but in each row the first 6 columns constitute one group (say X), while the next 7 columns constitute a second group (say Y). At the end of the file there is an extra text section, as follows

```
begin{limits}
 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1
end{limits}
```

where a 1 denotes membership of the X group, and -1 indicates membership of the Y group. A 0 can also be used to indicate groups other than X or Y, i.e. to effectively suppress columns. Table 10.7 shows the results from analyzing `ttest.tf6`. which calculates the sample means, and standard deviations required for a t test.

X_bar	X_std	Y_bar	Y_std	SE_diff	t	p
8.75	0.58224	9.7429	0.81824	0.400913	-2.4765	0.0308
8.2167	1.042	9.1143	1.3006	0.662051	-1.3558	0.2023
17.933	1.8886	9.4714	0.98778	0.816416	10.365	0.0000
-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0000
8.8333	0.82381	18.7	1.6258	0.736044	-13.405	0.0000

Table 10.7: t tests on groups across rows of a matrix

The two-tail p values are displayed, and a -1 is used to signify that a row contains groups with zero variance, so that the test cannot be performed. The minimum sample size for each group is 2 although much larger sample size should be used if possible.

10.2.5 Nonparametric tests across rows of a matrix

If the assumptions of normality and constant variance required by the previous t test are not justified, the same technique can be applied using nonparametric tests. Table 10.8 shows the results from analyzing `ttest.tf6` in this way using the Mann-Whitney U test. This requires larger samples than the previous t test and, if the group size is fairly large, a Kolmogorov-Smirnov 2-sample test can also be done.

MW_U	MW_Z	MW_2-tail_p
7.0	-1.9339	0.9814
11.0	-1.3609	0.9277
42.0	2.9286	0.0006
-1.0	-1.0	-1.0000
0.0	-2.9326	1.0000

Table 10.8: Nonparametric tests across rows

10.2.6 All possible pairwise tests (n vectors or a library file)

This option is used when you have several samples (column vectors) and wish to explore which samples differ significantly. The procedure takes in a library file referencing sets of vector files and then performs any combination of two-tailed t, Kolmogorov-Smirnov 2-sample, and/or Mann-Whitney U tests on all possible pairs. It is usual to select either just t tests for data that you know to be normally distributed, or just Mann-Whitney U tests otherwise. Because the number of tests is large, e.g., $3n(n-1)/2$ for all tests with n samples, be careful not to use it with too many samples.

For example, try it by reading in the library files `anova1.tfl` (or the smaller data set `npcorr.tfl` with three vectors of length 9 where the results are shown in table 10.9) and observing that significant differences are highlighted. This technique can be very useful in preliminary data analysis, for instance to identify potentially rogue columns in analysis of variance, i.e., pairs of columns associated with small p values. However, it is up to you to appreciate when the results are meaningful and to make the necessary adjustment to critical significance levels where the Bonferroni principle is required (due to multiple tests on the same data).

10.3 Tests

10.3.1 1-sample t test

This procedure is used when you have a sample that is known to be normally distributed and wish to test H_0: the mean is μ_0, where μ_0 is a known quantity. Table 10.10 shows the results for such a 1-sample t test on the data in test file `normal.tf1`. The procedure can first do a Shapiro-Wilks test for normality (page 133) if requested and then, for n values of x_i, it calculates the sample mean $\bar{x}$, sample variance s^2, standard error of

```
Mann-Whitney-U/Kolmogorov-Smirnov-D/unpaired-t tests

No. tests = 9, p(1%) = 0.001111, p(5%) = 0.005556  [Bonferroni]
column2.tf1 (data set 1)
column2.tf2 (data set 2)
N1 = 9, N2 = 9, MWU =  8.0     p = 0.00226  *
                KSD =  0.7778  p = 0.00109 **
                  T = -3.716   p = 0.00188  *
column2.tf1 (data set 1)
column2.tf3 (data set 3)
N1 = 9, N2 = 9, MWU =  21.0,  p = 0.08889
                KSD =  0.5556 p = 0.05545
                  T = -2.042  p = 0.05796
column2.tf2 (data set 2)
column2.tf3 (data set 3)
N1 = 9, N2 = 9, MWU = 55.5    p = 0.19589
                KSD = 0.4444  p = 0.20511
                  T = 1.461   p = 0.16350
```

Table 10.9: All possible comparisons

```
Number of x-values             = 50
Number of degrees of freedom   = 49
Theoretical mean (mu_0)        = 0.0
Sample mean (x_bar)            = -0.012579
Std. err. of mean (SE)         = 0.1422
TS = (x_bar - mu_0)/SE         = -0.1814
p = P(t >= TS) (upper tail p) = 0.5716
p = P(t =< TS) (lower tail p) = 0.4284
p for two tailed t test        =  0.8568
Difference D = x_bar - x_mu    = -0.02579
Lower 95% conf. limit for D    = -0.3116
Upper 95% conf. limit for D    = 0.26
Conclusion: Consider accepting equality of means
```

Table 10.10: One sample t test

the mean $s_{\bar{x}}$ and test statistic TS according to

$$\begin{aligned}\bar{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i\\ s^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2\\ s_{\bar{x}} &= \sqrt{s^2/n}\\ TS &= \frac{\bar{x}-\mu_0}{s_{\bar{x}}}\end{aligned}$$

where μ_0 is the supposed theoretical, user-supplied population mean. The significance levels for upper, lower, and two-tailed tests are calculated for a t distribution with $n-1$ degrees of freedom. You can then change μ_0 or select a new data set.

10.3.2 1-sample Kolmogorov-Smirnov test

This nonparametric procedure is used when you have a single sample (column vector) of reasonable size (say greater than 20) and wish to explore if it is consistent with some known distribution, e.g., normal, binomial, Poisson, gamma, etc. The test only works optimally with large samples where the null distribution is a continuous distribution that can be specified exactly and not defined using parameters estimated from the sample. It calculates the maximum positive difference D_n^+, negative difference D_n^-, and overall difference D_n = maximum of D_n^+ and D_n^- between the sample cumulative distribution function $S(x_i)$ and the theoretical *cdf* $F(x_i)$ under H_0, i.e., if frequencies $f(x_i)$ are observed for the variable x_i in a sample of size n, then

$$\begin{aligned}
S(x_i) &= \sum_{j=1}^{i} f(x_j)/n \\
F(x_i) &= P(x \le x_i) \\
D_n^+ &= \max(S(x_i) - F(x_i), 0), i = 1, 2, \dots, n \\
D_n^- &= \max(F(x_i) - S(x_{i-1}), 0), i = 2, 3, \dots, n \\
D_n &= \max(D_n^+, D_n^-).
\end{aligned}$$

The standardized statistics $Z = D\sqrt{n}$ are calculated for the D values appropriate for upper-tail, lower-tail, or two-tailed tests, then the exact significance levels are calculated by SimFIT for small samples by solving $P(D_n \le a/n)$ using the difference equation

$$\sum_{j=0}^{[2a]} (-1)^j ((2a - j)^j / j!) q_{r-j}(a) = 0, r = 2[a] + 1, 2[a] + 2, \dots$$

with initial conditions

$$\begin{aligned}
q_r(a) &= 1, r = 0 \\
&= r^r/r!, r = 1, \dots, [a] \\
&= r^r/r! - 2a \sum_{j=0}^{[r-a]} ((a + j)^{j-1}/j!)(r - a - j)^{r-j}/(r - j)!, r = [a + 1], \dots, 2[a],
\end{aligned}$$

where $[a]$ is the largest integer $\le a$ regardless of the sign of a, while the series

$$\lim_{n \to \infty} P\left(D_n \le \frac{z}{\sqrt{n}}\right) = 1 - 2 \sum_{i=1}^{\infty} (-1)^{i-1} \exp(-2i^2 z^2)$$

is used for large samples. For example, input the file `normal.tf1` and test to see if these numbers do come from a normal distribution. See if your own files `vector.1st` and `vector.2nd` come from a uniform or a beta distribution. Note that there are two ways to perform this test; you can state the parameters, or they can be estimated by the program from the sample, using the method of moments, or else maximum likelihood.

However, calculating parameters from samples compromises this test leading to a significant reduction in power. If you want to see if a sample comes from a binomial, Poisson, uniform, beta, gamma, lognormal normal, or Weibull distribution, etc., the data supplied must be of a type that is consistent with the supposed distribution, otherwise you will get error messages. Before you do any parametric test with a sample, you can always use this option to see if the sample is in fact consistent with the supposed distribution. An extremely valuable option provided is to view the best-fit *cdf* superimposed upon the sample cumulative distribution, which is a very convincing way to assess goodness of fit. Superposition of the best fit *pdf* on the sample histogram can also be requested, which is useful for discrete distributions but less useful for continuous distributions, since it requires large samples (say greater than 50) and the histogram shape depends on the number of bins selected. Table 10.11 illustrates the results when the test file `normal.tf1` is analyzed to see if the data are consistent with a normal distribution using the Kolmogorov-Smirnov test with parameters

```
Data: 50 numbers from a normal distribution mu = 0 and sigma = 1
Parameters estimated from sample are:
mu = -0.02579, se = 0.1422, 95%cl = (-0.3116, 0.26)
sigma = 1.006, sigma^2 = 1.011, 95%cl = (0.7055, 1.57)
Sample size = 50, i.e. number of x-values

H0: F(x) equals G(y) (x & theory are comparable) against
H1: F(x) not equal to G(y) (x & theory not comparable)
D = 0.09206
z = 0.651
p = 0.7559
H2: F(x) > G(y) (x tend to be smaller than theoretical)
D = 0.09206
z = 0.651
p = 0.3780
H3: F(x) < G(y) (x tend to be  larger than theoretical)
D = 0.0622
z = 0.4398
p = 0.4919
Shapiro-Wilks normality test:
W statistic = 0.9627
Sign. level = 0.1153     Tentatively accept normality
```

Table 10.11: Kolomogorov-Smirnov 1-sample and Shapiro-Wilks tests

estimated from the sample, and the Shapiro-Wilks test to be described shortly (page 133). Note that typical plots of the best fit normal distribution with the sample cumulative distribution, and best-fit density function overlayed on the sample histogram obtained using this procedure can be seen on page 22, while normal scores plots were discussed and illustrated on page 123. Note that significance levels are given in the table for upper-tail, lower-tail, and two-tail tests. In general you should only use the two-tail probability levels, reserving the one-tail tests for situations where the only possibility is either that the sample mean may be shifted to the right of the null distribution requiring an upper-tail test, or to the left requiring a lower-tail test

10.3.3 1-sample Shapiro-Wilks test for normality

This procedure is used when you have data and wish to test H_0: the sample is normally distributed. It is a very useful general test which may perform better than the Kolmogorov-Smirnov 1-sample test just described, but the power is low, so reasonably large sample sizes (say > 20) are required. The test statistic W, where $0 \le W \le 1$, is constructed by considering the regression of ordered sample values on the corresponding expected normal order statistics, so a normal scores plot should always be examined when testing for a normal distribution, and it should be approximately linear if a sample is from a normal distribution. For a sample of size n, this plot and the theory for the Shapiro-Wilks test, require the normal scores, i.e., the expected values of the rth largest order statistics given by

$$\begin{aligned} E(r,n) &= \frac{n!}{(r-1)!(n-r)!}\int_{-\infty}^{\infty} x[1-\Phi(x)]^{r-1}[\Phi(x)]^{n-r}\phi(x)\,dx,\\ \text{where } \phi(x) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right),\\ \text{and } \Phi(x) &= \int_{-\infty}^{x}\phi(u)\,du. \end{aligned}$$

Then the test statistic W uses the vector of expected values of a standard normal sample $x_1, x_2, \ldots, x_n$ and the corresponding covariance matrix, that is

$$m_i = E(x_i) \ (i = 1, 2, \ldots, n),$$
$$\text{and } v_{ij} = \text{cov}(x_i, x_j) \ (i, j = 1, 2, \ldots, n),$$

so that, for an ordered random sample $y_1, y_2, \ldots, y_n$,

$$W = \frac{\left(\sum_{i=1}^{n} a_i y_i\right)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2},$$
$$\text{where } a^T = m^T V^{-1}[(m^T V^{-1})(V^{-1}m)]^{-\frac{1}{2}}.$$

Finally, the significance level for the statistic W calculated from a sample is obtained by transformation to an approximately standard normal deviate using

$$z = \frac{(1 - W)^\lambda - \mu}{\sigma},$$

where λ is estimated from the sample and μ, and σ are the sample mean and standard deviation. Values of W close to 1 support normality, while values close to 0 suggest deviation from normality.

10.3.4 1-sample Dispersion and Fisher exact Poisson tests

This procedure is used when you have data in the form of non-negative integers (e.g. counts) and wish to test H_0: the sample is from a Poisson distribution. Given a sample of n observations x_i with sample mean $\bar{x} = \sum_{i=1}^{n} x_i/n$ from a Poisson distribution (page 419), the dispersion D given by

$$D = \sum_{i=1}^{n} (x_i - \bar{x})^2/\bar{x}$$

is approximately chi-square distributed with $n - 1$ degrees of freedom. A test for consistency with a Poisson distribution can be based on this D statistic but, with small samples, the more accurate Fisher exact test can be performed. This estimates the probability of the sample observed based on all partitions consistent with the sample mean, size and total. After performing these tests on a sample of nonnegative integers, this option then plots a histogram of the observed and expected frequencies (page 136). Table 10.12 shows the results from analyzing data in the test file `poisson.tf1` and also the results from using the previously discussed Kolmogorov 1-sample test with the same data. Clearly the data are consistent with a Poisson distribution. The mean and expectation of a Poisson distribution are identical and three cases can arise.

1. The sample variance exceeds the upper confidence limit for the sample mean indicating over-dispersion, i.e. too much clustering/clumping.

2. The sample variance is within the confidence limits for the sample mean indicating consistency with a Poisson distribution.

3. The sample variance is less than the lower confidence limit for the sample mean indicating under-dispersion, i.e. too much uniformity.

Output from the Kolmogorov-Smirnov 1-sample test for a Poisson distribution indicates if the variance is suspiciously small or large.

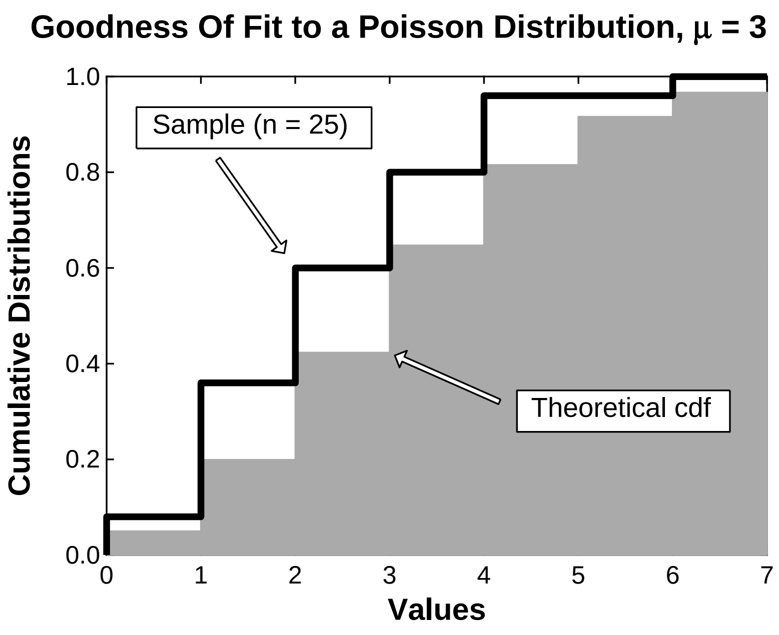
```
Dispersion and Fisher-exact Poisson tests
Sample size          = 40
Sample total         = 44
Sample ssq           = 80
Sample mean          = 1.1
Lower 95% con.lim.   = 0.7993
Upper 95% con.lim.   = 1.477
Sample variance      = 0.8103
Dispersion (D)       = 28.73
p = P(Chi-sq >= D)   = 0.88632
Number deg. freedom = 39
Fisher exact Prob.   = 0.91999

Kolmogorov-Smirnov one sample test
H0: F(x) equals G(y) (x & theory are comparable) against
H1: F(x) not equal to G(y) (x & theory not comparable)
D = 0.1079
z = 0.6822
p = 0.7003
H2: F(x) > G(y) (x tend to be smaller than theoretical)
D = 0.07597
z = 0.4805
p = 0.4808
H3: F(x) < G(y) (x tend to be larger than theoretical)
D = 0.1079
z = 0.6822
p = 0.3501
```

Table 10.12: Poisson distribution tests

10.3.5 Goodness of fit to a Poisson distribution

After using a Kolmogorov-Smirnov 1-sample or Fisher exact test to estimate goodness of fit to a Poisson distribution, the sample *cdf* can be compared with the theoretical *cdf* as in figure 10.3. The theoretical *cdf* is shown as a filled polygon for clarity. Also, sample and theoretical frequencies can be compared, as shown for a normal graph with bars as plotting symbols, and sample values displaced to avoid overlapping. Observe that the labels at $x = -1$ and $x = 8$ were suppressed, and bars were added as graphical objects to identify the bar types.



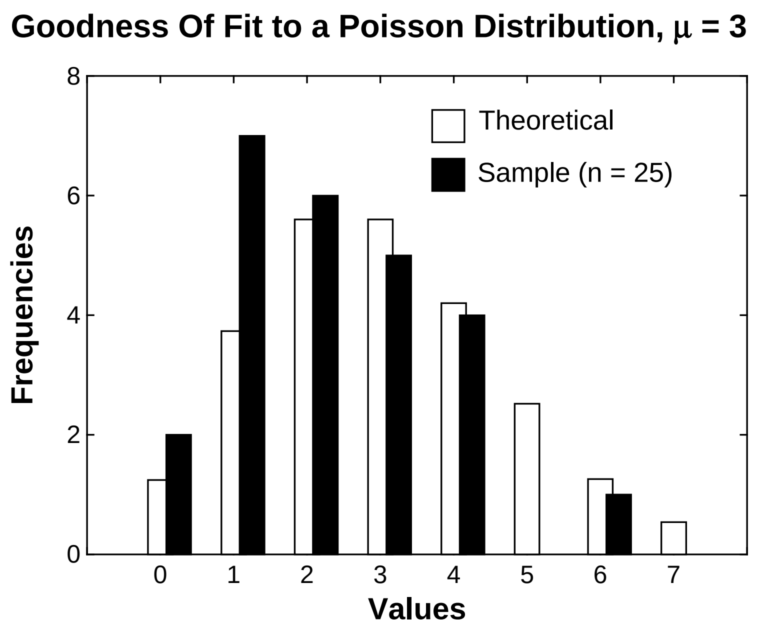


Figure 10.3: Goodness of fit to a Poisson distribution

10.3.6 2-sample unpaired *t* and variance ratio tests

This procedure is used when you have two samples $x = (x_1, x_2, \ldots, x_m)$ and $y = (y_1, y_2, \ldots, y_n)$ (i.e., two column vectors of measurements, not counts) which are assumed to come from two normal distributions with the same variance, and you wish to test H_0: the means of the two samples are equal. It is equivalent to 1-way analysis of variance (page 161) with just two columns. The test statistic U that is calculated is

$$\begin{aligned}
U &= \frac{\bar{x}-\bar{y}}{\sqrt{s_p^2\left(\dfrac{1}{m}+\dfrac{1}{n}\right)}},\\
\text{where } \bar{x} &= \sum_{i=1}^{m} x_i/m,\\
\bar{y} &= \sum_{i=1}^{n} y_i/n,\\
s_x^2 &= \sum_{i=1}^{m}(x_i-\bar{x})^2/(m-1),\\
s_y^2 &= \sum_{i=1}^{n}(y_i-\bar{y})^2/(n-1),\\
\text{and } s_p^2 &= \frac{(m-1)s_x^2+(n-1)s_y^2}{m+n-2}.
\end{aligned}$$

Here s_p^2 is the pooled variance estimate and U has a t distribution with $m+n-2$ degrees of freedom under H_0: the means are identical. The sample means and variances are calculated, then the test statistic and significance levels for lower, upper and two-tail tests. Note that before doing a t or paired t test, SIMFIT can do a Shapiro-Wilks test, which examines the correlation between the sample cumulative distribution and the expected order statistics to test for normal distributions. It can also do a variance ratio test for common variances, which calculates

$$\phi = \max\left(\frac{s_x^2}{s_y^2}, \frac{s_y^2}{s_x^2}\right).$$

A 2-tail probability for this F test with a and b degrees of freedom is twice the upper tail probability, as

$$P(F_{a,b} \geq \phi) = P(F_{b,a} \leq 1/\phi).$$

However, both the Shapiro-Wilks test, and also the F test are very weak with small samples, say less than 25. So, if you have large samples which do not pass these tests, ask yourself if doing a t test makes sense (since a t test depends upon the assumption that both samples are normal and with the same variance).

Note that the Satterthwaite procedure, using a t_c statistic with ν degrees of freedom calculated with the Welch correction for unequal variances is performed at the same time, using

$$\begin{aligned}
t_c &= \frac{\bar{x}-\bar{y}}{se(\bar{x}-\bar{y})}\\
se(\bar{x}-\bar{y}) &= \sqrt{\frac{s_x^2}{m}+\frac{s_y^2}{n}}\\
\nu &= \frac{se(\bar{x}-\bar{y})^4}{(s_x^2/m)^2/(m-1)+(s_y^2/n)^2/(n-1)}
\end{aligned}$$

and the results are displayed within square brackets adjacent to the uncorrected results. However, this should only be trusted if the data sets seem approximately normally distributed with fairly similar variances. Note that, every time SIMFIT estimates parameters by regression, it estimates the parameter standard error and does a t test for parameter redundancy. However, at any time subsequently, you can choose the option to compare two parameters and estimated standard errors from the curve fitting menus, which does the above test corrected for unequal variances. Table 10.13 shows the results from analyzing data in `ttest.tf4` and `ttest.tf5` which are not paired. Clearly the correction for unequal variance is unimportant in this case and the unpaired t test supports equality of the means.

```
Normal distribution test 1, Data: X-data for t test
Shapiro-Wilks statistic W = 0.9924
Significance level for W  = 1.0000  Tentatively accept normality

Normal distribution test 2, Data: Y-data for t test
Shapiro-Wilks statistic W = 0.998
Significance level for W  = 0.9999  Tentatively accept normality

F test for equality of variances
No. of x-values         = 12
Mean x                  = 120.0
Sample variance of x    = 457.5
Sample std. dev. of x   = 21.39
No. of y-values         = 7
Mean y                  = 101.0
Sample variance of y    = 425.3
Sample std. dev. of y   = 20.62
Variance ratio          = 1.076
Deg. of freedom (num)   = 11
Deg. of freedom (denom) = 6
P(F >= Variance ratio)  = 0.4894
Two tail p value        = 0.9788
Conclusion: Consider accepting equality of variances

Unpaired t test ([ ] = corrected for unequal variances)
Number of x-values           = 12
Number of y-values           = 7
Number of degrees of freedom = 17        [      13]
Unpaired t test statistic U  = 1.891     [  1.911 ]
p = P(t >= U) (upper tail p) = 0.0379    [  0.0391]
p = P(t =< U) (lower tail p) = 0.9621    [  0.9609]
p for two tailed t test      = 0.0757    [  0.0782]
Difference between means DM  = 19.0
Lower 95% con. limit for DM  = -2.194    [ -1.98  ]
Upper 95% con. limit for DM  = 40.19     [  39.98 ]
Conclusion: Consider accepting equality of means
```

Table 10.13: Unpaired t test

10.3.7 2-sample paired t test

This procedure is used when you have paired measurements, e.g., two successive measurements on the same subjects before and after treatments, and wish to test H_0: the mean of the differences between paired measurements is zero. Just as the unpaired t test is equivalent to analysis of variance with just two columns, the paired t test is equivalent to repeated measurements analysis of variance. For convenience, data for all paired tests also can be input as a n by 2 matrix rather than two vectors of length n. The paired t test is based on the assumption that the differences between corresponding pairs x_i and y_i are normally distributed, not necessarily the original data although this would normally have to be the case, and it requires the calculation of $\bar{d}$, s_d^2, and t_d given by

$$\begin{aligned}
d_i &= x_i - y_i \\
\bar{d} &= \sum_{i=1}^{n} d_i/n \\
s_d^2 &= \sum_{i=1}^{n} (d_i - \bar{d})^2/(n-1) \\
t_d &= \frac{\bar{d}}{\sqrt{s_d^2/n}}.
\end{aligned}$$

The test statistic t_d is again assumed to follow a t distribution with $n - 1$ degrees of freedom. For more details of the t distribution see page 422.

Table 10.14 shows the results from a paired t test with paired data from test files `ttest.tf2` and `ttest.tf3`, where the the test supports equality of means.

```
Paired t test
Number of degrees of freedom   =  9
Paired t test statistic S      = -0.904
p = P(t >= S)                  = 0.8052
p = P(t =< S)                  = 0.1948
p for two tailed t test        = 0.3895
Mean of differences MD         = -1.3
Lower 95% con. limit for MD    = -4.553
Upper 95% con. limit for MD    = 1.953
Conclusion: Consider accepting equality of means
```

Table 10.14: Paired t test

10.3.8 2-sample Kolmogorov-Smirnov test

This nonparametric procedure is used when you have two samples (column vectors) X of length m and Y of length n and you wish to test H_0: the samples are from the same, unspecified, distribution. It is a poor test unless both samples are fairly large (say > 20) and both come from a continuous and not a discrete distribution. The $D_{m,n}+$, $D_{m,n}-$ and $D_{m,n}$ values are obtained from the differences between the two sample cumulative distribution functions, then the test statistic

$$z = \sqrt{\frac{mn}{m+n}} D_{m,n}$$

is calculated. For small samples SIMFIT calculates significance levels using the formula

$$P(D_{m,n} \le d) = A(m,n) \bigg/ \binom{m+n}{n}$$

where $A(m,n)$ is the number of paths joining integer nodes from $(0,0)$ to (m,n) which lie entirely within the boundary lines defined by d in a plot with axes $0 \le X \le m$ and $0 \le Y \le n$, and where $A(u,v)$ at any intersection satisfies the recursion

$$A(u,v) = A(u-1,v) + A(u,v-1)$$

with boundary conditions $A(0, v) = A(u, 0) = 1$. However, for large samples, the asymptotic formula

$$\lim_{m,n\to\infty} P\left(\sqrt{\frac{mn}{m+n}}D_{m,n} \le z\right) = 1 - 2\sum_{i=1}^{\infty}(-1)^{i-1}\exp(-2i^2z^2)$$

is employed. For example, use the test files `ttest.tf4` and `ttest.tf5` to obtain the results shown in table 10.15. The test again supports equality of means. You could also try your own files `vector.1st` and

```
Size of X-data = 12
Size of Y-data =  7
H0: F(x) is  equal to G(y) (x and y are comparable) against
H1: F(x) not equal to G(y) (x and y not comparable)
D = 0.4405
z = 0.2095
p = 0.2653
H2: F(x) > G(y) (x tend to be smaller than y)
D = 0.0
z = 0.0
p = 0.5
H3: F(x) < G(y) (x tend to be  larger than y)
D = 0.4405
z = 0.2095
p = 0.1327
```

Table 10.15: Kolmogorov-Smirnov 2-sample test

`vector.2nd` (prepared previously) to illustrate a very important set of principles. For instance, it is obvious to you what the values in the two samples suggest about the possibility of a common distribution. What do the upper, lower and two tail tests indicate ? Do you agree ? What happens if you put your vector files in the other way round ? Once you have understood what happens to these data sets you will be a long way towards being able to analyze your own pairs of data sets. Note that, if data have been input from files, **simstat** saves the last set of files for re-analysis, for instance to do the next test.

10.3.9 2-sample Wilcoxon-Mann-Whitney U test

The Mann-Whitney U nonparametric procedure (which is equivalent to the Wilcoxon rank-sum test) is used when you have two samples (column vectors) and wish to test H_0: the samples have the same medians, against H_A: the distributions are not equivalent, e.g., one sample dominates the other in distribution. Although the test only works optimally for continuous data, it can be useful for scored data, where the order is meaningful but not the numerical magnitude of differences. The two samples, x of size m and y of size n, are combined, then the sums of the ranks of the two samples in the combined sample are used to calculate exact significance levels for small samples, or asymptotic values for large samples. The test statistic U is calculated from the ranks r_{xi} in the pooled sample, using average ranks for ties, as follows

$$R_x = \sum_{i=1}^{m} r_{xi}$$
$$U = R_x - \frac{m(m+1)}{2}.$$

The statistic U is also the number of times a score in sample y precedes a score in sample x, counting a half for tied scores, so large values suggest that x values tend to be larger than y values.

For example, do exactly as for the t test using `ttest.tf4` and `ttest.tf5` and compare the the results as displayed in table 10.16 with table 10.15 and table 10.13. The null hypothesis $H_0 : F(x) = G(y)$ is that two

```
Size of X-data = 12
Size of Y-data = 7
U = 62.5
z = 1.691
H0: F(x) is equal to G(y) (x and y are comparable)
    as null hypothesis against the alternatives:-
H1: F(x) not equal to G(y) (x and y not comparable)
p = 0.0873
H2: F(x) > G(y) (x tend to be smaller than y)
p = 0.9605
H3: F(x) < G(y) (x tend to be  larger than y)
p = 0.0436    Reject H0 at 5% s-level
```

Table 10.16: Wilcoxon-Mann-Whitney U test

samples are identically distributed, and the appropriate rejection regions are

$$\begin{aligned} &U_x \le u_\alpha \text{ for } H_1 : F(x) \ge G(y)\\ &U_y \le u_\alpha \text{ for } H_1 : F(x) \le G(y)\\ &U_x \le u_{\alpha/2} \text{ or } U_y \le u_{\alpha/2} \text{ for } H_1 : F(x) \ne G(y)\end{aligned}$$

where the critical points u_α can be calculated from the distribution of U. Defining $r_{m,n}(u)$ as the number of distinguishable arrangements of the m X and n Y variables such that in each sequence Y precedes X exactly u times, the recursions

$$\begin{aligned} r_{m,n}(u) &= r_{m,n-1}(u) + r_{m-1,n}(u-n)\\ P(U=u) &= r_{m,n} \Big/ \binom{m+n}{m}\\ &= p_{m,n}(u)\\ &= \left(\frac{n}{m+n}\right) p_{m,n-1}(u) + \left(\frac{m}{m+n}\right) p_{m-1,n}(u-n)\end{aligned}$$

are used by SɪᴍFɪTto calculate exact tail probabilities for $n, m \le 40$ or $m+n \le 50$, but for larger samples a normal approximation is used. The parameter z in table 10.16 is the approximate normal test statistic given by

$$\begin{aligned} z &= \frac{U - mn/2 \pm 0.5}{\sqrt{V(U)}}\\ \text{where } V(U) &= \frac{mn(m+n+1)}{12} - \frac{mnT}{(m+n)(m+n-1)}\\ \text{and } T &= \sum_{j=1}^{\tau} \frac{t_j(t_j-1)(t_j+1)}{12}\end{aligned}$$

with τ groups of ties containing t_j ties per group. The equivalence of this test using test statistic $U = U_x$ and the Wilcoxon rank-sum test using test statistic $R = R_x$ will be clear from the identities

$$\begin{aligned} U_x &= R_x - m(m+1)/2\\ U_y &= R_y - n(n+1)/2\\ U_x + U_y &= mn\\ R_x + R_y &= (m+n)(m+n+1)/2.\end{aligned}$$

Many people recommend the consistent use of this test instead of the t or Kolmogorov-Smirnov tests, so you should try to find out why we need two nonparametric tests. For instance; do they both give the same results?;

should you always use both tests?; are there circumstances when the two tests would give different results?; is rejection of H_0 in the one-tail test of table 10.16 to be taken seriously with such small sample sizes, and so on. Note that the Kruskal-Wallis test (page 165) is the extension of the Mann-Whitney U test to more than two independent samples.

10.3.10 2-sample Wilcoxon signed-ranks test

This procedure is used when you have two paired samples, e.g., two successive observations on the same subjects, and wish to test H_0: the median of the differences is zero. Just as the Mann-Whitney U test is the nonparametric equivalent of the unpaired t test, the Wilcoxon paired-sample signed-ranks test is the nonparametric equivalent of the paired t test. If the data are counts, scores, proportions, percentages, or any other type of non-normal data, then these tests should be used instead of the t tests. Table 10.17 shows the

```
Size of data = 10
Number of values suppressed = 0
W =  17.0
z = -1.027
H0: X median = Y median
    as null hypothesis against the alternatives:-
H1: Medians are not equal
p = 0.2480
H2: X median < Y median
p = 0.1240
H3: X median > Y median
p = 0.8047
```

Table 10.17: Wilcoxon signed-ranks test

results from analyzing the data in `ttest.tf2` and `ttest.tf3`, which was previously done using the paired t test (page139). The test examines the pairwise differences between two samples of size n to see if there is any evidence to support a difference in location between the two populations, i.e. a nonzero median for the vector of differences between the two samples. It is usual to first suppress any values with zero differences and to use a zero test median value. The vector of differences is replaced by a vector of absolute differences which is then ranked, followed by restoring the signs and calculating the sum of the positive ranks T^+, and the sum of negative ranks T^-, where clearly

$$T^+ + T^- = n(n+1)/2.$$

The null hypothesis $H_0 : M = M_0$ is that the median difference M equals a chosen median M_0, which is usually input as zero, and the appropriate rejection regions are

$$\begin{aligned}
&T^- \le t_\alpha \text{ for } H_1 : M > M_0\\
&T^+ \le t_\alpha \text{ for } H_1 : M < M_0\\
&T^+ \le t_{\alpha/2} \text{ or } T^- \le t_{\alpha/2} \text{ for } H_1 : M \ne M_0
\end{aligned}$$

where the critical points t_α can be calculated from the distribution of T, which is either T^+ or T^- such that $P(T \le t_\alpha) = \alpha$. If $u_n(k)$ is the number of ways to assign plus and minus signs to the first n integers, then

$$\begin{aligned}
P(T_n^+ = k) &= \frac{u_n(k)}{2^n}\\
&= \frac{u_{n-1}(k-n) + u_{n-1}(k)}{2^n}
\end{aligned}$$

which is used by SIMFIT to calculate exact tail probabilities for $n \leq 80$. The normal approximation z in table 10.17 is defined as

$$z = \frac{|A| - 0.5}{\sqrt{V}}$$
$$\text{where } A = [T - [n(n+1) - m(m+1)]/4]$$
$$\text{and } V = [n(n+1)(2n+1) - m(m+1)(2m+1) - R/2]/24.$$

Here m is the number of zero differences included in the analysis, if any, and $R = \Sigma r_i^2(r_i + 1)$ is the sum of tied ranks, excluding any due to zero differences and, for $n > 80$, tail areas are calculated using this normal approximation.

10.3.11 Chi-square test on observed and expected frequencies

This test is used when you have a sample of n observed frequencies O_i and a corresponding set of expected frequencies E_i and wish to test that the observed frequencies are consistent with the distribution generating the expected values, by calculating the statistic C given by

$$C = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}.$$

If the expected frequencies are exact and the observed values are representative of the supposed distribution, then C is asymptotically distributed as chi-square with ν degrees of freedom. In the usual case, the expected frequencies are not exact but are calculated using m parameters estimated from the sample, so that the degrees of freedom are given by

$$\nu = n - 1 - m.$$

Table 10.18 illustrates the results with test files `chisqd.tf2` and `chisqd.tf3` and one estimated parameter.

```
Number of partitions (bins)  = 6
Number of deg. of freedom    = 4
Chi-square test stat. C      = 1.531
p = P(chi-square >= C)       = 0.8212     Consider accepting H0
Upper tail 5% crit. point    = 9.488
Upper tail 1% crit. point    = 13.28
```

Table 10.18: Chi-square test on observed and expected frequencies

The test requires rather large samples, so that the expected values are all positive integers, if possible say ≥ 5, and the number of observed values, i.e. bins n, is sufficient for a reasonable number of degrees of freedom in the chi-square test. If the total number of observations is k, then the number of bins n used to partition the data is often recommended to be of the order

$$n \approx k^{0.4}$$

but this, and the intervals used to partition the data, depend on the shape of the assumed distribution. Of course, the significance level for the test will depend on the number of bins used, and the intervals selected to partition the sample of observations into bins.

Figure 10.4 illustrates a bar chart for these data that can be inspected to compare the observed and expected values visually.

10.3.12 Chi-square, Fisher-exact, and loglinear contingency table tests

A contingency table is an array of nonnegative frequencies with n rows and m columns, such as this table contained in SIMFIT test file `chisqd.tf4`, for 15 observations carried out on two populations to test for equal probabilities of success.

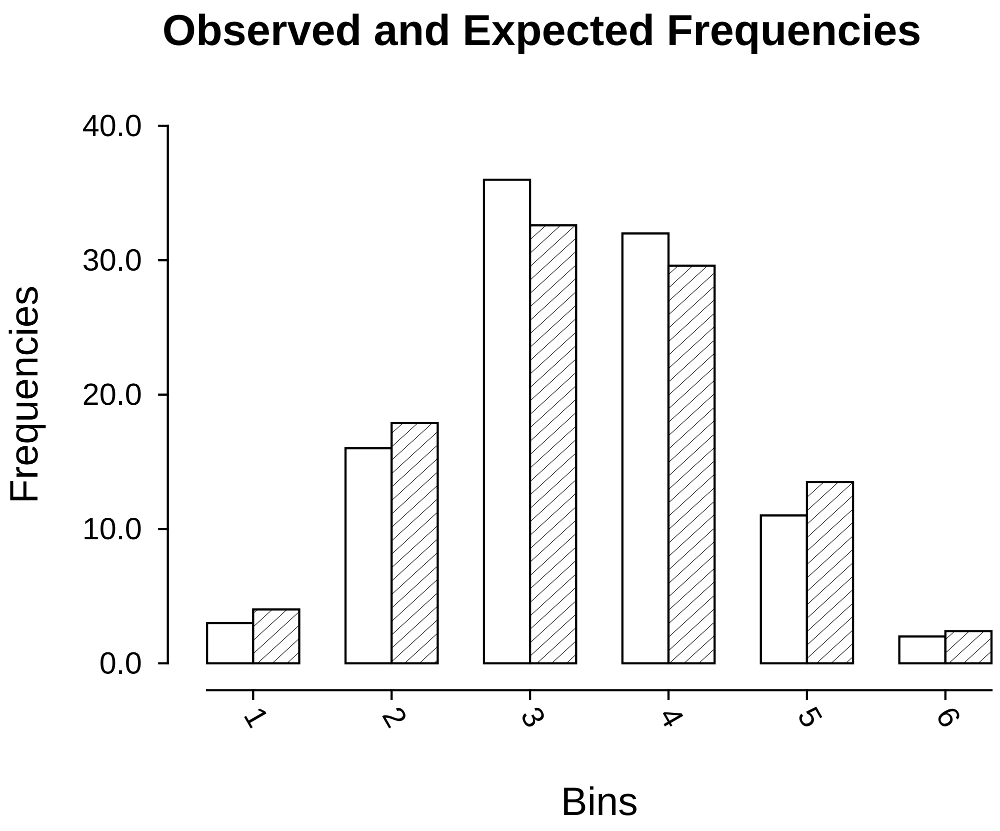


Figure 10.4: Observed and Expected frequencies

	Success	Failure	
Sample 1	3	3	6
Sample 2	7	2	9
	10	5	15

Here, the cell frequencies are $(3, 3, 7, 2)$, the sum of row frequencies known as row marginals are $(6, 9)$, the sum of column frequencies known as column marginals are $(10, 5)$, and obviously the row and column marginals must separately both add up to the total number of frequencies (15).

To be precise, in the general case there will be frequencies f_{ij} where $i = 1, 2, \ldots, n$, and $j = 1, 2, \ldots, m$, and it is wished to test for homogeneity, i.e. independence, or no association between the variables, which can be stated as the null hypothesis

$$H_0 : \mu_{ij} = \mu_{i.}\mu_{.j}, \text{ for } i = 1, 2, \ldots, n, \text{ and } j = 1, 2, \ldots, m$$

where each cell probability μ_{ij} is completely determined by the corresponding row marginal $\mu_{i.}$, and the column marginal $\mu_{.j}$. To examine a given data set SIMFIT provides the following three alternatives,which are based on the hypergeometric distribution (page 418) and the chi-square distribution (page 423).

1. **The chi-square contingency table test.**
   This is the easiest to perform and and interpret, and is the test most generally used. However, it must be emphasized that the test statistic is only asymptotically distributed as chi-square with $(n-1)(m-1)$

degrees of freedom in the limit for large samples. Where there are small frequencies the option to combine cells should be considered, and note that the Yate's continuity correction may be used where appropriate.

2. **The Fisher exact contingency table test.**
   This is very powerful and widely used, but sometimes suffers from being difficult to interpret with large samples, which also may lead to computational problems.

3. **The loglinear contingency table test.**
   This uses general linear modeling assuming a Poisson error distribution and log link, but it does require some expertise on the part of users.

10.3.12.1 The chi-square contingency table test

For all tables, SIMFIT calculates a chi-square test statistic C from the observed frequencies f_{ij}, and expected frequencies e_{ij}, and also a likelihood ratio test statistic L defined in terms of the expected values e_{ij} and marginals $f_{i.}$ and $f_{.j}$ as follows

$$\begin{aligned}
e_{ij} &= f_{i.}f_{.j}/N \\
C &= \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{(f_{ij}-e_{ij})^2}{e_{ij}} \\
L &= -2\log\lambda \\
&= 2\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}\log(f_{ij}/e_{ij})
\end{aligned}$$

It is often recommended to combine cells where the expected values are small, say $e_{ij} < 0.5$, and this facility is provided.

Select chi-square contingency table analysis, then analyze the above data which leads to table 10.19 showing

```
Number of rows        = 2
Number of columns     = 2
Chi-sq. test stat. C = 0.3125
Degrees of freedom    = 1
p = P(chi-sq. >= C)   = 0.5762
Upper tail 5% point   = 3.841
Upper tail 1% point   = 6.635
L = -2*log(lambda)    = 1.243
p = P(chi-sq. >= L)   = 0.2649
Yate's correction used in chi-square
```

Table 10.19: Chi-square and likelihood ratio contingency table tests: 2 by 2

the results from calculation of the approximate chi-square test statistic with the Yate's continuity correction

$$C = \frac{N(|f_{11}f_{22} - f_{12}f_{21}| - N/2)^2}{r_1 r_2 c_1 c_2}$$

for this 2 by 2 contingency table, where N is the sum of frequencies f_{ij}, r_i are the row marginals, and c_j are the column marginals. Clearly, the results do not suggest rejecting H_0.

```
Data: Test file chisqd.tf4: Fisher exact test
 Observed   Rearranged so R1 = smallest marginal, C2 >= C1
    3   3            3   2
    7   2            3   7

 p(  r) = p for f(1,1) = r after rearranging and adjusting
 p(  0) = 0.041958
 p(  1) = 0.251748
 p(  2) = 0.419580
 p(  3) = 0.239760  p(*), observed frequencies
 p(  4) = 0.044955
 p(  5) = 0.001998
 P_Sums, for 1-tail and 2-tail test statistics
 P_sum1 = 0.041958  sum of p(r) =< p(*) for r < 3
 P_sum2 = 0.953047  sum of all p(r) for r =< 3
 P_sum3 = 0.286713  sum of all p(r) for r >= 3
 P_sum4 = 0.046953  sum of p(r) =< p(*) for r > 3
 P_sum5 = 1.000000  P_sum2 + P_sum4
 P_sum6 = 0.328671  P_sum1 + P_sum3
```

Table 10.20: Fisher exact contingency table test 1

10.3.12.2 The Fisher exact contingency table test

For 2 by 2 contingency tables, and $N \le 100$, results are displayed. as in table 10.20. For convenience, this test starts by rearranging the data table until r_1 is the smallest marginal and $c_2 \ge c_1$. Then all hypothetical tables that are possible with the same marginals are considered, but now for $r = f_{11}$ for $r = 0, 1, \ldots, r_1$ as follows, where the observed frequencies are indicated by stars (*).

0	5	1	4	2	3	*3	*2	4	1	5	0
6	4	5	5	4	6	*3	*7	2	8	1	9

Assuming the null hypothesis, the probabilities $p(r)$ for tables with $f_{11} = r$ are then calculated for a hypergeometric distribution using

$$\begin{aligned} p(r) &= \frac{r_1!r_2!c_1!c_2!}{f_{11}!f_{21}!f_{12}!f_{22}!N!} \\ &= \frac{r_1!r_2!c_1!c_2!}{N!r!(r_1-r)!(c_1-r)!(N-c_1-r_1+r)!}, \quad r = 0, 1, \ldots, r_1. \end{aligned}$$

With the tables under consideration it is clear that, had the outcome been as for the hypothetical tables indicated by $p(0)$, $p(4)$, or $p(5)$ then the possibility of rejecting H_0 would have to be considered. However, the current data $p(3)$, indicated by $p(*)$ would be accepted, as for the chi-square test on the same data. With less obvious results, various one-tailed and two-tailed tests can be based on considering probabilities for more extreme contingency tables, or sums of such probabilities. As an example consider the following data

	Boys	Girls	
Left-handed	6 (18%)	12 (22%)	18
Right-handed	28 (82%)	24 (67%)	52
	34	36	70

and possible hypotheses for this sample
H_0: left-handedness is not less common in boys than girls
H_A: left-handedness is less common in boys than girls.
Table 10.21 presents the results. Here, adding up the probabilities for the observed table $p(6) = p(*)$ and

```
Observed   Rearranged so R1 = smallest marginal, C2 >= C1
   6  12           6  12
  28  24          28  24

p(  r) = p for f(1,1) = r after rearranging and adjusting
p(  0) = 0.000000
p(  1) = 0.000013
p(  2) = 0.000177
p(  3) = 0.001436
p(  4) = 0.007590
p(  5) = 0.027720
p(  6) = 0.072572  p(*), observed frequencies
p(  7) = 0.139338
p(  8) = 0.198959
p(  9) = 0.212877
p( 10) = 0.171062
p( 11) = 0.102959
p( 12) = 0.046046
p( 13) = 0.015082
p( 14) = 0.003535
p( 15) = 0.000571
p( 16) = 0.000060
p( 17) = 0.000004
p( 18) = 0.000000
P_Sums, for 1-tail and 2-tail test statistics
P_sum1 = 0.036936  sum of p(r) =< p(*) for r < 6
P_sum2 = 0.109508  sum of all p(r) for r =< 6
P_sum3 = 0.963064  sum of all p(r) for r >= 6
P_sum4 = 0.065297  sum of p(r) =< p(*) for r > 6
P_sum5 = 0.174805  P_sum2 + P_sum4
P_sum6 = 1.000000  P_sum1 + P_sum3
```

Table 10.21: Fisher exact contingency table test 2

all the possible tables more extreme than this that would favor H_A against H_0 we see that the appropriate one-tailed p value is

$$p(0) + p(1) + p(2) + p(3) + p(4) + p(5) + p(6) = 0.109508$$

and so, for this sample with $\alpha = 0.05$ we would not consider rejecting H_0.

10.3.12.3 The loglinear contingency table test

Analysis of the data in `chisqd.tf5` shown in table 10.22 and table 10.23 illustrates another feature of contingency table analysis. For 2 by 2 tables the Fisher exact, chi-square, and likelihood ratio tests are usually adequate but, for larger contingency tables with no very small cell frequencies it may be useful to fit a log-linear model. To do this, SIMFIT defines dummy indicator variables for the rows and columns (page 55), then fits a generalized linear model (page 49) assuming a Poisson error distribution and log link, but imposing the constraints that the sum of row coefficients is zero and the sum of column coefficients is zero, to avoid fitting an overdetermined model (page 52). The advantage of this approach is that the deviance, predicted frequencies, deviance residuals, and leverages can be calculated for the model

$$\log(\mu_{ij}) = \theta + \alpha_i + \beta_j,$$

```
Observed chi-square frequencies
      6      15      10      38      62      26
     16      12       9      22      36       5
Number of rows       = 2
Number of columns    = 6
Chi-sq. test stat. C = 18.59
No. deg. of freedom  = 5
p = P(chi-sq. >= C)  = 0.0023     Reject H0 at 1% sig.level
Upper tail 5% point  = 11.07
Upper tail 1% point  = 15.09
L = -2*log(lambda)   = 19.24
P(chi-sq. >= L)      = 0.0017     Reject H0 at 1% sig.level
```

Table 10.22: Chi-square and likelihood ratio contingency table tests: 2 by 6

```
Deviance (D) = 1.924E+01, deg.free. = 5
P(chi-sq>=D) = 0.0017  Reject H0 at 1% sig.level
Parameter   Estimate  Std.Err.  ..95% con. lim....    p
Constant       2.856    0.0748      2.66      3.05  0.0000
Row  1        0.2255     0.064    0.0611      0.39  0.0168   *
Row  2       -0.2255     0.064     -0.39   -0.0611  0.0168   *
Col  1       -0.4831     0.189    -0.969   0.00257  0.0508  **
Col  2       -0.2783     0.173    -0.724     0.168  0.1696 ***
Col  3       -0.6297     0.201     -1.15    -0.112  0.0260   *
Col  4        0.5202     0.128      0.19     0.851  0.0098
Col  5         1.011      0.11     0.727      1.29  0.0003
Col  6       -0.1401     0.164    -0.562     0.281  0.4320 ***
     Data      Model     Delta  Residual  Leverage
        6      13.44     -7.44   -2.2808    0.6442
       15      16.49     -1.49   -0.3737    0.6518
       10      11.61     -1.61   -0.4833    0.6397
       38      36.65      1.35    0.2210    0.7017
       62      59.87      2.13    0.2740    0.7593
       26      18.94      7.06    1.5350    0.6578
       16       8.56      7.44    2.2661    0.4414
       12      10.51      1.49    0.4507    0.4533
        9       7.39      1.61    0.5713    0.4343
       22      23.35     -1.35   -0.2814    0.5317
       36      38.13     -2.13   -0.3486    0.6220
        5      12.06     -7.06   -2.3061    0.4628
```

Table 10.23: Loglinear contingency table analysis

where μ_{ij} are the expected cell frequencies expressed as functions of an overall mean θ, row coefficients α_i, and column coefficients β_j. The row and column coefficients reflect the main effects of the categories, according to the above model, where

$$\sum_{i=1}^{n}\alpha_i = \sum_{j=1}^{m}\beta_j = 0$$

and the deviance, which is a likelihood ratio test statistic, can be used to test the justification for a mixed term γ_{ij} in the saturated model

$$\log(\mu_{ij}) = \theta + \alpha_i + \beta_j + \gamma_{ij},$$

which fits exactly, i.e., with zero deviance. SimFIT performs a chi-square test on the deviance to test the null hypotheses of homogeneity, which is the same as testing that all γ_{ij} are zero, the effect of individual cells can be assessed from the leverages, and various deviance residuals plots can be done to estimate goodness of fit of the assumed log-linear model.

10.3.13 McNemar test

The McNemar test is used to analyze paired observations of a dichotomous variable, i.e. where there can only be one of two possible values such as: success/failure, +/-, 0/1, etc. and it is of interest to examine if the paired values are associated or are independent.

To be precise, consider the possible outcome from testing fifty specimens of sputum cultured on two different culture media, A and B, with the intention of detecting a particular bacterium. The four possible outcomes were as follows.

Type	Medium A	Medium B	Number
Both	+	+	20
A only	+	-	12
B only	-	+	2
Neither	-	-	16

These data can be arranged as a 2 by 2 contingency table, such as this table contained in SimFIT test file `mcnemar.tf1`.

	B +	B -	Total
A +	20	12	32
A -	2	16	18
Total	22	28	50

Here, the cell frequencies are $(f_{11} = 20, f_{12} = 12, f_{21} = 2, f_{22} = 16)$, the sum of row frequencies known as row marginals are $(32, 18)$, the sum of column frequencies known as column marginals are $(22, 28)$, and obviously the row and column marginals must separately both add up to the total number of frequencies $(n = 50)$.

Analysis of these data produces table 10.24 where frequencies f_{ij} are analyzed by calculating the χ^2 test

```
McNemar test

H0: Expected value of [(f(1,2) - (f(2,1))/n] = 0
Data for 2 by 2 McNemar test
Number of rows/columns  = 2
Chi-sq. test stat. C    = 5.786
Number deg. of freedom  = 1
p = P(chi-sq. >= C)     = 0.0162     Reject H0 at 5% level
Upper tail 5% point     = 3.841
Upper tail 1% point     = 6.635

Continuity correction used in chi-square
```

Table 10.24: McNemar 2 by 2 test

statistic given by

$$\chi^2 = \frac{(|f_{12} - f_{21}| - 1)^2}{f_{12} + f_{21}}.$$

which has an approximate chi-square distribution with 1 degree of freedom. The outcome emphasizes the obvious fact that culture medium A is more effective than culture medium B.

Note that this test does not perform so well with small frequencies and, in particular, if $r = 2$ and

$$f_{12} + f_{21} \le 20$$

a warning will be displayed. In such cases it may be preferable to do a binomial test using $N = f_{12} + f_{21}$ then $X = f_{12}$ or $X = f_{21}$ to check if $\hat{p} = X/N$ is consistent with a binomial distribution with parameters N and $p = 0.5$. Since in the 2 by 2 case the McNemar test is equivalent to testing if two sample estimates for a binomial probability parameter differ significantly, we can use SimFIT to calculate exact 95% confidence limits as follows

$$\begin{aligned} \text{For } \ 2/14: &\quad 0.0178 \le \hat{p} = 0.1429 \le 0.4281 \\ \text{For } 12/14: &\quad 0.5719 \le \hat{p} = 0.8571 \le 0.9822 \end{aligned}$$

which convincingly demonstrates the superiority of culture medium A over culture medium B.

To explain the logic behind this analysis, note that the overall proportion of successes with medium A is $(f_{11} + f_{12})/n$, while the overall proportion of successes with medium B is $(f_{11} + f_{21})/n$, so that the difference between these estimates for the probability of success depends only on $f_{12} - f_{21}$, and the null hypothesis for such a 2 by 2 table can be expressed as expectations in several equivalent ways without using the diagonal frequencies f_{ii} except in the sample size n, such as

$$\begin{aligned} H_0 : E\left(\frac{f_{12} - f_{21}}{n}\right) &= 0, \text{ or} \\ H_0 : E\left(\frac{f_{12}}{f_{21}}\right) &= 1. \end{aligned}$$

Note that it is important that tables for the McNemar test are set up correctly to reflect the pairwise nature of the data, so an additional example is given using data in test file `mcnemar.tf2` for the case where medication A was applied to one arm and medication B to the other arm with subjects suffering from a rash on both arms.

```
Chi-sq. test stat. C          = 0.5625
Number of degrees of freedom  = 1
p = P(chi-sq. >= C)           = 0.4533     Consider accepting H0
Upper tail 5% point           = 3.841
Upper tail 1% point           = 6.635
```

The outcome is that there is no evidence to support a significant difference between medications A and B in this experiment.

More generally, for larger r by r tables with identical paired row and column categorical variables, the continuity correction is not used, and the appropriate test statistic is

$$\chi^2 = \sum_{i=1}^{r} \sum_{j>i} \frac{(f_{ij} - f_{ji})^2}{f_{ij} + f_{ji}}$$

with $r(r-1)/2$ degrees of freedom. Unlike the normal contingency table analysis where the null hypothesis is independence of rows and columns, with this test there is intentional association between rows and columns. The test statistic does not use the diagonal frequencies f_{ii} and is testing whether the upper right corner of the table is symmetrical with the lower left corner. The SimFIT test file `mcnemar.tf3` contains data for a such a 3 by 3 McNemar table, as follows.

Table 10.25 illustrates this test by showing that the analysis of data in `mcnemar.tf1` is consistent with association between the variables.

```
Data for McNemar test
    173      20       7
     15      51       2
      5       3      24
H0: intentional association between row and column data.
Data: Data for McNemar test (details at end of file)
Number of rows/columns     = 3
Chi-sq. test stat. C       = 1.248
Number degrees of freedom  = 3
p = P(chi-sq. >= C)        = 0.7416     Consider accepting H0
Upper tail 5% point        = 7.815
Upper tail 1% point        = 11.34
```

Table 10.25: McNemar 3 by 3 test

10.3.14 Cochran Q repeated measures test on a matrix of 0,1 values

This procedure is used for a randomized block or repeated-measures design with a dichotomous variable. The blocks (e.g., subjects) are in rows from 1 to n of a matrix while the attributes, which can be either 0 or 1, are in groups, that is, columns 1 to m. So, with n blocks, m groups, G_i as the number of attributes equal to 1 in group i, and B_j as the number of attributes equal to 1 in block j, then the statistic Q is calculated, where

$$Q = \frac{(m-1)\left[\sum_{i=1}^{m} G_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} G_i\right)^2\right]}{\sum_{j=1}^{n} B_j - \frac{1}{m}\sum_{j=1}^{n} B_j^2}$$

and Q is distributed as approximately chi-square with $m-1$ degrees of freedom. It is recommended that m should be at least 4 and mn should be at least 24 for the approximation to be satisfactory.

For example, try the test file `cochranq.tf1` to obtain the results shown in table 10.26, noting that rows with

```
Data for Cochran Q test
          A B C D E
subject-1 0 0 0 1 0
subject-2 1 1 1 1 1
subject-3 0 0 0 1 1
subject-4 1 1 0 1 0
subject-5 0 1 1 1 1
subject-6 0 1 0 0 1
subject-7 0 0 1 1 1
subject-8 0 0 1 1 0
Number of blocks (rows) = 7
Number of groups (cols) = 5
Cochran Q value         = 6.947
p = P(chi-sqd. >= Q)    = 0.1387
95% chi-sq. point       = 9.488
99% chi-sq. point       = 13.28
```

Table 10.26: Cochran Q repeated measures test

all 0 or all 1 are not counted, while you can, optionally have an extra column of successive integers in order from 1 to n in the first column to help you identify the subjects in the results file. Clearly, the test provides no reason to reject the null hypothesis that the binary response of the subjects is the same for the variables called A, B, C, D, E in table 10.26.

10.3.15 The binomial test

This procedure, which is based on the binomial distribution (page 417), is used with dichotomous data, i.e., where an experiment has only two possible outcomes and it is wished to test H_0: binomial $p = p_0$ for some $0 \le p_0 \le 1$. For instance, to test if success and failure are subject to pre-determined probabilities, e.g., equally likely. You input the number of successes, k, the number of Bernoulli trials, N, and the supposed probability of success, p, then the program calculates the probabilities associated with k, N, p, and $l = N - k$ including the estimated probability parameter $\hat{p}$ with 95% confidence limits, and the two-tail binomial test statistic. The probabilities, which can be used for upper-tail, lower-tail, or two-tail testing are

$$\begin{aligned}
\hat{p} &= k/N \\
P(X = k) &= \binom{N}{k} p^k (1-p)^{N-k} \\
P(X > k) &= \sum_{i=k+1}^{N} \binom{N}{i} p^i (1-p)^{N-i} \\
P(X < k) &= \sum_{i=0}^{k-1} \binom{N}{i} p^i (1-p)^{N-i} \\
P(X = l) &= \binom{N}{l} p^l (1-p)^{N-l} \\
P(X > l) &= \sum_{i=l+1}^{N} \binom{N}{i} p^i (1-p)^{N-i} \\
P(X < l) &= \sum_{i=0}^{l-1} \binom{N}{i} p^i (1-p)^{N-i} \\
P(\text{two tail}) &= \min(P(X \ge k), P(X \le k)) + \min(P(X \ge l), P(X \le l)).
\end{aligned}$$

Table 10.27 shows, for example, that the probability of obtaining five successes (or alternatively five failures) in an experiment with equiprobable outcome would not lead to rejection of $H_0 : p = 0.5$ in a two tail test. Note, for instance, that the exact confidence limit for the estimated probability includes 0.5. Many life scientists when asked what is the minimal sample size to be used in an experiment, e.g. the number of experimental animals in a trial, would use a minimum of six, since the null hypothesis of no effect would never be rejected with a sample size of five.

10.3.16 The sign test

This procedure, which is also based on the binomial distribution (page 417) but assuming the special case $p = 0.5$, is used with dichotomous data, i.e., where an experiment has only two possible outcomes and it is wished to test if success and failure are equally likely. The test is rather weak and large samples, say greater than 20, are usually recommended. For example, just enter the number of positives and negatives and observe the probabilities calculated. Table 10.28 could be used, for instance, to find out how many consecutive successes you would have to observe before the likelihood of an equiprobable outcome would be questioned. Obviously five successes and five failures is perfectly consistent with the null hypothesis $H_0 : p = 0.5$, but see next what happens when the pattern of successes and failures is considered. Note that the Friedman test (page 169) is the extension of the sign test to more than two matched samples.

```
Successes K = 5
Trials N    = 5
L = (N - K) = 0
p-theory    = 0.50000
p-estimate  = 1.00000 (95% c.l. = 0.47818,1.00000)
P( X >  K ) = 0.00000
P( X <  K ) = 0.96875
P( X =  K ) = 0.03125
P( X >= K ) = 0.03125
P( X =< K ) = 1.00000
P( X >  L ) = 0.96875
P( X <  L ) = 0.00000
P( X =  L ) = 0.03125
P( X >= L ) = 1.00000
P( X =< L ) = 0.03125
Two tail binomial test statistic = 0.06250
```

Table 10.27: Binomial test

```
Sign test analysis with m + n = 10
P( +ve =  m ) = 0.24609,  m = 5
P( +ve >  m ) = 0.37695
P( +ve <  m ) = 0.37695
P( +ve >= m ) = 0.62305
P( +ve =< m ) = 0.62305
P( -ve =  n ) = 0.24609,  n = 5
P( -ve <  n ) = 0.37695
P( -ve >  n ) = 0.37695
P( -ve =< n ) = 0.62305
P( -ve >= n ) = 0.62305
Two tail sign test statistic = 1.00000
```

Table 10.28: Sign test

10.3.17 The run test

This is also based on an application of the binomial distribution (page 417) and is used when the sequence of successes and failures (presumed in the null hypothesis to be equally likely) is of interest, not just the overall proportions. For instance, the sequence

$$+++--++---+-$$

or alternatively

$$111001100010$$

has twelve items with six runs, as will be clear by adding brackets like this

$$(aaa)(bb)(aa)(bbb)(a)(b).$$

You can perform this test by providing the number of items, signs, then runs, and you will be warned if the number of runs is inconsistent with the number of positive and negative signs, otherwise the probability of the number of runs given the number of positives and negatives will be calculated. Again, rather large samples are recommended. For instance, what is the probability of a sample of ten new born babies consisting of five boys and five girls? What if all the boys were born first, then all the girls, that is, two runs? We have seen

```
Number of -ve numbers = 5
Number of +ve numbers = 5
Number of runs        = 2
Probability(runs =< observed;
given no. of +ve, -ve numbers)     = 0.00794    Reject H0 at 1% s-level
Critical number. for 1% sig. level = 2
Critical number for 5% sig. level  = 3
Probability(runs =< observed;
given number of non zero numbers)  = 0.01953    Reject H0 at 5% s-level
Probability(signs =< observed)
(Two tail sign test statistic)     = 1.00000
```

Table 10.29: Run test

in table 10.28 that the sign test alone does not help, but table 10.29 would confirm what most would believe intuitively: the event may not represent random sampling but could suggest the operation of other factors. In this way the run test, particularly when conditional upon the number of successes and failures, is using information from the sequence of outcomes and is therefore more powerful than the sign test alone.

The run test can be valuable in the analysis of residuals if there is a natural ordering, for instance, when the residuals are arranged to correspond to the order of a single independent variable. This is not possible if there are replicates, or several independent variables so, to use the run test in such circumstances, the residuals must be arranged in some meaningful sequence, such as the order in time of the observation, otherwise arbitrary results can be obtained by rearranging the order of residuals. Given the numbers of positive and negative residuals, the probability of any possible number of runs can be calculated by enumerating all possible arrangements. For instance, the random number of runs R given m positive and n negative residuals (redefining if necessary so that $m \le n$) depends on whether the number of runs is even or odd as follows

$$
\begin{aligned}
P(R = 2k) &= \frac{2\binom{m-1}{k-1}\binom{n-1}{k-1}}{\binom{m+n}{m}},\\
\text{or } P(R = 2k+1) &= \frac{\binom{m-1}{k-1}\binom{n-1}{k} + \binom{m-1}{k}\binom{n-1}{k-1}}{\binom{m+n}{m}}.
\end{aligned}
$$

Here the maximum number of runs is $2m + 1$ if $m < n$, or $2m$ if $m = n$, and $k = 1, 2, \ldots, m \le n$. However, in the special case that $m > 20$ and $n > 20$, the probabilities of r runs can be estimated by using a normal distribution with

$$
\begin{aligned}
\mu &= \frac{2mn}{m+n} + 1,\\
\sigma^2 &= \frac{2mn(2mn - m - n)}{(m+n)^2(m+n-1)},\\
\text{and } z &= \frac{r - \mu + 0.5}{\sigma},
\end{aligned}
$$

where the usual continuity correction is employed.

The previous conditional probabilities depend on the values of m and n, but it is sometimes useful to know the absolute probability of R runs given $N = n + m$ nonzero residuals. There will always be at least one run, so the probability of r runs occurring depends on the number of ways of choosing break points where a sequence of residuals changes sign. This will be the same as the number of ways of choosing $r-1$ items from $N-1$ without respect to order, divided by the total number of possible configurations, i.e. the probability of $r-1$ successes in $N-1$ independent Bernoulli trials given by

$$P(R=r) = \binom{N-1}{r-1}\left(\frac{1}{2}\right)^{N-1}.$$

This is the value referred to as the probability of runs given the number of nonzero residuals in table 10.29.

10.3.18 The F test for excess variance

This procedure, which is based on the F distribution (page 423), is used when you have fitted two nested models, e.g., polynomials of different degrees, to the same data and wish to use parsimony to see if the extra parameters are justified. You input the weighted sums of squares $WSSQ_1$ for model 1 with m_1 parameters and $WSSQ_2$ for model 2 with m_2 parameters, and the sample size n, when the following test statistic is calculated

$$F(m_2-m_1, n-m_2) = \frac{(WSSQ_1 - WSSQ_2)/(m_2-m_1)}{WSSQ_2/(n-m_2)}.$$

Table 10.30 illustrates how the test is performed. This test for parameter redundancy is also widely used

```
Q1 ((W)SSQ for model 1)  = 12.0
Q2 ((W)SSQ for model 2)  = 10.0
M1 (no. params. model 1) = 2
M2 (no. params. model 2) = 3
NPTS (no. exper. points) = 12
Numerator deg. freedom   = 1
Denominator deg. freedom = 9
F test statistic TS      = 1.8
P(F >= TS)               = 0.2126
P(F =< TS)               = 0.7874
5% upper tail crit. pnt. = 5.117
1% upper tail crit. pnt. = 1.056
Conclusion:
Model 2 is not justified ... Tentatively accept model 1
```

Table 10.30: F test for exess variance

with models that are not linear or nested and, in such circumstances, it must be interpreted with caution as no more than a useful guide. However, as the use of this test is so common, SIMFIT provides a way to store the necessary parameters in an archive file `w_ftests.cfg` after any fitting, so that the results can be recalled retrospectively to assess model validity. Note that, when you select to recover stored values for this test you must ensure that the data are retrieved in the correct order. The best way to ensure this is by using a systematic technique for assigning meaningful titles as the data are stored.

The justification for the F test can be illustrated by successive fitting of polynomials

$$\begin{aligned}
H_0 &: f(x) = \alpha_0 \\
H_1 &: f(x) = \alpha_0 + \alpha_1 x \\
H_2 &: f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \\
&\dots \\
H_k &: f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_k x^k
\end{aligned}$$

in a situation where experimental error is normal with zero mean and constant variance, and the true model is a polynomial, a situation that will never be encountered in real life. The important distributional results, illustrated for the case of two models i and j, with $j > i \geq 0$, so that the number of points and parameters satisfy $n > m_j > m_i$ while the sums of squares are $Q_i > Q_j$, then

1. $(Q_i - Q_j)/\sigma^2$ is $\chi^2(m_j - m_i)$ under model i
2. Q_j/σ^2 is $\chi^2(n - m_j)$ under model j
3. Q_j and $Q_i - Q_j$ are independent under model j.

So the likelihood ratio test statistic

$$F = \frac{(Q_i - Q_j)/(m_j - m_i)}{Q_j/(n - m_j)}$$

is distributed as $F(m_j - m_i, n - m_j)$ if the true model is model i, which is a special case of model j in the nested hierarchy of the polynomial class of linear models.

10.3.19 Nonparametric tests using rstest

When it is not certain that your data are consistent with a known distribution for which special tests have been devised, it is advisable to use nonparametric tests. Many of these are available at appropriate points from **simstat** as follows. Kolmogorov-Smirnov 1-sample (page 132) and 2-sample (page 139), Mann-Whitney U (page 140), Wilcoxon signed ranks (page 142), chi-square (page 143), Cochran Q (page 151), sign (page 152), run (page 153), Kruskall-Wallis (page 165), Friedman (page 169), and nonparametric correlation (page 199) procedures. However, for convenience, program **rstest** should be used, and this also provides further tests as now described.

10.3.20 Runs up or down test for randomness

The runs up test can be conducted on a vector of observations $x_1, x_2, \ldots, x_n$ provided n is fairly large (at least 1000) and there are no ties in the data. The runs down test is done by multiplying the sample by -1 then repeating the runs up test. Table 10.31 illustrates the results from analyzing `g08eaf.tf1`, showing no

```
 Title of data
Data for G08EAF: 1000 U(0,1) pseudo-random numbers
 Size of sample                   = 1000
 CU (chi-sq.stat. for runs up)    = 7.747
 Degrees of freedom               = 4
 P(chi-sq. >= CU) (upper tail p)  = 0.1013
 CD (chi-sq.stat. for runs down)  = 4.640
 Degrees of freedom               = 4
 P(chi-sq. >= CD) (upper tail p)  = 0.3263
```

Table 10.31: Runs up or down test for randomness

evidence against randomness. The number of runs up c_i of length i are calculated for increasing values of i up to a limit $r - 1$, all runs of length greater than $r - 1$ being counted as runs of length r. Then the chi-square statistic

$$\chi^2 = (c - \mu_c)^T \Sigma_c^{-1} (c - \mu_c)$$

with r degrees of freedom is calculated, where

$$\begin{aligned} c &= c_1, c_2, \ldots, c_r, \text{ vector of counts} \\ \mu_c &= e_1, e_2, \ldots, e_r, \text{ vector of expected values} \\ \Sigma_c &= \text{ covariance matrix.} \end{aligned}$$

Note that the default maximum value for r is set by SIMFIT at four, which should be sufficient for many purposes. If the maximum number of runs requested is so large there are empty bins for large values of r which would compromise the chi-squared test, then this maximum value is temporarily reduced to the r level corresponding to the largest observed run, and the degrees of freedom in the chi-square test are reduced accordingly.

Using $p = i - 1$ Levene (Ann. Math. Stat. 23 (1952) 34-56) gives formulas for calculating the expected values for runs c_i of length $i < r$ as

$$n\frac{p^2 + 3p + 1}{(p+3)!} - \frac{p^3 + 3p^2 - p - 4}{(p+3)!}$$

and for runs c_i for i greater than or equal to r as

$$n\frac{p+1}{(p+2)!} - \frac{p^2 + p - 1}{(p+2)!}.$$

Also formulas for the covariance matrix of c_i are given

10.3.21 Median test

The median test examines the difference between the medians of two samples to test the null hypothesis

H_0: the medians are the same,

against the alternative hypothesis that they are different. Table 10.32 presents the results from analyzing

```
 Current data sets X and Y are:
Data for G08ACF: the median test
 Number of X-values = 16
Data for G08ACF: the median test
 Number Y-values = 23
 Results for median test:
 H0: medians are the same
 Number of X-scores below pooled median = 13
 Number of Y-scores below pooled median = 6
 Probability under H0             = 0.0009  Reject H0 at 1% sig.level
```

Table 10.32: Median test

g08acf.tf1 and g08acf.tf2. The test procedure is first to calculate the median for the pooled sample, then form a two by two contingency table 10.3.12 for scores above and below this pooled median. For small samples ($n_1 + n_2 < 40$) a Fisher exact test is used 10.3.12, otherwise a chi-squared test 10.3.11 is used.

For instance, with sample 1 of size n_1 and sample 2 of size n_2 the pooled median M is calculated for the combined sample of size $n = n_1 + n_2$. Then i_1, the number of values less than this pooled median in sample 1, and i_2, the number of values less than this pooled median are calculated. If the medians of these two samples are identical then we could expect i_1 to be about $n_1/2$ and i_2 to be about $n_2/2$ which can be tested using the following 2 by 2 frequency table.

	Sample1	Sample2	Total
Scores $< M$	i_1	i_2	$i_1 + i_2$
Scores $\geq M$	$n_1 - i_1$	$n_2 - i_2$	$n - (i_1 + i_2)$
Total	n_1	n_2	n

10.3.22 Mood's test and David's test for equal dispersion

These are used to test the null hypothesis of equal dispersions, i.e. equal variances. Table 10.33 presents

```
  Current data sets X and Y are:
Data for G08BAF: Mood-David tests for equal dispersions
 Number of X-values = 6
Data for G08BAF: Mood-David tests for equal dispersions
 Number of Y-values = 6
 Results for the Mood test
 H0: dispersions are equal
 H1: X-dispersion > Y-dispersion
 H2: X-dispersion < Y-dispersion
 The Mood test statistic = 75.5
 Probability under H0    = 0.8339
 Probability under H1    = 0.4170
 Probability under H2    = 0.5830
 Results for the David test
 H0: dispersions are equal
 H1: X-dispersion > Y-dispersion
 H2: X-dispersion < Y-dispersion
 The David test statistic = 9.467
 Probability under H0     = 0.3972
 Probability under H1     = 0.8014
 Probability under H2     = 0.1986
```

Table 10.33: Mood-David equal dispersion tests

the results from analyzing g08baf.tf1 and g08baf.tf2. If the two samples are of size n_1 and n_2, so that $n = n_1 + n_2$, then the ranks r_i in the pooled sample are calculated. The two test statistics W and V are defined as follows.

- Mood's test assumes that the two samples have the same mean so that

$$W = \sum_{i=1}^{n_1} \left(r_i - \frac{n+1}{2} \right)^2,$$

  which is the sum of squares of deviations from the average rank in the pooled sample, is approximately normally distributed for large n. The test statistic is

$$z = \frac{W - n_1(n^2 - 1)/12}{\sqrt{n_1 n_2 (n+1)(n^2 - 4)/180}}.$$

- David's test uses the mean rank

$$\bar{r} = \sum_{i=1}^{n_1} r_i / n_1$$

  to reduce the effect of the assumption of equal means in the calculation of

$$V = \frac{1}{n_1 - 1} \sum_{i=1}^{n_1} (r_i - \bar{r})^2,$$

  and V is also approximately normally distributed for large n. The test statistic is

$$z = \frac{V - n(n+1)/12}{\sqrt{n n_2 (n+1)(3(n+1)(n_1+1) - n n_1)/360 n_1 (n_1 - 1)}}.$$

10.3.23 Kendall coefficient of concordance

This test is used to measure the degree of agreement between k comparisons of n objects. Table 10.34 presents

```
HO: no agreement between comparisons
Data title: Data for G08DAF
Number of columns (objects)  = 10
Number of rows (comparisons) = 3
Kendall coefficient W        = 0.8277
P(chi-sq >= W)               = 0.0078  Reject H0 at 1% sig.level
```

Table 10.34: Kendall coefficient of concordance: results

the results from analyzing `g08daf.tf1`, i.e. the data file shown in table 10.35, which illustrates the format

```
Data for G08DAF
    3     10
 1.0  4.5  2.0  4.5   3.0  7.5  6.0  9.0  7.5 10.0
 2.5  1.0  2.5  4.5   4.5  8.0  9.0  6.5 10.0  6.5
 2.0  1.0  4.5  4.5   4.5  4.5  8.0  8.0  8.0 10.0
8
Rows can be comparisons or variables    (i = 1,2,...,k)
Columns can be objects  or observations (j = 1,2,...,n)
If the A(i,j) are ranks of object j in comparison i, then
the A(i,j) must be > 0 and ties must be averages so that
sum of ranks A(i,j) for j = 1,2,...,n must be n(n + 1)/2.
However A can be input as original data, when the rows must
be variables, and the columns must be the observations to
be ranked automatically by the program.
```

Table 10.35: Kendall coefficient of concordance: data pre-ranked

for supplying data for analysis in pre-ranked form. Ranks r_{ij} for the the rank of object j in comparison i (with tied values being given averages) are used to calculate the n column rank sums R_j, which would be approximately equal to the average rank sum $k(n+1)/2$ under

H_0 : there is no agreement.

For total agreement the R_j would have values from some permutation of $k, 2k, \ldots, nk$, and the total squared deviation of these is $k^2n(n^2-1)/12$. Then the coefficient W is calculated according to

$$W = \frac{\sum_{j=1}^{n}(R_j - k(n+1)/2)^2}{k^2n(n^2-1)/12 - k\sum T}$$

which lies between 0 for complete disagreement and 1 for complete agreement. Here the denominator correction for ties uses T defined as

$$T = \sum t(t^2-1)/12$$

where t is the number of occurrences of each tied rank within a comparison. For large samples ($n > 7$), $k(n-1)W$ is approximately χ^2_{n-1} distributed, otherwise tables should be used for accurate significance levels.

Table 10.36 emphasizes that un-ranked data as in test file `kendall.tf1` can be supplied for analysis, when SIMFIT will generate the ranks internally before calculating the test statistic which, in this case leads to $W = 0.9241, p = 0.0013$.

```
Data for Kendall coefficient of concordance
3 12
10.4 10.8 11.1 10.2 10.3 10.2 10.7 10.5 10.8 11.2 10.6 11.4
 7.4  7.6  7.9  7.2  7.4  7.1  7.4  7.2  7.8  7.7  7.8  8.3
17.0 17.0 20.0 14.5 15.5 13.0 19.5 16.0 21.0 20.0 18.0 22.0
7
Columns 1 to 10 are bird species
   Rows 1 to 3  are wing length, tail length, and bill length
From example 19.5 of Zar Biostatistical Analysis 3rd Edn. p437
The following ranks are calculated internally by Simfit
4.0  8.5 10.0  1.5  3.0  1.5  7.0  5.0  8.5 11.0  6.0 12.0
5.0  7.0 11.0  2.5  5.0  1.0  5.0  2.5  9.5  8.0  9.5 12.0
5.5  5.5  9.5  2.0  3.0  1.0  8.0  4.0 11.0  9.5  7.0 12.0
```

Table 10.36: Kendall coefficient of concordance: data un-ranked

Part 11

Analysis of variance

11.1 Introduction

In studying the distribution of the variance estimate from a sample of size n from a normal distribution with mean μ and variance σ^2, you will have encountered the following decomposition of a sum of squares

$$\sum_{i=1}^{n}\left(\frac{y_i-\mu}{\sigma}\right)^2=\sum_{i=1}^{n}\left(\frac{y_i-\bar{y}}{\sigma}\right)^2+\left(\frac{\bar{y}-\mu}{\sigma/\sqrt{n}}\right)^2$$

into independent chi-square variables with $n-1$ and 1 degree of freedom respectively. Analysis of variance is an extension of this procedure based on linear models, assuming normality and constant variance, then partitioning of chi-square variables (page 423) into two or more independent components, invoking Cochran's theorem (page 423) and comparing the ratios to F variables (page 423) with the appropriate degrees of freedom for variance ratio tests. It can be used, for instance, when you have a set of samples (columns vectors) that come from normal distributions with the same variance and wish to test if all the samples have the same mean. Due to the widespread use of this technique, many people use it even though the original data are not normally distributed with the same variance, by applying variance stabilizing transformations (page 163), like the square root with counts, which can sometimes transform non-normal data into transformed data that are approximately normally distributed. An outline of the theory necessary for several widely used designs follows, but you should never make the common mistake of supposing that ANOVA is model free: ANOVA is always based upon data collected as replicates and organized into cells, where it is assumed that all the data are normally distributed with the same variance but with mean values that differ from cell to cell according to an assumed general linear model.

11.2 Variance homogeneity tests (n samples or library file)

It is often stated that ANOVA procedures are relatively insensitive to small departures from normality, but are much more affected by differences in variances between groups so, for that reason, variance-stabilizing transformations are frequently resorted to. Variance homogeneity tests are best done interactively on the data set, so that the effect of transformations on variance stabilizations can be judged before proceeding to ANOVA.

Table 11.1 illustrates analysis of data in the test file `anova1.tf1` for homogeneity of variance, using the Bartlett test, and also the Levene test.
With two normal samples the F test is recommended, and this can be performed routinely in SIMFIT as part of the t test procedure (page 136), which is actually equivalent to 1-way ANOVA when there are only two samples. Where there are n normal samples the Bartlett test is recommended, and this is just the same as the F test when there are two samples of the same size. If there are k groups, with sample size n_i, $\nu_i = n_i - 1$, and sample variances s_i^2, then the pooled variance estimate s_p^2, and parameters B and C can be calculated as

```
Homogeneity of variance test 1: Bartlett
      Transformation = x (untransformed data)
                   B = 0.69
                   C = 1.08
                 B/C = 0.6.3895
                NDOF = 4
P(chi-square >= B/C) = 0.9586
 Upper tail 1% point = 13.277
 Upper tail 5% point = 9.4877

Homogeneity of variance test 2: Levene (median)
     Transformation = x (untransformed data)
                  W = 0.18458
               DOF1 = 4
               DOF2 = 25
          P(F >= W) = 0.9442
Upper tail 1% point = 4.1774
Upper tail 5% point = 2.7587
```

Table 11.1: Bartlett and Levene tests for homogeneity of variance

follows,

$$s_p^2 = \frac{\sum_{i=1}^{k} \nu_i s_i^2}{\sum_{i=1}^{k} \nu_i}$$

$$B = \log(s_p^2) \sum_{i=1}^{k} \nu_i - \sum_{i=1}^{k} \nu_i \log(s_i^2)$$

$$C = 1 + \frac{1}{3(k-1)} \left(\sum_{i=1}^{k} \frac{1}{\nu_i} - \frac{1}{\sum_{i=1}^{k} \nu_i} \right).$$

To test homogeneity of variance, the Bartlett test statistic B/C is approximately chi-square distributed with $k - 1$ degrees of freedom.

When normality cannot be assumed, the Levene test can be performed. If the total number of observations is $N = \sum_{i=1}^{k} n_i$, then the test statistic W is defined as

$$W = \frac{(N-k) \sum_{i=1}^{k} n_i (Z_{i.} - Z_{..})^2}{(k-1) \sum_{i=1}^{k} \sum_{j=1}^{n_i} (Z_{ij} - Z_{i.})^2},$$

where $Z_{..}$ is the mean of all Z_{ij}, and $Z_{i.}$ is the group mean of the Z_{ij}. If Y_{ij} is observation j in group i the definitions are

$$\begin{aligned} Z_{ij} &= |Y_{ij} - Y_{i.}| \\ Z_{..} &= \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n_i} Z_{ij} \\ Z_{i.} &= \frac{1}{n_i}\sum_{j=1}^{n_i} Z_{ij}, \end{aligned}$$

but note that there are several ways to define $Y_{i.}$. Usually this is taken to be the median of group i, but if there are long tails in the distribution as with the Cauchy distribution (page 422), the the trimmed mean can be used (page 288). The group mean can also be used if the data are similar to a normal distribution. To test variance homogeneity, the Levene test statistic W is approximately F distributed with $k-1$ and $N-k$ degrees of freedom.

Table 11.1 illustrates that the null hypothesis

$$H_0 : \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2$$

cannot be rejected for the data in test file `anova1.tf1`.

11.3 Variance stabilizing transformations

A number of transformations are in use that attempt to create new data that is more approximately normally distributed than the original data, or at least has more constant variance, as the two aims can not usually both be achieved. If the distribution of X is known, then the variance of any function of X can of course be calculated. However, to a very crude first approximation, if a random variable X is transformed by $Y = f(X)$, then the variances are related by the differential equation

$$V(Y) \approx \left(\frac{dY}{dX}\right)^2 V(X)$$

which yields $f(.)$ on integration, e.g., if $V(Y) =$ constant is required, given $V(X)$.

Note that SimFIT provides the ability to test the commonly used transformations, to be discussed next, whenever the previous test is used, to make sure that variances are equal before proceeding to the analysis of variance.

11.3.1 Angular transformation

This arcsine transformation is sometimes used for binomial data with parameters N and p, e.g., for X successes in N trials, when

$$\begin{aligned} X &\sim b(N,p) \\ Y &= \arcsin(\sqrt{X/N}) \\ E(Y) &\simeq \arcsin(\sqrt{p}) \\ V(Y) &\simeq 1/(4N) \text{ (using radial measure).} \end{aligned}$$

However, note that the variance of the transformed data is only constant in situations where there are constant binomial denominators.

11.3.2 Square root transformation

This is often used for counts, e.g., for Poisson variables with mean μ, when

$$\begin{aligned} X &\sim \text{Poisson}(\mu)\\ Y &= \sqrt{x}\\ E(Y) &\simeq \sqrt{\mu}\\ V(Y) &\simeq 1/4. \end{aligned}$$

11.3.3 Log transformation

When the variance of X is proportional to a known power α of $E(X)$, then the power transformation $Y = X^{\beta}$ will stabilize variance for $\beta = 1 - \alpha/2$. The angular and square root transformations are, of course, just special cases of this, but a singular case of interest is the constant coefficient of variation situation $V(X) \propto E(X)^2$ which justifies the log transform, as follows

$$\begin{aligned} E(X) &= \mu\\ V(X) &\propto \mu^2\\ Y &= \log X\\ V(Y) &= k, \text{ a constant.} \end{aligned}$$

11.4 1-way and Kruskal-Wallis (n samples or library file)

This procedure is used when you have columns (i.e. samples) of normally distributed measurements with the same variance and wish to test if all the means are equal. With two columns it is equivalent to the two-sample unpaired t test (page 136), so it can be regarded as an extension of this test to cases with more than two columns. Suppose a random variable Y is measured for groups $i = 1, 2, \ldots, k$ and subjects $j = 1, 2, \ldots n_i$, and it is assumed that the appropriate general linear model for the $n = \sum_{i=1}^{k} n_i$ observations is

$$\begin{aligned} y_{ij} &= \mu + \alpha_i + e_{ij}\\ \sum_{i=1}^{k} \alpha_i &= 0 \end{aligned}$$

where the errors e_{ij} are independently normally distributed with zero mean and common variance σ^2.

Then the 1-way ANOVA null hypothesis is

$$H_0 : \alpha_i = 0, \text{ for } i = 1, 2, \ldots, k,$$

that is, the means for all k groups are equal, and the basic equations are as follows.

$$\bar{y}_i = \sum_{j=1}^{n_i} y_{ij}/n_i$$

$$\bar{y} = \sum_{i=1}^{k}\sum_{j=1}^{n_i} y_{ij}/n$$

$$\sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\bar{y})^2 = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\bar{y}_i)^2 + \sum_{i=1}^{k} n_i(\bar{y}_i-\bar{y})^2$$

$$\text{Total } SSQ = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\bar{y})^2, \text{ with } DF = n-1$$

$$\text{Residual } SSQ = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\bar{y}_i)^2, \text{ with } DF = n-k$$

$$\text{Group } SSQ = \sum_{i=1}^{k} n_i(\bar{y}_i-\bar{y})^2, \text{ with } DF = k-1.$$

Here Total SSQ is the overall sum of squares, Group SSQ is the between groups (i.e. among groups) sum of squares, and Residual SSQ is the residual (i.e. within groups, or error) sum of squares. The mean sums of squares and F value can be calculated from these using

$$\begin{aligned}
\text{Total } SSQ &= \text{ Residual } SSQ + \text{ Group } SSQ\\
\text{Total } DF &= \text{ Residual } DF + \text{ Group } DF\\
\text{Group } MS &= \frac{\text{Group } SSQ}{\text{Group } DF}\\
\text{Residual MS } &= \frac{\text{Residual } SSQ}{\text{Residual } DF}\\
F &= \frac{\text{Group } MS}{\text{Residual } MS},
\end{aligned}$$

so that the degrees of freedom for the F variance ratio to test if the between groups MS is significantly larger than the residual MS are $k-1$ and $n-k$. The SimFIT 1-way ANOVA procedure allows you to include or exclude selected groups, i.e., data columns, and to employ variance stabilizing transformations if required, but it also provides a nonparametric test, and it allows you to explore which column or columns differ significantly in the event of the F value leading to a rejection of H_0.

As the assumptions of the linear model will not often be justified, the nonparametric Kruskal-Wallis test can be done at the same time, or as an alternative to the parametric 1-way ANOVA just described. This is in reality an extension of the Mann-Whitney U test (page 140) to k independent samples, which is designed to test H_0: the medians are all equal. The test statistic H is calculated as

$$H = \frac{12}{n(n+1)}\sum_{i=1}^{k}\frac{R_i^2}{n_i} - 3(n+1)$$

where R_i is the sum of the ranks of the n_i observations in group i, and $n = \sum_{i=1}^{k} n_i$. This test is actually a 1-way ANOVA carried out on the ranks of the data. The p value are calculated exactly for small samples, but the fact that H approximately follows a χ^2_{k-1} distribution is used for large samples. If there are ties, then H is corrected by dividing by λ where

$$\lambda = 1 - \frac{\sum_{i=1}^{m}(t_i^3 - t_i)}{n^3 - n}$$

where t_i is the number of tied scores in the ith group of ties, and m is the number of groups of tied ranks. The test is $3/\pi$ times as powerful as the 1-way ANOVA test when the parametric test is justified, but it is more powerful, and should always be used, if the assumptions of the linear normal model are not appropriate. As it is unusual for the sample sizes to be large enough to verify that all the samples are normally distributed and with the same variance, rejection of H_0 in the Kruskal-Wallis test (which is the higher order analogue of the Mann-Whitney U test, just as 1-way ANOVA is the higher analogue of the t test) should always be taken seriously.

To see how these tests work in practise, read in the matrix test file `tukey.tf1` which refers to a data set where the column vectors are the groups, and you will get the results shown in Table 11.2. The null hypothesis,

```
One Way Analysis of Variance: (Grand Mean 4.316E+01)

Transformation:-  x (untransformed data)
Source              SSQ      NDOF    MSQ        F     p
Between Groups       2193.0      4  548.4    56.15  0.0000
Residual              244.1     25  9.765
Total                2438.0     29

Kruskal-Wallis Nonparametric One Way Analysis of Variance

Test statistic   NDOF    p
          23.3      4  0.0001
```

Table 11.2: ANOVA example 1(a): 1-way and the Kruskal-Wallis test

that all the columns are normally distributed with the same mean and variance, would be rejected at the 5% significance level if $p < 0.05$, or at the 1% significance level if $p < 0.01$, which suggests, in this case, that at least one pair of columns differ significantly. Note that each time you do an analysis, the Kruskal-Wallis nonparametric test based on ranks can be done at the same time, or instead of ANOVA. In this case the same conclusion is reached but, of course, it is up to you which result to rely on. Also, you can interactively suppress or restore columns in the data set and you can select variance stabilizing transformations if necessary (page 163). These can automatically divide sample values by 100 if your data are as percentages rather than proportions and a square root, arc sine, logit or similar transformation is called for.

11.5 Tukey Q test (n samples or library file)

This post-ANOVA procedure is used when you have k normal samples with the same variance and 1-way ANOVA suggests that at least one pair of columns differ significantly. For example, after analyzing `tukeyq.tf1` as just described, then selecting the Tukey Q test, Table 11.3 will be displayed. Note that the means are ranked and columns with means between those of extreme columns that differ significantly are not tested, according to the protocol that is recommended for this test. This involves a systematic procedure where the largest mean is compared to the smallest, then the largest mean is compared with the second largest, and so on. If no difference is found between two means then it is concluded that no difference exists between any means enclosed by these two, and so no testing is done. Evidently, for these data, column 5 differs significantly from columns 1, 2, 3, and 4, and column 3 differs significantly from column 1. The test statistic

```
Tukey Q-test with  5 means and 10 comparisons
5% point = 4.189, 1% point = 5.125
Columns     Q          p           5%         1%     NB     NA
  5  1       20.55   0.0001          *          *     6      6
  5  2       14.16   0.0001          *          *     6      6
  5  4       13.48   0.0001          *          *     6      6
  5  3       11.14   0.0001          *          *     6      6
  3  1       9.406   0.0001          *          *     6      6
  3  2       3.018   0.2377         NS         NS     6      6
  3  4      [[2.338  0.4792]] No-Test   No-Test     6      6
  4  1       7.068   0.0005          *          *     6      6
  4  2     [[0.6793  0.9885]] No-Test   No-Test     6      6
  2  1       6.388   0.0013          *          *     6      6
[ 5%] and/or [[ 1%]] No-Test results given for reference only
```

Table 11.3: ANOVA example 1(b): 1-way and the Tukey Q test

Q for comparing columns A and B with sample sizes n_A and n_B is

$$
\begin{aligned}
Q &= \frac{\bar{y}_B - \bar{y}_A}{SE}\\
\text{where } SE &= \sqrt{\frac{s^2}{n}}, \text{ if } n_A = n_B\\
SE &= \sqrt{\frac{s^2}{2}\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}, \text{ if } n_A \neq n_B\\
s^2 &= \text{error } MS
\end{aligned}
$$

and the significance level for Q is calculated as a studentized range.

11.6 Plotting 1-way data

After analyzing a selected subset of data, possibly transformed, it is useful to be able to inspect the columns of data so as to identify obvious differences. This can be done by plotting the selected columns as a scattergram, or by displaying the selected columns as a box and whisker plot, with medians, quartiles and ranges, or as a range and percentiles plot.

Figure 11.1 illustrates a box and whisker plot for the data in `anova1.tfl` where the samples have different sample sizes, so this is best done using such a library file. Clearly, a matrix file cannot be used where the samples have different sample sizes. In this plot the lower limit shows the lowest value, the bottom of the box shows the lower quartile, the mid-bar shows the median, the upper part of the box shows the upper quartile, while the upper limit shows the largest value. Another useful way to display these positions is the range and percentiles plot, as in figure 11.2, where a set of symbols joined by lines indicates the positions of the same percentile points as the box and whisker plot. With numerous samples this plot is easier to interpret than a crowded box and whisker plot.

Alternatively, a bar chart or line and symbol plot can be constructed with the means of selected columns, and with error bars calculated for 95% confidence limits, or as selected multiples of the sample standard errors or sample standard deviations. This option is also provided after doing 1-way ANOVA.

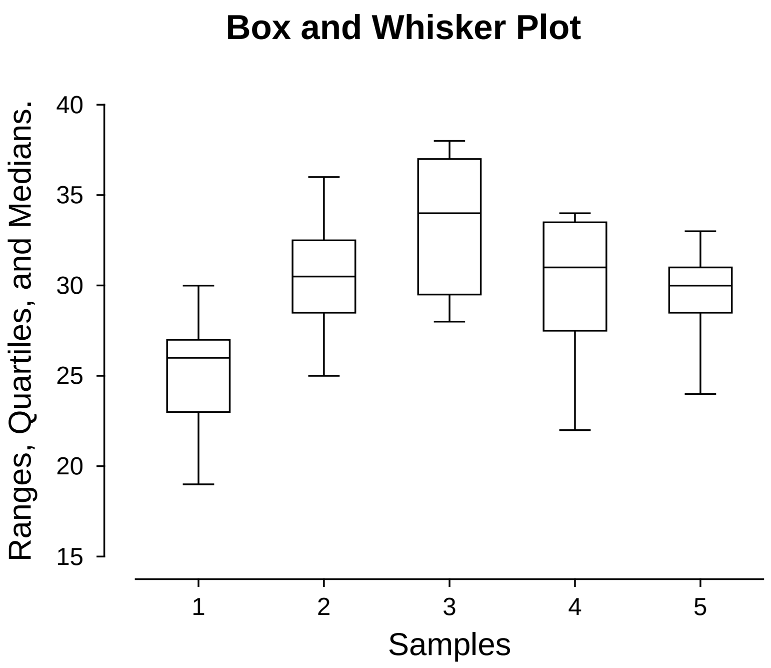


Figure 11.1: Box and whisker plot

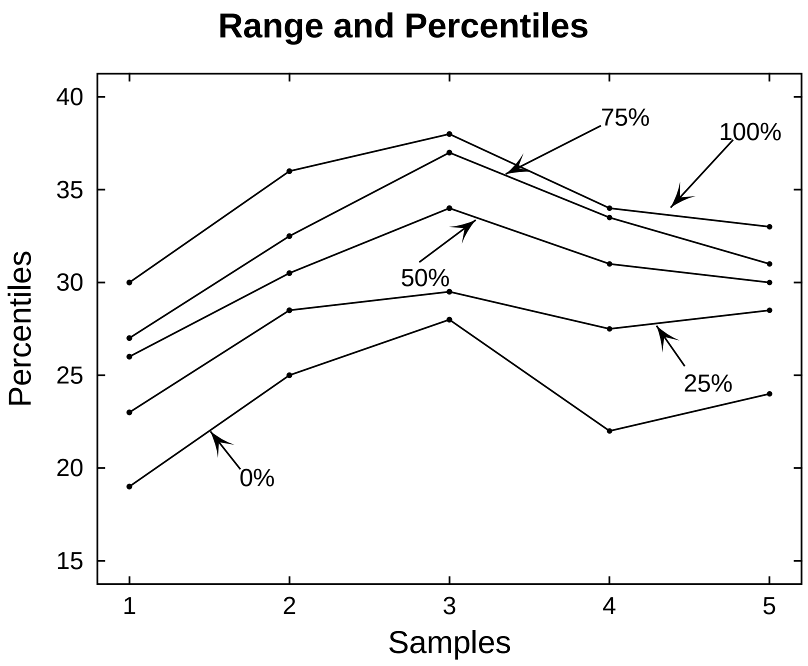


Figure 11.2: Range and percentiles plot

11.7 2-way and the Friedman test (one matrix)

This procedure is used when you want to include row and column effects in a completely randomized design, i.e., assuming no interaction and one replicate per cell so that the appropriate linear model is

$$y_{ij} = \mu + \alpha_i + \beta_j + e_{ij}$$
$$\sum_{i=1}^{r} \alpha_i = 0$$
$$\sum_{j=1}^{c} \beta_j = 0$$

for a data matrix with r rows and c columns, i.e. $n = rc$. The mean sums of squares and degrees of freedom for row and column effects are worked out, then the appropriate F and p values are calculated. Using R_i for the row sums, C_j for the column sums, and $T = \sum_{i=1}^{r} R_i = \sum_{j=1}^{c} C_j$ for the sum of observations, these are

$$\text{Row } SSQ = \sum_{i=1}^{r} R_i^2/c - T^2/n, \text{ with } DF = r - 1$$
$$\text{Column } SSQ = \sum_{j=1}^{c} C_j^2/r - T^2/n, \text{ with } DF = c - 1$$
$$\text{Total } SSQ = \sum_{i=1}^{r} \sum_{j=1}^{c} y_{ij}^2 - T^2/n, \text{ with } DF = n - 1$$
$$\text{Residual } SSQ = \text{Total } SSQ - \text{Row } SSQ - \text{Column } SSQ, \text{ with } DF = (r-1)(c-1)$$

where Row SSQ is the between rows sums of squares, Column SSQ is the between columns sum of squares, Total SSQ is the total sum of squares and Residual SSQ is the residual, or error sum of squares. Now two F statistics can be calculated from the mean sums of squares as

$$F_R = \frac{\text{Rows } MS}{\text{Residual } MS}$$
$$F_C = \frac{\text{Column } MS}{\text{Residual } MS}.$$

The statistic F_R is compared with $F(r - 1, (r - 1)(c - 1))$ to test

$$H_R : \alpha_i = 0, i = 1, 2, \dots, r$$

i.e., absence of row effects, while F_C is compared with $F(c - 1, (r - 1)(c - 1))$ to test

$$H_C : \beta_j = 0, j = 1, 2, \dots, c$$

i.e., absence of column effects.

If the data matrix represents scores etc., rather than normally distributed variables with identical variances, then the matrix can be analyzed as a two way table with k rows and n columns using the nonparametric Friedman 2-way ANOVA procedure, which is an analogue of the sign test (page 152) for multiple matched samples designed to test H_0 : all medians are equal, against the alternative, H_1 : they come from different populations. The procedure ranks column scores as r_{ij} for row i and column j, assigning average ranks for ties, works out rank sums as $t_i = \sum_{j=1}^{l} r_{ij}$, then calculates FR given by

$$FR = \frac{12}{nk(k+1)} \sum_{i=1}^{k} (t_i - n(k+1)/2)^2 .$$

where k is the number of groups and n the number of cases. Note however that SIMFIT test files have the more usual transposed format with k columns of groups and n rows of observations as in `anova2.tf1`. For small samples, exact significance levels are calculated, while for large samples it is assumed that FR follows a χ^2_{k-1} distribution. For practise you should try the test file `anova2.tf1` which is analyzed as shown in Table 11.4. Note that there are now two p values for the two independent significance tests, and observe that, as in the

```
2-Way Analysis of Variance: (Grand mean  2.000E+00)

Source                 SSQ  NDOF       MSSQ          F      p
Between rows           0.0    17        0.0        0.0  1.0000
Between columns      8.583     2      4.292      5.421  0.0090
Residual             26.92    34     0.7917
Total                35.50    53

Friedman Nonparametric Two-Way Analysis of Variance

Test Statistic     = 8.583
Degees of Freedom = 2
Significance       = 0.0137
```

Table 11.4: ANOVA example 2: 2-way and the Friedman test

previous 1-way ANOVA test, the corresponding nonparametric (Friedman) test can be done at the same time, or instead of the parametric test if required.

11.8 3-way and Latin Square design (one matrix)

The linear model for a m by m Latin Square ANOVA is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + e_{ijk}$$
$$\sum_{i=1}^{m} \alpha_i = 0$$
$$\sum_{j=1}^{m} \beta_j = 0$$
$$\sum_{k=1}^{m} \gamma_k = 0$$

where α_i, β_j and γ_k represent the row, column and treatment effect, and e_{ijk} is assumed to be normally distributed with zero mean and variance σ^2. The sum of squares partition is now

$$\text{Total } SSQ = \text{ Row } SSQ + \text{ Column } SSQ + \text{ Treatment } SSQ + \text{ Residual } SSQ$$

where the m^2 observations are arranged in the form of a m by m matrix so that every treatment occurs once in each row and column. This design, which is used for economical reasons to account for row, column, and treatment effects, leads to the three variance ratios

$$F_R = \frac{\text{Row } MS}{\text{Residual } MS}$$
$$F_C = \frac{\text{Column } MS}{\text{Residual } MS}$$
$$F_T = \frac{\text{Treatment } MS}{\text{Residual } MS}$$

to use in F tests with $m - 1$, and $(m - 1)(m - 2)$ degrees of freedom. Note that SIMFIT data files for Latin square designs with m treatment levels have $2m$ rows and m columns, where the first m by m block identifies the treatments, and the next m by m block of data are the observations. When designing such experiments, the particular Latin square used should be chosen randomly if possible as described on page 291. For instance, try the test file anova3.tf1, which should be consulted for details, noting that integers (1, 2, 3, 4, 5) are used instead of the usual letters (A, B, C, D, E) in the data file header to indicate the position of the treatments. Note that, in Table 11.5, there are now three p values for significance testing between rows, columns, and

```
Three Way Analysis of Variance: (Grand mean 7.186E+00)

Source        NDOF     SSQ         MSQ       F       p
Rows             4    29.42       7.356    9.027   0.0013
Columns          4    22.99       5.749    7.055   0.0037
Treatments       4   0.5423      0.1356   0.1664   0.9514
Error           12    9.779      0.8149
Total           24    62.74       2.614
Row means:
 8.136   6.008   8.804   6.428   6.552
Column means:
 5.838   6.322   7.462   7.942   8.364
Treatment means:
 7.318   7.244   7.206     6.9    7.26
```

Table 11.5: ANOVA example 3: 3-way and Latin square design

treatments.

11.9 Groups and subgroups (one matrix)

The linear models for ANOVA are easy to manipulate mathematically and trivial to implement in computer programs, and this has lead to a vast number of possible designs for ANOVA procedures. This situation is likely to bewilder users, and may easily mislead the unwary, as it stretches credulity to the limit to believe that experiments, which almost invariably reflect nonlinear non-normal phenomena, can be analyzed in a meaningful way by such elementary models. Nevertheless, ANOVA remains valuable for preliminary data exploration, or in situations like clinical or agricultural trials, where only gross effects are of interest and precise modelling is out of the question, so a further versatile and flexible ANOVA technique is provided by SIMFIT for two-way hierarchical classification with subgroups of possibly unequal size, assuming a fixed effects model. Suppose, for instance, that there are $k \geq 2$ treatment groups, with group i subdivided into l_i treatment subgroups, where subgroup j contains n_{ij} observations. That is, observation y_{mij} is observation m in subgroup j of group i where

$$1 \leq i \leq k, 1 \leq j \leq l_i, 1 \leq m \leq n_{ij}.$$

The between groups, between subgroups within groups, and residual sums of squares are

$$\begin{aligned}\text{Group } SSQ &= \sum_{i=1}^{k} n_{i.}(\bar{y}_{.i.} - \bar{y}_{...})^2\\ \text{Subgroup } SSQ &= \sum_{i=1}^{k}\sum_{j=1}^{l_i} n_{ij}(\bar{y}_{.ij} - \bar{y}_{.i.})^2\\ \text{Residual } SSQ &= \sum_{i=1}^{i}\sum_{j=1}^{l_i}\sum_{m=1}^{n_{ij}} (y_{mij} - \bar{y}_{.ij})^2\end{aligned}$$

which, using $l = \sum_{i=1}^{k} l_i$ and $n = \sum_{i=1}^{k} n_{i.}$, and normalizing give the variance ratios

$$\begin{aligned}F_G &= \frac{\text{Group } SSQ/(k-1)}{\text{Residual } SSQ/(n-l)}\\ F_S &= \frac{\text{Subgroup } SSQ/(l-k)}{\text{Residual } SSQ/(n-l)}\end{aligned}$$

to test for between groups and between subgroups effects. To practise, an appropriate test file is anova4.tf1, which should be consulted for details, and the results are shown in table 11.6. Of course, there are now

```
Groups/Subgroups 2-Way ANOVA

Transformation = x (untransformed data)
Source              SSQ      NDOF    F       p
Between Groups       0.4748     1  16.15  0.0007
Subgroups            0.8162     6  4.626  0.0047
Residual             0.5587    19
Total                  1.85    26

Group  Subgroup     Mean
    1         1      2.1
    1         2    2.233
    1         3      2.4
    1         4    2.433
    1         5      1.8
    2         1    1.867
    2         2     1.86
    2         3    2.133
Group 1 mean = 2.206 (16 Observations)
Group 2 mean = 1.936 (11 Observations)
Grand   mean = 2.096 (27 Observations)
```

Table 11.6: ANOVA example 4: arbitrary groups and subgroups

two p values for significance testing and, also note that, because this technique allows for many designs that cannot be represented by rectangular matrices, the data files must have three columns and n rows: column one contains the group numbers, column two contains the subgroup numbers, and column three contains the observations as a vector in the order of groups and subgroups within groups. By defining groups and subgroups correctly a large number of ANOVA techniques can be done using this procedure.

11.10 Factorial design (one matrix)

Factorial ANOVA is employed when two or more factors are used together at more than one level, possibly with blocking, and the technique is best illustrated by a simple example. For instance, table 11.7 shows the results from analyzing data in the test file `anova5.tf1`. which has two factors, A and B say, but no blocking.

```
Factorial ANOVA

Transformation = x (untransformed data)
Source              SSQ  NDOF      MS        F      p
Blocks              0.0     0     0.0      0.0  0.0000
Effect 1 (A)     1386.0     1  1386.0    60.53  0.0000
Effect 2 (B)      70.31     1   70.31    3.071  0.0989
Effect 3 (A*B)      4.9     1     4.9   0.2140  0.6499
Residual          366.4    16    22.9
Total            1828.0    19

Overall mean
 21.82
Treatment means
Effect  1
 13.5  30.15
Std.Err. of difference in means = 2.14
Effect  2
 23.7  19.95
Std.Err. of difference in means = 2.14
Effect  3
 14.88  12.12  32.52  27.78
Std.Err. of difference in means = 3.026
```

Table 11.7: ANOVA example 5: factorial design

The appropriate linear model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + e_{ijk}$$

where there are a levels of factor A, b levels of factor B and n replicates per cell, that is, n observations at each fixed pair of i and j values. As usual, μ is the mean, α_i is the effect of A at level i, β_j is the effect of B at level j, $(\alpha\beta)_{ij}$ is the effect of the interaction between A and B at levels i and j, and e_{ijk} is the random error component at replicate k. Also there are the necessary constraints that $\sum_{i=1}^{a}\alpha_i = 0$, $\sum_{j=1}^{b}\beta_j = 0$, $\sum_{i=1}^{a}(\alpha\beta)_{ij} = 0$, and $\sum_{j=1}^{b}(\alpha\beta)_{ij} = 0$. The null hypotheses would be

$$H_0 : \alpha_i = 0, \text{ for } i = 1, 2, \ldots, a$$

to test for the effects of factor A,

$$H_0 : \beta_j = 0, \text{ for } j = 1, 2, \ldots, b$$

to test for the effects of factor B, and

$$H_0 : (\alpha\beta)_{ij} = 0, \text{ for all } i, j$$

to test for possible AB interactions. The analysis of variance table is based upon calculating F statistics as ratios of sums of squares that arise from the partitioning of the total corrected sum of squares as follows

$$
\begin{aligned}
\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(y_{ijk}-\bar{y}_{...})^2 &= \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}[(\bar{y}_{i..}-\bar{y}_{...})+(\bar{y}_{.j.}-\bar{y}_{...})\\
&\quad +(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y}_{...})+(y_{ijk}-\bar{y}_{ij.})]^2\\
&= bn\sum_{i=1}^{a}(\bar{y}_{i..}-\bar{y}_{...})^2+an\sum_{j=1}^{b}(\bar{y}_{.j.}-\bar{y}_{...})^2\\
&\quad +n\sum_{i=1}^{a}\sum_{j=1}^{b}(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y}_{...})^2+\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}(\bar{y}_{ijk}-\bar{y}_{ij.})^2
\end{aligned}
$$

It is clear from the F statistics and significance levels p in table 11.7 that, with these data, A has a large effect, B has a small effect, and there is no significant interaction. Figure 11.3 illustrates a graphical

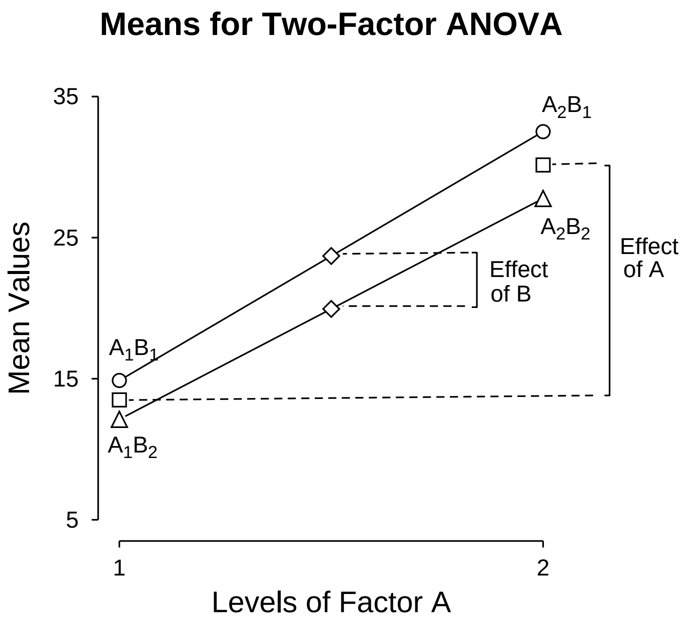


Figure 11.3: Plotting interactions in Factorial ANOVA

technique for studying interactions in factorial ANOVA that can be very useful with limited data sets, say with only two factors. First of all, note that the factorial ANOVA table outputs results in standard order, e.g. $A_1B_1, A_1B_2, A_2B_1, A_2B_2$ and so on, while the actual coefficients $\alpha_i, \beta_j, (\alpha\beta)_{ij}$ in the model can be estimated by subtracting the grand mean from the corresponding treatment means. In the marginals plot, the line connecting the circles is for observations with B at level 1 and the line connecting the triangles is for observations with B at level 2. The squares are the overall means of observations with factor A at level 1 (13.5) and level 2 (30.15), while the diamonds are the overall means of observations with factor B (i.e. 23.7 and 19.95) from table 11.7. Parallel lines indicate the lack of interaction between factors A and B while the larger shift for variation in A as opposed to the much smaller effect of changes in levels of B merely reinforces the conclusions reached previously from the p values in table 11.7. If the data set contains blocking, as with test files `anova5.tf2` and `anova5.tf4`, then there will be extra information in the ANOVA table corresponding to the blocks, e.g., to replace the values shown as zero in table 11.7 as there is no blocking with the data in `anova5.tf1`.

11.11 Repeated measures (one matrix)

This procedure is used when you have paired measurements, and wish to test for absence of treatment effects. With two samples it is equivalent to the two-sample paired t test (page 138), so it can be regarded as an extension of this test to cases with more than two columns. If the rows of a data matrix represent the effects of different column-wise treatments on the same subjects, so that the values are serially correlated, and it is wished to test for significant treatment effects irrespective of differences between subjects, then repeated-measurements design is appropriate. The simplest, model-free, approach is to treat this as a special case of 2-way ANOVA where only between-column effects are considered and between-row effects, i.e., between subject variances, are expected to be appreciable, but are not considered. Many further specialized techniques are also possible, when it is reasonable to attempt to model the treatment effects, e.g., when the columns represent observations in sequence of, say, time or drug concentration, but often such effects are best fitted by nonlinear rather than linear models. A useful way to visualize repeated-measurements ANOVA data with small samples ($\leq$ 12 subjects) is to input the matrix into the exhaustive analysis of a matrix procedure and plot the matrix with rows identified by different symbols. Table 11.8 shows the results from analyzing data in the test file `anova6.tf1` which consists of three sections, a Mauchly sphericity test, the ANOVA table, and a Hotelling T^2 test, all of which will now be discussed.

In order for the normal two-way univariate ANOVA to be appropriate, sphericity of the covariance matrix of orthonormal contrasts is required. The test is based on a orthonormal contrast matrix, for example a Helmert matrix of the form

$$C = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 & 0 & \dots \\ 1/\sqrt{6} & 1/\sqrt{6} & -2/\sqrt{6} & 0 & 0 & \dots \\ 1/\sqrt{12} & 1/\sqrt{12} & 1/\sqrt{12} & -3/\sqrt{12} & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

which, for m columns, has dimensions $m-1$ by m, and where every row sum is zero, every row has length unity, and all the rows are orthogonal. Such Helmert conrasts compare each successive column mean with the average of the preceding (or following) column means but, in the subsequent discussion, any orthonormal contrast matrix leads to the same end result, namely, when the covariance matrix of orthonormal contrasts satisfies the sphericity condition, then the sums of squares used to construct the F test statistics will be independent chi-square variables and the two-way univariate ANOVA technique will be the most powerful technique to test for equality of column means. The sphericity test uses the sample covariance matrix S to construct the Mauchly W statistic given by

$$W = \frac{|CSC^T|}{[Tr(CSC^T)/(m-1)]^{m-1}}.$$

If S is estimated with ν degrees of freedom then

$$\chi^2 = -\left[\nu - \frac{2m^2-3m+3}{6(m-1)}\right]\log W$$

is approximately distributed as chi-square with $m(m-1)/2-1$ degrees of freedom. Clearly, the results in table 11.8 show that the hypothesis of sphericity cannot be rejected, and the results from two-way ANOVA can be tentatively accepted. However, in some instances, it may be necessary to alter the degrees of freedom for the F statistics as discussed next.

The model for univariate repeated measures with m treatments used once on each of n subjects is a mixed model of the form

$$y_{ij} = \mu + \tau_i + \beta_j + e_{ij},$$

where τ_i is the fixed effect of treatment i so that $\sum_{i=1}^m \tau_i = 0$, and β_j is the random effect of subject j with mean zero, and $\sum_{j=1}^n \beta_j = 0$. Hence the decomposition of the sum of squares is

$$\sum_{i=1}^m \sum_{j=1}^n (y_{ij} - \bar{y}_{.j})^2 = n\sum_{i=1}^m (\bar{y}_{i.} - \bar{y}_{..})^2 + \sum_{i=1}^m \sum_{j=1}^n (y_{ij} - \bar{y}_{i.} - \bar{y}_{.j} + \bar{y}_{..})^2,$$

```
Sphericity test on CV of Helmert orthonormal contrasts

H0: Covariance matrix = k*Identity (for some k > 0)

Number of small eigenvalues = 0 (i.e. < 0.0000001)
Number of variables (m)     = 4
Sample size (n)             = 5
Determinant of CV           = 154.9
Trace of CV                 = 28.2
Mauchly W statistic         = 0.1865
LRTS (-2*log(lambda))       = 4.572
Degrees of Freedom          = 5
p = P(chi-square >= LRTS)   = 0.4704
e (Geisser-Greenhouse)      = 0.6049
e (Huynh-Feldt)             = 1.0
e (lower bound)             = 0.3333

Repeat-measures ANOVA: (Grand mean  2.490E+01)

Source        SSQ    NDOF   MSSQ       F    p
Subjects    680.8       4
Treatments  698.2       3  232.7   24.76  0.0000
                                          0.0006 (Greenhouse-Geisser)
                                          0.0000 (Huyhn-Feldt)
                                          0.0076 (Lower-bound)
Remainder   112.8      12    9.4
Total      1492.0      19

Friedman Nonparametric Two-Way Analysis of Variance

Test Statistic    = 13.56
Degees of Freedom = 3
Significance      = 0.0036

Hotelling one sample T-square test

H0: Column means are all equal

Number of rows = 5, Number of columns = 4
Hotelling T-square         = 170.5
F Statistic (FTS)          = 28.41
Degrees of Freedom (d1,d2) = 3, 2
p = P(F(d1,d2) >= FTS)     = 0.0342  Reject H0 at 5% sig.level
```

Table 11.8: ANOVA example 6: repeated measures

that is

$$SS_{\text{Within subjects}} = SS_{\text{treatments}} + SS_{\text{Error}}$$

with degrees of freedom

$$n(m-1) = (m-1) + (m-1)(n-1).$$

To test the hypothesis of no treatment effect, that is

$$H_0 : \tau_i = 0 \text{ for } i = 1, 2, \ldots, m,$$

the appropriate test statistic would be

$$F = \frac{SS_{\text{treatment}}/(m-1)}{SS_{\text{Error}}/[(m-1)(n-1)]}$$

but, to make this test more robust, it may be necessary to adjust the degrees of freedom when calculating critical levels. In fact the degrees of freedom should be taken as

$$\begin{aligned} \text{Numerator degrees of freedom } &= \epsilon(m-1) \\ \text{Denominator degrees of freedom } &= \epsilon(m-1)(n-1) \end{aligned}$$

where there are four possibilities for the correction factor ϵ, all with $0 \le \epsilon \le 1$.

1. The default epsilon.
   This is $\epsilon = 1$, which is the correct choice if the sphericity criterion is met.
2. The Greenhouse-Geisser epsilon.
   This is
   $$\epsilon = \frac{(\sum_{i=1}^{m-1} \lambda_i)^2}{(m-1)\sum_{i=1}^{m-1} \lambda_i^2}$$
   where λ_i are the eigenvalues of the covariance matrix of orthonormal contrasts, and it could be used if the sphericity criterion is not met, although some argue that it is an ultraconservative estimate.
3. The Huyhn-Feldt epsilon.
   This is can also be used when the sphericity criterion is not met, and it is constructed from the Greenhouse-Geisser estimate $\hat{\epsilon}$ as follows
   $$\begin{aligned} a &= n(m-1)\hat{\epsilon} - 2 \\ b &= (m-1)(n - G - (m-1)\hat{\epsilon}) \\ \epsilon &= \min(1, a/b), \end{aligned}$$
   where G is the number of groups. It is generally recommended to use this estimate if the ANOVA probabilities given by the various adjustments differ appreciably.
4. The lower bound epsilon.
   This is defined as
   $$\epsilon = 1/(m-1)$$
   which is the smallest value and results in using the F statistic with 1 and $n-1$ degrees of freedom.

If the sphericity criterion is not met, then it is possible to use multivariate techniques such as MANOVA as long as $n > m$, as these do not require sphericity, but these will always be less powerful than the univariate ANOVA just discussed. One possibility is to use the Hotelling T^2 test to see if the column means differ significantly, and the results displayed in table 11.8 were obtained in this way. Again a matrix C of orthonormal contrasts is used together with the vector of column means

$$\bar{y} = (\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_m)^T$$

to construct the statistic

$$T^2 = n(C\bar{y})^T(CSC^T)^{-1}(C\bar{y})$$

since

$$\frac{(n-m+1)T^2}{(n-1)(m-1)} \sim F(m-1, n-m+1)$$

if all column means are equal.

Part 12

Analysis of proportions

12.1 Introduction

Suppose that a total of N observations can be classified into k categories with frequencies consisting of y_i observations in category i, so that $0 \le y_i \le N$ and $\sum_{i=1}^{k} y_i = N$, then there are k proportions defined as

$$p_i = y_i/N,$$

of which only $k-1$ are independent due to the fact that

$$\sum_{i=1}^{k} p_i = 1.$$

If these proportions are then interpreted as estimates of the multinomial probabilities (page 418) and it is wished to make inferences about these probabilities, then we are in a situation that loosely can be described as analysis of proportions, or analysis of categorical data. Since the observations are integer counts and not measurements, they are not normally distributed, so techniques like ANOVA should not be used and specialized methods to analyze frequencies must be employed.

12.1.1 Dichotomous data

If there only two categories, such as success or failure, male or female, dead or alive, etc., the data are referred to as dichotomous, and there is only one parameter to consider. So the analysis of two-category data is based on the binomial distribution (page 417) which is required when y successes have been recorded in N trials and it is wished to explore possible variations in the binomial parameter estimate

$$\hat{p} = y/N,$$

and its unsymmetrical confidence limits (see page 285), possibly as ordered by an indexing parameter x. The SimFIT analysis of proportions procedure accepts a matrix of such y, N data then calculates the binomial parameters and derived parameters such as the Odds

$$\text{Odds } = \hat{p}/(1-\hat{p}), \text{ where } 0 < \hat{p} < 1,$$

and $\log(\text{Odds})$, along with standard errors and confidence limits. It also does a chi-square contingency table test and a likelihood ratio test for common binomial parameters. Sometimes the proportions of successes in sample groups are in arbitrary order, but sometimes an actual indexing parameter is required, as when proportions in the same groups are evolving in time. As an example, read in `binomial.tf3`, which has (y, N, x) data, to see how a parameter x is added equal to the order in the data file. It will be seen from the results in table 12.1 that confidence limits are calculated for the parameter estimates and for the differences between parameter estimates, giving some idea which parameters differ significantly when compared independently. Logs of

```
To test H0: equal binomial p-values
Sample-size                     = 5
Overall sum of Y                = 202
Overall sum of N                = 458
Overall estimate of binomial p  = 0.4410
Lower 95% con. limit            = 0.3950
Upper 95% con. limit            = 0.4879
-2 log lambda (-2LL)            = 118.3, NDOF = 4
p = P(chi-sq. >= -2LL)          = 0.0000 Reject H0 at 1% s-level
Chi-sq. test stat (C)           = 112.9, NDOF = 4
p = P(chi-sq. >= C)             = 0.0000 Reject H0 at 1% s-level

      y      N  lower-95%   p-hat   upper-95%
     23     84   0.18214  0.27381   0.38201
     12     78   0.08210  0.15385   0.25332
     31    111   0.19829  0.27928   0.37241
     65     92   0.60242  0.70652   0.79688
     71     93   0.66404  0.76344   0.84542

d(i,j) = p_hat(i) - p_hat(j), NNT = 1/|d(i,j)|
i,j lower-95% d(i,j)  upper-95% Result p_sig. Var(d(i,j))  NNT (95%cl)
1,2 -0.00455  0.11996  0.24448 Not sig 0.0590  0.00404     8 (*.*) NS
1,3 -0.13219 -0.00547  0.12125 Not sig 0.9326  0.00418   183 (*.*) NS
1,4 -0.56595 -0.43271 -0.29948 (1)<(4) 0.0000  0.00462     2 (1,3) NNH
1,5 -0.61829 -0.48963 -0.36097 (1)<(5) 0.0000  0.00431     2 (1,3) NNH
2,3 -0.24109 -0.12543 -0.00977 (2)<(3) 0.0335  0.00348     8 (4,102) NNH
2,4 -0.67543 -0.55268 -0.42992 (2)<(4) 0.0000  0.00392     2 (1,3) NNH
2,5 -0.72737 -0.60959 -0.49182 (2)<(5) 0.0000  0.00361     2 (1,3) NNH
3,4 -0.55224 -0.42724 -0.30225 (3)<(4) 0.0000  0.00407     2 (1,3) NNH
3,5 -0.60427 -0.48416 -0.36405 (3)<(5) 0.0000  0.00376     2 (1,3) NNH
4,5 -0.18387 -0.05692  0.07004 Not sig 0.3795  0.00420    18 (*.*) NS
p_sig. = significance level
NNH = Number needed to harm (i)<(j), NS = Not sig, NC = Not calculated
```

Table 12.1: Analysis of proportions: dichotomous data

differences and odds with confidence limits can also be tabulated. You could then read in `binomial.tf2` as an example of (y, N) data, to see what to do if a parameter has to be set. Note that tests are done on the data without referencing the indexing parameter x, but plotting the estimates of proportions with confidence limits depends upon the parameter x, if only for spacing out. Experiment with the various ways of plotting the proportions to detect significant differences visually, as when confidence limits do not overlap, indicating statistically significant differences. Figure 12.1 shows the data from `binomial.tf2` plotted as binomial parameter estimates with the overall 95% confidence limits, along with the same data in Log-Odds format, obtained by transferring the data as $Y = p/(1 - p)$ and $X = x$ directly from the Log-Odds plot into the advanced graphics option then choosing the reverse semi-log transformation, i.e., where x is mapped to y and $\log y$ is mapped to x. Note that the error bars are exact in all plots and are therefore unsymmetrical.
Observe that, in figure 12.1, estimates are both above and below the overall mean (solid circle) and overall 95% confidence limits (dotted lines), indicating a significant partitioning into two groups, while the same conclusion can be reached by observing the error bar overlaps in the Log-Odds plot. If it is suspected that the parameter estimate is varying as a function of x or several independent variables, then logistic regression using the GLM option can be used. For further details of the error bars and more advanced plotting see

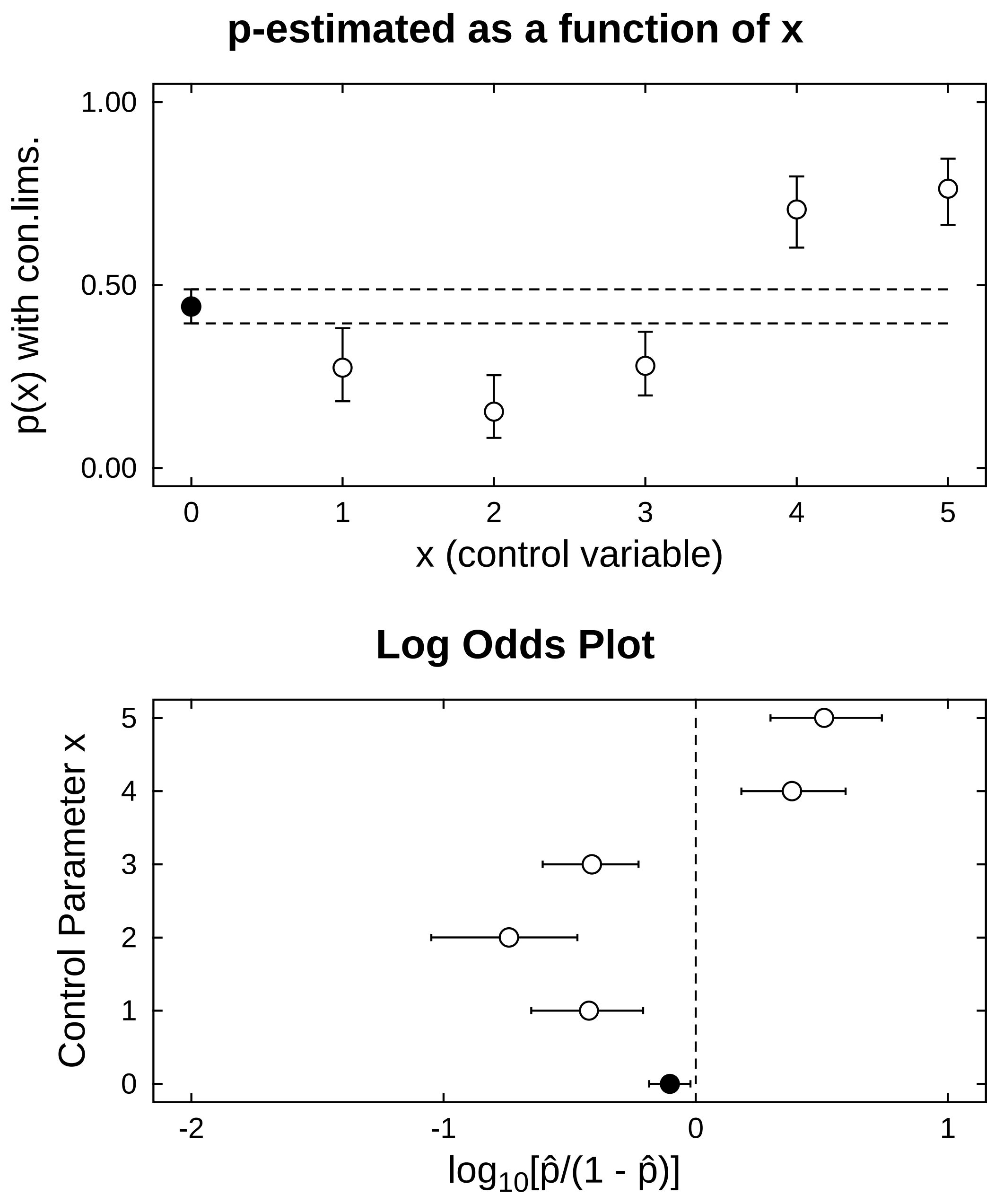


Figure 12.1: Plotting analysis of proportions data

page 188

12.1.2 Binomial parameter confidence limits

It is obvious that a binomial parameter estimate $\hat{p} = y/N$ for the true population parameter p must satisfy

$$0 \le \hat{p} \le 1$$

and so the confidence limits should also be constrained to this range. Hence any accurate confidence limits cannot be symmetrical but must be skewed and so, when a binomial parameter is estimated, it is not possible to report the result in the usual way as $\hat{p} \pm \hat{s}$, or as $\hat{p}(\hat{p} - \hat{s}, \hat{p} + \hat{s})$, where $\hat{s}$ is estimated from the sample and percentiles of a standard normal distribution. Nevertheless, many users of computer packages do not understand this and prefer an approximate expression using the normal distribution because, as long as the sample is large and $p \approx 0.5$, a binomial distribution can be approximated by a normal distribution. For that reason a large sample 95% approximate central confidence range for the true population parameter p is often constructed using

$$\tilde{p} - \tilde{s} \le p \le \tilde{p} + \tilde{s}, \text{ where } \tilde{s} = Z_{\alpha/2}\sqrt{\tilde{p}(1 - \tilde{p})/\tilde{N}}$$

with $\tilde{N} = N + 4$, and $\tilde{p} = (y + 2)/\tilde{N}$.

It is clear that for large samples with $y \approx N/2$ the normal approximation will be adequate but, in order to check the closeness of the approximate limits to the exact ones in any given case, SimFIT provides tables to check the values. For instance, analysis of the test file `binomial.tf4` yields the following comparison.

$\hat{p} = (y/N)$ with exact unsymmetrical small sample limits

y	N	Lower-95%	$\hat{p}$	Upper-95%
23	84	0.182144	0.273810	0.382008
12	78	0.082102	0.153846	0.253321
31	111	0.198289	0.279279	0.372414
91	92	0.940922	0.989130	0.999725
1	93	0.000272	0.010753	0.058458

$\tilde{p} = (y + 2)/(N + 4)$ with approximate central limits $[\tilde{p} \pm \tilde{s}]$

y	N	Lower-95%	$\tilde{p}$	Upper-95%	$\tilde{s}$
23	84	0.189866	0.284091	0.378316	0.094225
12	78	0.089290	0.170732	0.252173	0.081442
31	111	0.204283	0.286957	0.369630	0.082673
91	92	0.933945	0.968750	1.003555	0.034805 ***
1	93	-0.003524	0.030928	0.065380	0.034452 ***

*** Indicates parameter limits outside range (0,1)

The column $\hat{s}$ indicates the amount $\hat{s}$ added to and subtracted from $\hat{p}$ to derive the limits so that the results can be reported as $\hat{p} \pm \hat{s}$. It will be seen that modifying the data in test file `binomial.tf3` to make test file `binomial.tf4` by editing in a couple of extreme values causes the approximate method to overflow or underflow as indicated by ***. Actually the numerical calculation to estimate the exact confidence takes much longer than estimation of the normal approximation, so SimFIT allows users to choose the method to use when analyzing large samples.

12.1.3 Differences between binomial parameter estimates

For cases where the number of samples is relatively small, it is also sometimes helpful to examine tables that highlight significant differences between estimates as follows, using the test file `binomial.tf4`.

$d(i,j) = \hat{p}_i - \hat{p}_j, \quad NNT = 1/|d(i,j)|$

i	j	Lower-95%	$d(i,j)$	Upper-95%	Result	p_sig.	$Var(d(i,j))$	NNT	(95%cl)
1	2	-0.00455	0.11996	0.24448	Not significant	0.0590	0.00404	8	(*.*) NS
1	3	-0.13219	-0.00547	0.12125	Not significant	0.9326	0.00418	183	(*.*) NS
1	4	-0.81300	-0.71532	-0.61764	(1) < (4)	0.0000	0.00248	1	(*.*) NC
1	5	0.16542	0.26306	0.36069	(1) > (5)	0.0000	0.00248	4	(3,6)
2	3	-0.24109	-0.12543	-0.00977	(2) < (3)	0.0355	0.00348	8	(4,102) NNH
2	4	-0.91811	-0.83528	-0.75246	(2) < (4)	0.0000	0.00179	1	(*.*) NC
2	5	0.06033	0.14309	0.22586	(2) > (5)	0.0007	0.00178	7	(4,17) NC
3	4	-0.79596	-0.70985	-0.62374	(3) < (4)	0.0000	0.00193	1	(*.*) NC
3	5	0.18247	0.26853	0.35458	(3) > (5)	0.0000	0.00193	4	(3,5)
4	5	0.94857	0.97838	1.00818	(4) > (5)	0.0000	0.00023	1	(*.*) NC

p_sig. = significance, NNH = Number needed to harm, NS = Not significant, NC = Not calculated

Note that when the lower limit is negative and the upper limit is positive the confidence range includes zero so that the difference between estimates is not significantly different from zero. When the parameters are listed as different, the result can be interpreted as stricter (since $\alpha/2$ is used) than a one–sided lower tail or upper tail test (where α would normally be used). A purist would argue that, as three tests are being done on the same data, the Bonferroni principle would require that significance levels should be divided by three anyway.

It should be noted that the number needed to treat NNT is simply the reciprocal of the absolute difference $d(i,j) = p_i - p_j$, except that, to avoid overflow, this is constrained to the range $1 \le NNT \le 10^6$. Where confidence limits for NNT cannot be estimated, this is indicated by NC, and when the probability difference is negative the number needed to harm is indicated by NNH instead of the confidence range, as will be seen in the above table.

Often it is required to calculate pairwise differences between adjacent lines of the data file, this will happen automatically if the sample size exceeds a certain limiting size. For instance a sample size of size N requires $\binom{N}{2}$ lines of table to output the results from all pairwise comparisons whereas restricting analysis to adjacent pairs only requires $N/2$. Here, for instance, is the result from analyzing the test file `binomial.tf5` where there are twenty lines of data requiring a table with only ten rows.

$d(i,j) = \hat{p}_i - \hat{p}_j, \quad NNT = 1/|d(i,j)|$

i	j	Lower-95%	$d(i,j)$	Upper-95%	Result	p_sig.	$Var(d(i,j))$	NNT	(95%cl)
1	2	0.06707	0.20000	0.33293	(1) > (2)	0.0032	0.00460	5	(3,15)
3	4	0.01315	0.15000	0.28685	(3) > (4)	0.0317	0.00488	7	(3,76)
5	6	0.06707	0.20000	0.33293	(5) > (6)	0.0032	0.00460	5	(3,15)
7	8	0.07604	0.20000	0.32396	(7) > (8)	0.0016	0.00400	5	(3,13)
9	10	0.08322	0.20000	0.31678	(9) > (10)	0.0008	0.00355	5	(3,12)
11	12	0.11684	0.21000	0.30316	(11) > (12)	0.0000	0.00226	5	(3,9)
13	14	-0.23118	-0.14000	-0.04882	(13) < (14)	0.0026	0.00216	7	(4,20) NNH
15	16	-0.46386	-0.35000	-0.23614	(15) < (16)	0.0000	0.00338	3	(2,4) NNH
17	18	-0.15509	-0.05000	0.05509	Not sig.	0.3511	0.00288	20	(*.*) NS
19	20	-0.12002	0.00000	0.12002	Not sig.	1.0000	0.00375	>999999	(*.*) NS

p_sig. = significance, NNH = Number needed to harm, NS = Not significant, NC = Not calculated

12.1.4 Confidence limits for analysis of two proportions

Given two proportions p_i and p_j estimated as

$$\hat{p}_i = y_i/N_i$$
$$\hat{p}_j = y_j/N_j$$

it is often wished to estimate confidence limits for the relative risk RR_{ij}, the difference between proportions DP_{ij}, and the odds ratio OR_{ij}, defined as

$$\begin{aligned}
RR_{ij} &= \hat{p}_i/\hat{p}_j \\
DP_{ij} &= \hat{p}_i - \hat{p}_j \\
OR_{ij} &= \hat{p}_i(1-\hat{p}_j)/[\hat{p}_j(1-\hat{p}_i)].
\end{aligned}$$

First of all note that, for small proportions, the odds ratios and relative risks are similar in magnitude. Then it should be recognized that, unlike the case of single proportions, exact confidence limits can not easily be estimated. However, approximate central $100(1-\alpha)\%$ confidence limits can be calculated using

$$\begin{aligned}
\log(RR_{ij}) &\pm Z_{\alpha/2}\sqrt{\frac{1-\hat{p}_i}{N_i\hat{p}_i} + \frac{1-\hat{p}_j}{N_j\hat{p}_j}} \\
DP_{ij} &\pm Z_{\alpha/2}\sqrt{\frac{\hat{p}_i(1-\hat{p}_i)}{N_i} + \frac{\hat{p}_j(1-\hat{p}_j)}{N_j}} \\
\log(OR_{ij}) &\pm Z_{\alpha/2}\sqrt{\frac{1}{y_i} + \frac{1}{N_i-y_i} + \frac{1}{y_j} + \frac{1}{N_j-y_j}}
\end{aligned}$$

provided $\hat{p}_i$ and $\hat{p}_j$ are not too close to 0 or 1. Here $Z_{\alpha/2}$ is the upper $100(1-\alpha/2)$ percentage point for the standard normal distribution, and confidence limits for RR_{ij} and OR_{ij} can be obtained using the exponential function.

To further clarify table 12.1 it should be pointed out that the confidence limits for individual probabilities are exact and unsymmetrical, as explained on page 285, but for pairwise comparisons the large sample normal approximations are employed. If the confidence regions estimated by this procedure include zero the differences are reported as not significant, otherwise the relative magnitudes of the pair are indicated. As elsewhere in SIMFIT the significance level can be set by the user. Also it should be noted that the number needed to treat (NNT)is simply the reciprocal of the absolute value of the difference $d(i, j) = p_i - p_j$ except that, to avoid overflow, this is constrained to the range $1 \le NNT \le 10^6$.

12.2 Meta analysis

A pair of success/failure classifications with y successes in N trials, i.e. with frequencies $n_{11} = y_1$, $n_{12} = N_1 - y_1$, $n_{21} = y_2$, and $n_{22} = N_2 - y_2$, results in a 2 by 2 contingency table, and meta analysis is used for exploring k sets of such 2 by 2 contingency tables. That is, each row of each table is a pair of numbers of successes and number of failures, so that the Odds ratio in contingency table k can be defined as

$$\begin{aligned}
\text{Odds ratio}_k &= \frac{y_{1k}/(N_{1k}-y_{1k})}{y_{2k}/(N_{2k}-y_{2k})} \\
&= \frac{n_{11k}n_{22k}}{n_{12k}n_{21k}}.
\end{aligned}$$

Typically, the individual contingency tables would be for partitioning of groups before and after treatment, and a common situation would be where the aim of the meta analysis would be to assess differences between the results summarized in the individual contingency tables, or to construct a best possible Odds ratio taking into account the sample sizes for appropriate weighting. Suppose, for instance, that contingency table number k is

n_{11k}	n_{12k}	n_{1+k}
n_{21k}	n_{22k}	n_{2+k}
n_{+1k}	n_{+2k}	n_{++k}

where the marginals are indicated by plus signs in the usual way. Then, assuming conditional independence and a hypergeometric distribution (page 418), the mean and variance of n_{11k} are given by

$$\begin{aligned} E(n_{11k}) &= n_{1+k}n_{+1k}/n_{++k} \\ V(n_{11k}) &= \frac{n_{1+k}n_{2+k}n_{+1k}n_{+2k}}{n^2_{++k}(n_{++k}-1)}, \end{aligned}$$

and, to test for significant differences between m contingency tables, the Cochran-Mantel-Haenszel test statistic CMH, given by

$$CMH = \frac{\left\{\left|\sum_{k=1}^{m}(n_{11k} - E(n_{11k}))\right| - \frac{1}{2}\right)^2}{\sum_{k=1}^{m} V(n_{11k})}$$

can be regarded as an approximately chi-square variable with one degree of freedom. Some authors omit the continuity correction and sometimes the variance estimate is taken to be

$$\hat{V}(n_{11k}) = n_{1+k}n_{2+k}n_{+1k}n_{+2k}/n^3_{++k}.$$

As an example, read in `meta.tf1` and observe the calculation of the test statistic as shown in table 12.2. The estimated common odds ratio $\hat{\theta}_{MH}$ presented in table 12.2 is calculated allowing for random effects using

$$\hat{\theta}_{MH} = \frac{\sum_{k=1}^{m}(n_{11k}n_{22k}/n_{++k})}{\sum_{k=1}^{m}(n_{12k}n_{21k}/n_{++k})},$$

while the variance is used to construct the confidence limits from

$$\begin{aligned} \hat{\sigma}^2[\log(\hat{\theta}_{MH})] = & \frac{\sum_{k=1}^{m}(n_{11k}+n_{22k})n_{11k}n_{22k}/n^2_{++k}}{2\left(\sum_{k=1}^{m} n_{11k}n_{22k}/n_{++k}\right)^2} \\ & + \frac{\sum_{k=1}^{m}[(n_{11k}+n_{22k})n_{12k}n_{21k} + (n_{12k}+n_{n21k})n_{11k}n_{22k}]/n^2_{++k}}{2\left(\sum_{k=1}^{m} n_{11k}n_{22k}/n_{++k}\right)\left(\sum_{k=1}^{m} n_{12k}n_{21k}/n_{++k}\right)} \\ & + \frac{\sum_{k=1}^{m}(n_{12k}+n_{21k})n_{12k}n_{21k}/n^2_{++k}}{2\left(\sum_{k=1}^{m} n_{12k}n_{21k}/n_{++k}\right)^2}. \end{aligned}$$

Also, in table 12.2, the overall 2 by 2 contingency table using the pooled sample assuming a fixed effects model is listed for reference, along with the overall odds ratio and estimated confidence limits calculated using the expressions presented previously for an arbitrary log odds ratio (page 183). Table 12.3
illustrates another technique to study sets of 2 by 2 contingency tables. SIMFIT can calculate all the standard probability statistics for sets of paired experiments. In this case the pairwise differences are illustrated along with the number needed to treat i.e. $NNT = 1/d$, but it should be remembered that such estimates have to be interpreted with care. For instance, the differences and log ratios change sign when the rows are interchanged.

```
To test H0: equal binomial p-values
Number of 2 by 2 tables        = 8 (sum of Y = 4081, sum of N = 8419)
Overall estimate of binomial-p = 0.4847 (95%cl = 0.4740,0.4955)
-2 log lambda (-2LL)           = 310.9, NDOF = 15
p = P(chi-sq. >= -2LL)         = 0.0000 Reject H0 at 1% s-level
Chi-sq. test stat (C)          = 306.9, NDOF = 15
p = P(chi-sq. >= C)            = 0.0000 Reject H0 at 1% s-level

Cochran-Mantel-Haenszel 2 x 2 x k Meta Analysis
      y      N  Odds Ratio   E[n(1,1)] Var[n(1,1)]
    126    226     2.19600   113.00000    16.89720
     35     96
    908   1596     2.14296   773.23448   179.30144
    497   1304
    913   1660     2.17526   799.28296   149.27849
    336    934
    235    407     2.85034       203.5    31.13376
     58    179
    402    710     2.31915       355.0    57.07177
    121    336
    182    338     1.58796       169.0    28.33333
     72    170
     60    159     2.36915        53.0         9.0
     11     54
    104    193     2.00321        96.5    11.04518
     21     57
H0: conditional independence (all odds ratios = 1)
CMH Test Statistic     = 2.794E+02
p = P(chi-sq. >= CMH)  = 0.0000 Reject H0 at 1% s-level
Common Odds Ratio      = 2.174, 95%cl = (1.914, 2.471)

Overall 2 by 2 table
       y   N - y
    2930    2359
    1151    1979
Overall Odds Ratio = 2.136 95%cl = (1.95, 2.338)
```

Table 12.2: Analysis of proportions: meta analysis

Again, it should be emphasized that SimFIT outputs values and confidence limits both for the differences $d_{1,2} = \hat{p}_1 - \hat{p}_2$ and the calculated $NNT = 1/d_{1,2}$ values, but the choice between these quantities for data interpretation is controversial. To appreciate the reason why a value of NNT calculated from a sample is just a coarse estimate of the size of a sample needed to treat in order to obtain one additional cure, and could be very misleading, consider the situation of binomial trials with exactly known probabilities p_1 and p_2, and $p_1 > p_2$. The condition that the expectation of a binomial variable X_1 with probability p_1 should be one greater than than a binomial variable X_2 with probability p_2 given a sample size N is

$$\begin{aligned} E(X_1) &= E(X_2) + 1 \\ Np_1 &= Np_2 + 1, \text{ so that} \\ N &= \frac{1}{p_1 - p_2}. \end{aligned}$$

```
d(i,j) = p_hat(i) - p_hat(j), NNT = 1/|d(i,j)|
   i    j     d(i,j) lower-95% upper-95%     Result       Var(d)  NNT (95%c.l.)
   1    2    0.19294   0.07691   0.30897, p( 1) > p( 2)  0.00350    6 (3,14)
   3    4    0.18779   0.15194   0.22364, p( 3) > p( 4)  0.00033    6 (4,7)
   5    6    0.19026   0.15127   0.22924, p( 5) > p( 6)  0.00040    6 (4,7)
   7    8    0.25337   0.16969   0.33706, p( 7) > p( 8)  0.00182    4 (2,6)
   9   10    0.20608   0.14312   0.26903, p( 9) > p(10)  0.00103    5 (3,7)
  11   12    0.11493   0.02360   0.20626, p(11) > p(12)  0.00217    9 (4,43)
  13   14    0.17365   0.04245   0.30486, p(13) > p(14)  0.00448    6 (3,24)
  15   16    0.17044   0.02682   0.31406, p(15) > p(16)  0.00537    6 (3,38)
```

Table 12.3: Analysis of proportions: risk difference

Of course NNT calculated from data is not the exact N as just derived but is given by the random function

$$NNT = \frac{1}{\hat{p}_1 - \hat{p}_2}$$

where there is experimental uncertainty in the parameter estimates. This is one reason why many experts recommend relying on conclusions based directly on the difference $d_{1,2}$, because this quantity is more robust for the purpose of hypothesis testing than NNT where reciprocation exaggerates random effects. Another reason is that it is possible to calculate accurate confidence limits for the difference $d_{1,2}$, but confidence limits calculated for NNT are unsymmetrical and much less intuitive.

Figure 12.2 shows the Log-Odds-Ratio plot with confidence limits resulting from this analysis, after transferring to advanced graphics as just described for ordinary analysis of proportions. The relative position of

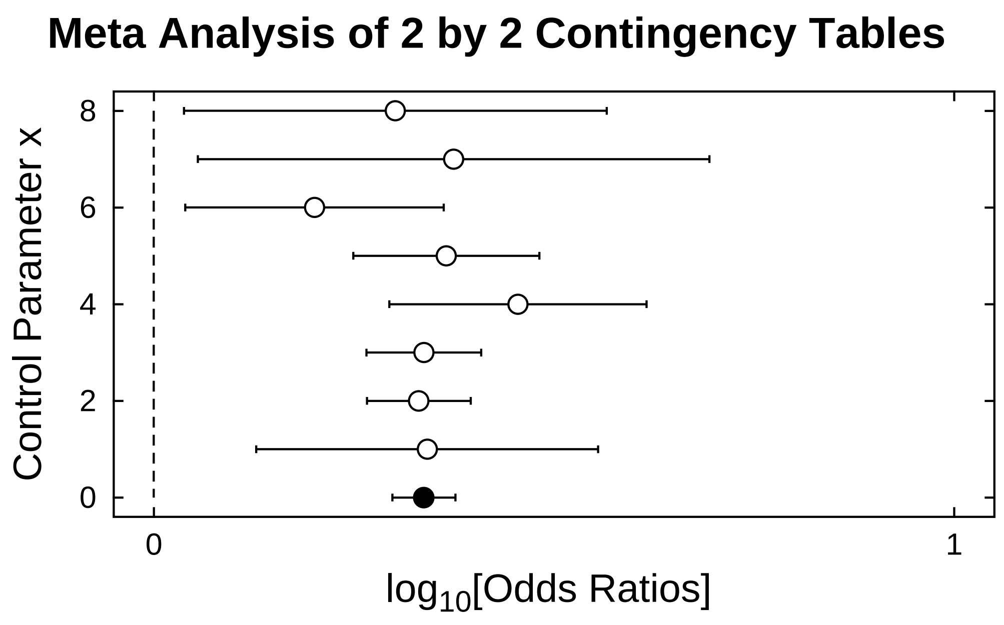


Figure 12.2: Meta analysis and log odds ratios

the data with respect to the line Log-Odds-Ratio = 0 clearly indicates a shift from 50:50 but non-disjoint confidence limits do not suggest statistically significant differences. For further details of the error bars and more advanced plotting see page 188

Contingency table analysis is compromised when cells have zero frequencies, as many of the usual summary statistics become undefined. Structural zeros are handled by applying loglinear GLM analysis but sampling zeros presumably arise from small samples with extreme probabilities. Such tables can be analyzed by exact methods, but usually a positive constant is added to all the frequencies to avoid the problems. Table 12.4 illustrates how this problem is handled in SIMFIT when analyzing data in the test file `meta.tf4`; the correction

```
Cochran-Mantel-Haenszel 2 x 2 x k Meta Analysis
      y      N  Odds Ratio   E[n(1,1)] Var[n(1,1)]
*** 0.01 added to all cells for next calculation
      0      6     0.83361     0.01091     0.00544
      0      5
*** 0.01 added to all cells for next calculation
      3      6       601.0        1.51     0.61686
      0      6
*** 0.01 added to all cells for next calculation
      6      6  1199.00995        4.01     0.73008
      2      6
*** 0.01 added to all cells for next calculation
      5      6     0.00825        5.51     0.25454
      6      6
*** 0.01 added to all cells for next calculation
      2      2     0.40120     2.01426     0.00476
      5      5
H0: conditional independence (all odds ratios = 1)
CMH Test Statistic = 3.862
P(chi-sq. >= CMH)  = 0.0494 Reject H0 at 5% s-level
Common Odds Ratio  = 6.749, 95%cl = (1.144, 39.81)
```

Table 12.4: Analysis of proportion: meta analysis with zero frequencies

of adding 0.01 to all contingency tables frequencies being indicated. Values ranging from 0.00000001 to 0.5 have been suggested elsewhere for this purpose, but all such choices are a compromise and, if possible, sampling should be continued until all frequencies are nonzero.

12.3 Bioassay, estimating percentiles

Where it is required to construct a dose response curve from sets of (y, N) data at different levels of an independent variable, x, it is sometimes useful to apply probit analysis or logistic regression to estimate percentiles, like LD50 (page 113) using generalized linear models (page 49). To observe how this works, read in the test file `ld50.tf1` and try the various options for choosing models, plotting graphs and examining residuals.

12.4 Trichotomous data

This procedure is used when an experiment has three possible outcomes, e.g., an egg can fail to hatch, hatch male, or hatch female, and you wish to compare the outcome from a set of experiments. For example, read in `trinom.tf1` then `trinom.tf2` to see how to detect significant differences graphically (i.e., where there are non-overlapping confidence regions) in trinomial proportions, i.e., where groups can be split into three categories. For details of the trinomial distribution see page 418 and for plotting contours see page 287.

12.5 Plotting binomial error bars

Figure 12.3 shows binomial parameter estimates for y successes in N trials. The error bars represent exact, unsymmetrical confidence limits (see page 285), not those calculated using the normal approximation.

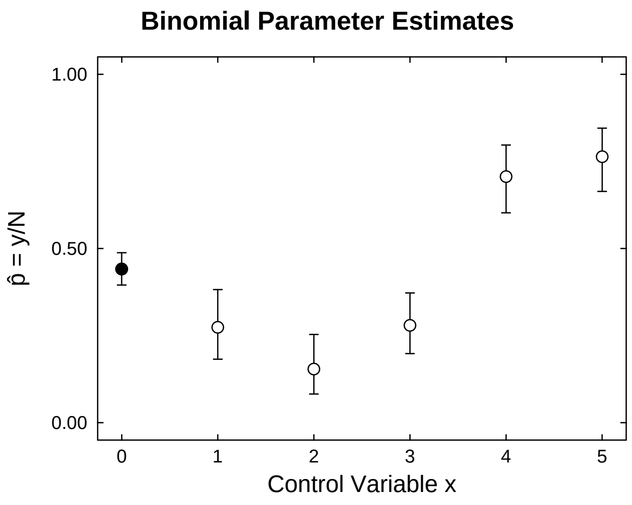


Figure 12.3: Binomial parameter error bars

12.6 Plotting Log-Odds error bars

Figure 12.3 can also be manipulated by transforming the estimates $\hat{p} = y/N$ and confidence limits. For instance, the ratio of success to failure (i.e. Odds $y/(N - y)$) or the logarithm (i.e. Log Odds) can be used, as in figure 12.4, to emphasize deviation from a fixed p value, e.g. $p = 0.5$ with a log-odds of 0. Figure 12.4 was created from a simple log-odds plot by using the [Advanced] option to transfer the x, $\hat{p}/(1 - \hat{p})$ data into **simplot**, then selecting a reverse y-semilog transform.

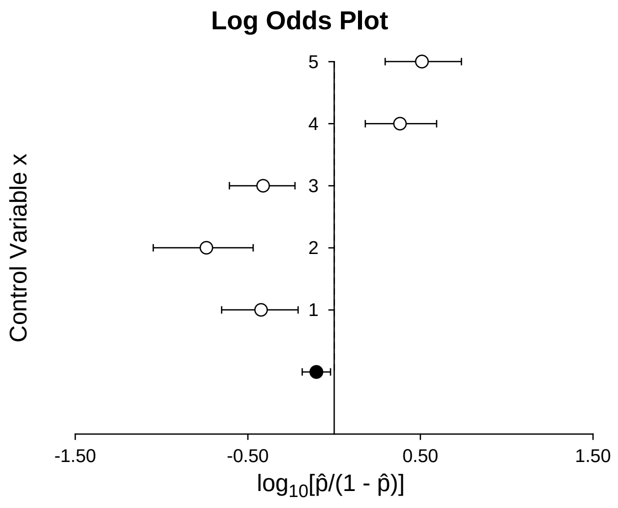


Figure 12.4: Log-Odds error bars

12.7 Plotting meta analysis error bars

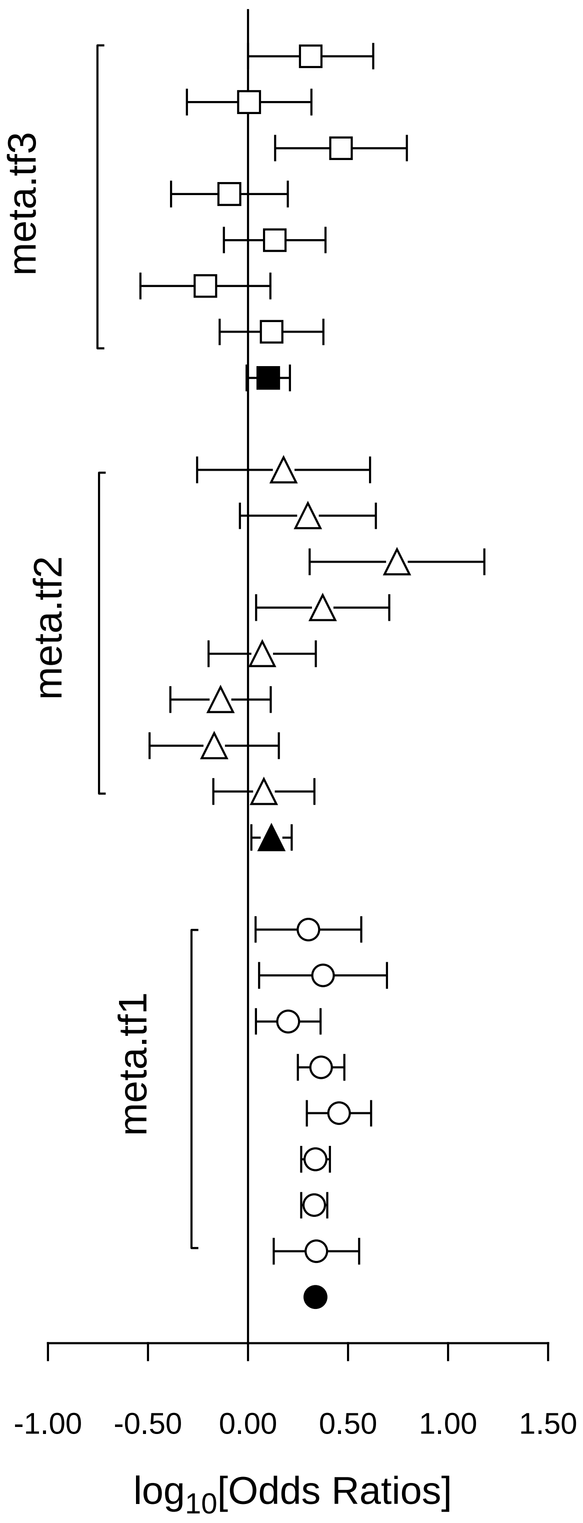


Figure 12.5: Log-Odds-Ratios error bars

It is often useful to plot Log-Odds-Ratios, so the creation of figure 12.5 will be outlined.

(1) **The data**
Test files `meta.tf1`, `meta.tf2`, and `meta.tf3` were analyzed in sequence using the SIMFIT Meta Analysis procedure (page 183). Note that, in these files, column 3 contains spacing coordinates so that data will be plotted consecutively.

(2) **The ASCII coordinate files**
During Meta Analysis, $100(1-\alpha)\%$ confidence limits on the Log-Odds-Ratio resulting from a 2 by 2 contingency tables with cell frequencies n_{ij} can be constructed from the approximation $\hat{e}$ where

$$\hat{e} = Z_{\alpha/2}\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}.$$

When Log-Odds-Ratios with error bars are displayed, the overall values (shown as filled symbols) with error bars are also plotted with a x coordinate one less than smallest x value on the input file. For this figure, error bar coordinates were transferred into the project archive using the [Advanced] option to save ASCII coordinate files.

(3) **Creating the composite plot**
Program **simplot** was opened and the six error bar coordinate files were retrieved from the project archive. Experienced users would do this more easily using a library file of course. Reverse y-semilog transformation was selected, symbols were chosen, axes, title, and legends were edited, then half bracket hooks identifying the data were added as arrows and extra text.

(4) **Creating the PostScript file**
Vertical format was chosen then, using the option to stretch PostScript files (page 210), the y coordinate was stretched by a factor of two.

(5) **Editing the PostScript file**
To create the final PostScript file for LaTeX a tighter bounding box was calculated using **gsview** then, using **notepad**, clipping coordinates at the top of the file were set equal to the BoundingBox coordinates, to suppress excess white space. This can also be done using the [Style] option to omit painting a white background, so that PostScript files are created with transparent backgrounds, i.e. no white space, and clipping is irrelevant.

Part 13

Multivariate statistics

13.1 Introduction

It is assumed that the data are in the form of a n by m matrix, where the n rows represent cases and the m columns are variables, and it is wished to explore relationships between the rows and columns. Frequently this requires graphical display, so data files for multivariate analysis often have row and column labels in addition to the nm observations.

13.2 Correlation: parametric (Pearson product moment)

Given any set of n nonsingular (x_i, y_i) pairs, a correlation coefficient r can be calculated as

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $-1 \le r \le 1$ and, using b_{xy} for the slope of the regression of X on Y, and b_{yx} for the slope of the regression of Y on X, it will be shown later that

$$r^2 = b_{yx} b_{xy}.$$

However, only when X is normally distributed given Y, and Y is normally distributed given X can simple statistical tests be used for significant linear correlation. Figure 13.1 illustrates how the elliptical contours of constant probability for a bivariate normal distribution discussed on page 421 are aligned with the X and Y axes when X and Y are uncorrelated, i.e., $\rho = 0$ but are inclined otherwise. In this example $\mu_X = \mu_Y = 0$ and $\sigma_x = \sigma_Y = 1$, but in the upper figure $\rho = 0$, while in the lower figure $\rho = 0.9$. The Pearson product moment correlation coefficient r is an estimator of ρ, and it can can be used to test for independence of X and Y. For instance, when the (x_i, y_i) pairs are from such a bivariate normal distribution, the statistic

$$t = r\sqrt{\frac{n-2}{1-r^2}}$$

has a Student's t-distribution with $n - 2$ degrees of freedom.

The SIMFIT product moment correlation procedure can be used when you have a data matrix X consisting of $m > 1$ columns of $n > 1$ measurements (not counts or categorical data) and wish to test for pairwise linear correlations, i.e., where pairs of columns can be regarded as consistent with a joint normal distribution. In matrix notation, the relationships between such a n by m data matrix X, the same matrix Y after centering by

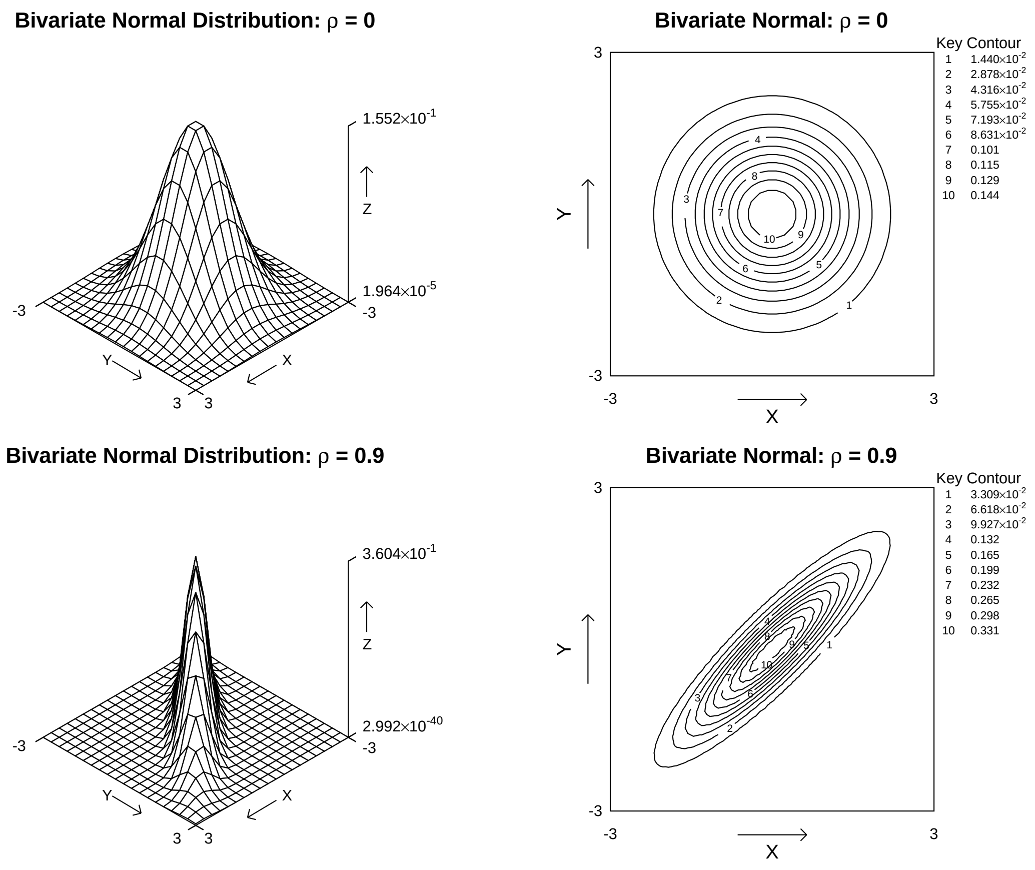


Figure 13.1: Bivariate density surfaces and contours

subtracting each column mean from the corresponding column, the sum of squares and products matrix C, the covariance matrix S, the correlation matrix R, and the diagonal matrix D of standard deviations are

$$\begin{aligned} C &= Y^T Y \\ S &= \frac{1}{n-1} C \\ D &= \operatorname{diag}(\sqrt{s_{11}}, \sqrt{s_{22}}, \dots, \sqrt{s_{mm}}) \\ R &= D^{-1} S D^{-1} \\ S &= DRD. \end{aligned}$$

So, for all pairs of columns, the sample correlation coefficients r_{jk} are given by

$$r_{jk} = \frac{s_{jk}}{\sqrt{s_{jj} s_{kk}}},$$
$$\text{where } s_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k),$$

and the corresponding t_{jk} values and significance levels p_{jk} are calculated then output in matrix format with the correlations as a strict upper triangular matrix, and the significance levels as a strict lower triangular matrix.

Table 13.1 shows the results from analyzing the test file `g02baf.tf1`, which refers to a set of 3 column vectors of length 5. To be more precise, the values a_{ij} for matrix A in table 13.1 are interpreted as now described.

```
Matrix A, Pearson correlation results
Upper triangle = r, Lower = corresponding two-tail p values

  ..... -0.5704  0.1670
 0.3153   ..... -0.7486
 0.7883 0.1455    .....

Test for absence of any significant correlations
H0: correlation matrix is the identity matrix
Determinant          = 0.229
Test statistic (TS) = 3.194
Degrees of freedom  = 3
p = P(chi-sq >= TS) = 0.3627
```

Table 13.1: Correlation: Pearson product moment analysis

For $j > i$ in the upper triangle, then $a_{ij} = r_{ij} = r_{ji}$ are the correlation coefficients, while for $i > j$ in the lower triangle $a_{ij} = p_{ij} = p_{ji}$ are the corresponding two-tail probabilities. The self-correlations are all 1, of course, and so they are represented by dotted lines. Table 13.1 indicates that none of the correlations are significant in this case, that is, the probability of obtaining such pairwise linearity in a random swarm of points is not low, but after the correlation matrix the results of a likelihood ratio test for the absence of significant correlations are displayed. To test the hypothesis of no significant correlations, i.e. H_0: the covariance matrix is diagonal, or equivalently H_0: the correlation matrix R is the identity matrix, the statistic

$$-2\log\lambda = -(n - (2m + 11)/6)\log|R|$$

is used, which has the asymptotic chi-square distribution with $m(m-1)/2$ degrees of freedom.

After the results have been calculated you can choose pairs of columns for further analysis, as shown for the test file `cluster.tf1` in table 13.2, where there seem to be significant correlations. First the test for significant correlation was done, then columns 1 and 2 were selected for further analysis, consisting of all the statistics necessary to study the regression of column 1 on column 2 and vice versa. Various graphical techniques are then possible to visualize correlation between the columns selected by superimposing best-fit lines or confidence region ellipses (page 196). Highly significant linear correlation is indicated by best-fit lines with similar slopes as well as r values close to 1 and small p values. Note that, after a correlation analysis, a line can be added to the scattergram to indicate the extent of rotation of the axes of the ellipses from coincidence with the X, Y axes.

13.2.1 Plotting lines on correlation diagrams

You can plot either both unweighted regression lines, the unweighted reduced major axis line, or the unweighted major axis line on such scattergrams and the difference between these types will now be outlined.

For n pairs (x_i, y_i) with mean $x = \bar{x}$ and mean $y = \bar{y}$, the variances and covariance required are

```
Test for absence of any significant correlations
H0: correlation matrix is the identity matrix
Determinant          = 0.002476
Test statistic (TS) = 45.01
Degrees of freedom  = 28
P(chi-sq >= TS)      = 0.0220 Reject H0 at 5% sig.level

For the next analysis: X is column  1, Y is column  2

Linear regression: y(x) = A + B*x, x(y) = C + D*y

Sample size =  12
For X: mean =  8.8333, std. dev. = 5.7814, var. = 33.424
For Y: mean =  9.9167, std. dev. = 7.5973, var. = 57.72

Parameter     Estimate     Std.Err.  Est./Std.Err.     p
B (slope)      0.69583      0.35252        1.9739    0.0766
A (const)       3.7702       3.6748        1.0260    0.3291
r (Ppmcc)      0.52951      0.26826        1.9739    0.0766
r-squared      0.28038, y-variation due to x = 28.04%
z(Fisher)      0.58946, Note: z = (1/2)log[(1+r)/(1-r)],
r^2=B*D, t=r*sqrt[(n-2)/(1-r^2))]=Est./Std.Err. for B,D,and r
The Pearson product-moment corr. coeff. r estimates rho and
95% conf. limits using z are  -0.0771 =< rho =<  0.85

      Source     Sum of squares  ndof  Mean square     F-value    p
due to regression      178.02       1       178.02      3.8962  0.7665
about regression       456.90      10        45.69
total                  634.92      11
Conclusion: B is not significantly different from zero (p > 0.05)
            A is not significantly different from zero (p > 0.05)

The two best-fit unweighted regression lines are:
y(x) =  3.7702 + 0.69583*x, x(y) = 4.8375 + 0.40294*y
```

Table 13.2: Correlation: analysis of selected columns

$$S_{xx} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$
$$S_{yy} = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$$
$$S_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}).$$

Also, for an arbitrary point (x_i, y_i) and a straight line defined by $y = a + bx$ the squares of the vertical, horizontal, and orthogonal (i.e. perpendicular) distances, v_i^2, h_i^2, and o_i^2 between the point and the line are

$$\begin{aligned} v_i^2 &= [y_i - (a + bx_i)]^2 \\ h_i^2 &= v_i^2/b^2 \\ o_i^2 &= v_i^2/(1 + b^2). \end{aligned}$$

13.2.1.1 Ordinary least squares

If x is regarded as an exact variable free from random variation or measurement error while y has random variation, then the best fit line from minimizing the sum of v_i^2 is

$$y_1(x) = \hat{\beta}_1 x + [\bar{y} - \hat{\beta}_1 \bar{x}]$$

where $\hat{\beta}_1 = S_{xy}/S_{xx}$. However, if y is regarded as an exact variable while x has random variation, then the best fit line for x as a function of y from minimizing the sum of h_i^2 would be

$$x_2(y) = (1/\hat{\beta}_2)y + [\bar{x} - (1/\hat{\beta}_2)\bar{y}]$$

where $\hat{\beta}_2 = S_{yy}/S_{xy}$ or, rearranging to express the line as $y_2(x)$,

$$y_2(x) = \hat{\beta}_2 x + [\bar{y} - \hat{\beta}_2 \bar{x}],$$

emphasizing that the slope of the regression line for $y_2(x)$ is the reciprocal of the slope for $x_2(y)$. Since neither of these two best fit lines can be regarded as satisfactory, SimFIT plots both lines such that $y_1(x)$ covers the range of x values while $x_2(y)$ covers the range of y values. However these two lines intersect at $(\bar{x}, \bar{y})$ and, from the fact that the ratio of slopes equals the square of the correlation coefficient, that is,

$$r^2 = \hat{\beta}_1/\hat{\beta}_2,$$

then two best fit lines with similar slopes suggests strong linear correlation, whereas one line almost parallel to the x axis and the other almost parallel to the y axis would indicate negligible linear correlation. For instance, if there is no linear correlation between x and y, then the slope of the regression line for $y(x)$ i.e. $\hat{\beta}_1$ would be zero, as would be the slope of the regression line for $x(y)$ i.e. $1/\hat{\beta}_2$ leading to $r^2 = 0$. Conversely strong linear correlation would lead to $\hat{\beta}_1 = \hat{\beta}_2$ and $r^2 = 1$.

The major axis and reduced major axis lines to be discussed next are attempts to get round the necessity to plot two lines and just have one best fit line intermediate between these two lines to represent the correlation.

13.2.1.2 The major axis line

Here it is the sum of o_i^2, the squares of the orthogonal distances between the points and the best fit line, that is minimized to yield the slope as

$$\hat{\beta}_3 = \frac{1}{2}\left(\hat{\beta}_2 - (1/\hat{\beta}_1) + \gamma\sqrt{4 + (\hat{\beta}_2 - (1/\hat{\beta}_1))^2}\right)$$

where $\gamma = 1$ if $S_{xy} > 0$, $\gamma = 0$ if $S_{xy} = 0$, and $\gamma = -1$ if $S_{xy} < 0$, so that the major axis line is

$$y_3(x) = \hat{\beta}_3 x + [\bar{y} - \hat{\beta}_3 \bar{x}].$$

Actually $\hat{\beta}_3$ is the slope of the first principal component axis and so it points in the direction of maximum variability.

13.2.1.3 The reduced major axis line

Instead of minimizing the sum of squares of the vertical distances v_i^2, or horizontal distances h_i^2, it is possible to minimize the sum of the areas of the triangles formed by the v_i, h_i with the best fit line as hypotenuse, i.e. $v_i h_i/2$, to obtain the reduced major axis line as

$$y_4(x) = \hat{\beta}_4 x + [\bar{y} - \hat{\beta}_4 \bar{x}].$$

Here

$$\begin{aligned}\hat{\beta}_4 &= \gamma\sqrt{S_{yy}/S_{xx}} \\ &= \gamma\sqrt{\hat{\beta}_1\hat{\beta}_2}\end{aligned}$$

so that the slope of the reduced major axis line is the geometric mean of the slopes of the regression of y on x and x on y.

13.2.1.4 Plotting scattergrams, clusters, and connections

Plotting both regression lines is the most useful and least controversial as in figure 13.2.

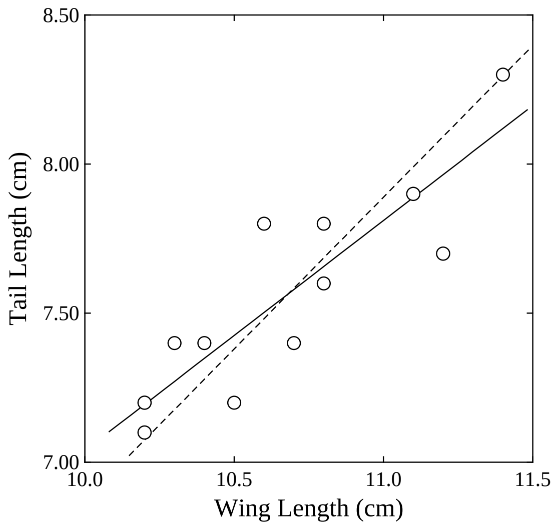


Figure 13.2: Correlations and scattergrams

If a single line must be plotted to summarize the overall correlation it should be the reduced major axis line, as this allows for uncertainty in both variables and is not so controversial as the major axis line, which requires both axes to have similar units, as in allometry. It should not be either of the regression lines, since the line plotted must be independent of which variable is regarded as x and which is regarded as y.

Clusters can be plotted as sideways displaced and reduced symbols, while connections need individual data files for distinct lines and symbols, as in figure 13.3.

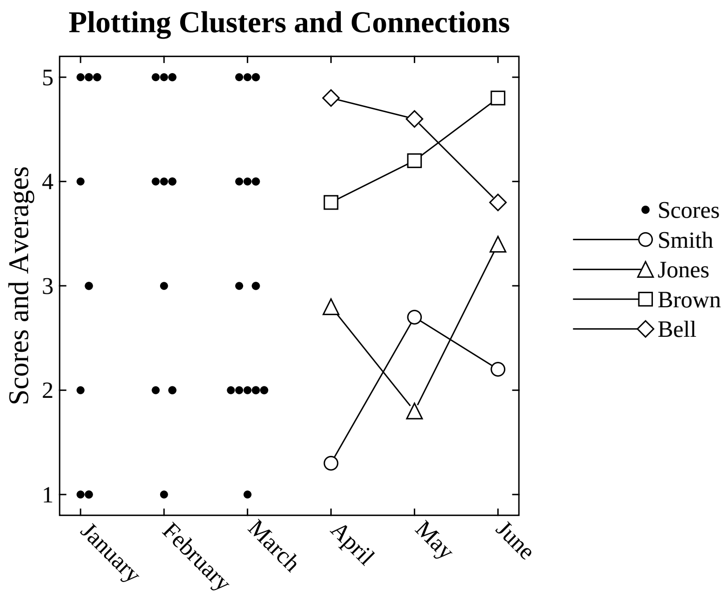


Figure 13.3: Clusters and connections

13.2.2 Plotting bivariate confidence ellipses: basic theory

For a p-variate normal sample of size n with mean $\bar{x}$ and variance matrix estimate S, the region

$$P\left\{(\bar{x}-\mu)^T S^{-1}(\bar{x}-\mu) \le \frac{p(n-1)}{n(n-p)} F^{\alpha}_{p,n-p}\right\} \le 1-\alpha$$

can be regarded as a $100(1-\alpha)\%$ confidence region for μ. Figure 13.4 illustrates this for columns 1 and 2 of `cluster.tf1` discussed previously (page 193). Alternatively, the region satisfying

$$P\left\{(x-\bar{x})^T S^{-1}(x-\bar{x}) \le \frac{p(n^2-1)}{n(n-p)} F^{\alpha}_{p,n-p}\right\} \le 1-\alpha$$

can be interpreted as a region that with probability $1-\alpha$ would contain another independent observation x, as shown for the swarm of points in figure 13.4. The μ confidence region contracts with increasing n, limiting

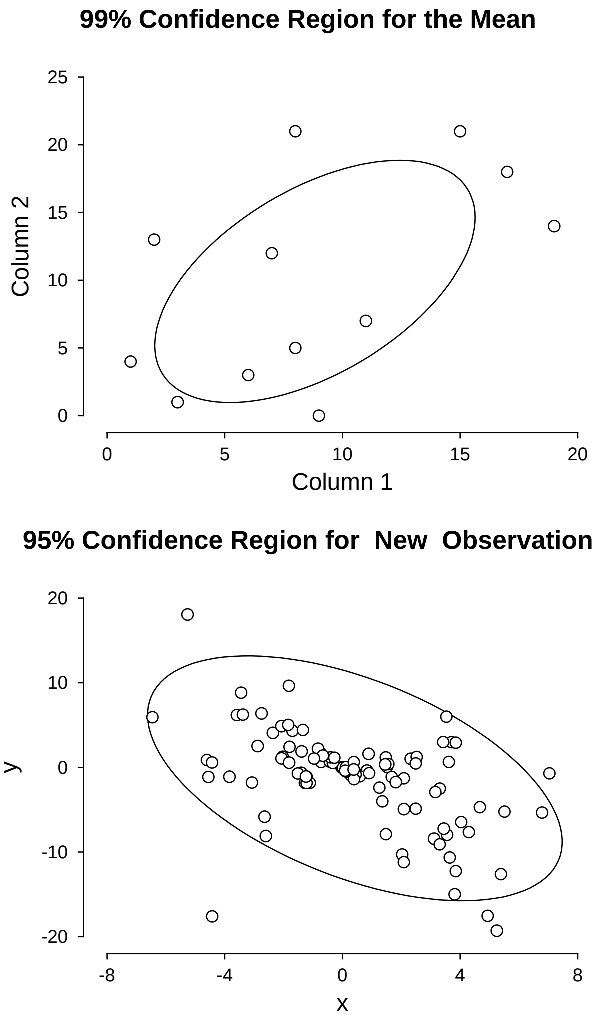


Figure 13.4: Confidence ellipses for a bivariate normal distribution

application to small samples, but the new observation ellipse does not, making it useful for visualizing if data do represent a bivariate normal distribution, while inclination of the principal axes away from parallel with the plot axes demonstrates linear correlation. This technique is only justified if the data are from a bivariate normal distribution and are independent of the variables in the other columns, as indicated by the correlation matrix.

13.2.3 Plotting bivariate confidence ellipses: regions

Often a two dimensional swarm of points results from projecting data that have been partitioned into groups into a subspace of lower dimension in order to visualize the distances between putative groups, e.g., after principal components analysis or similar. If the projections are approximately bivariate normal then confidence ellipses can be added, as in figure 13.5.

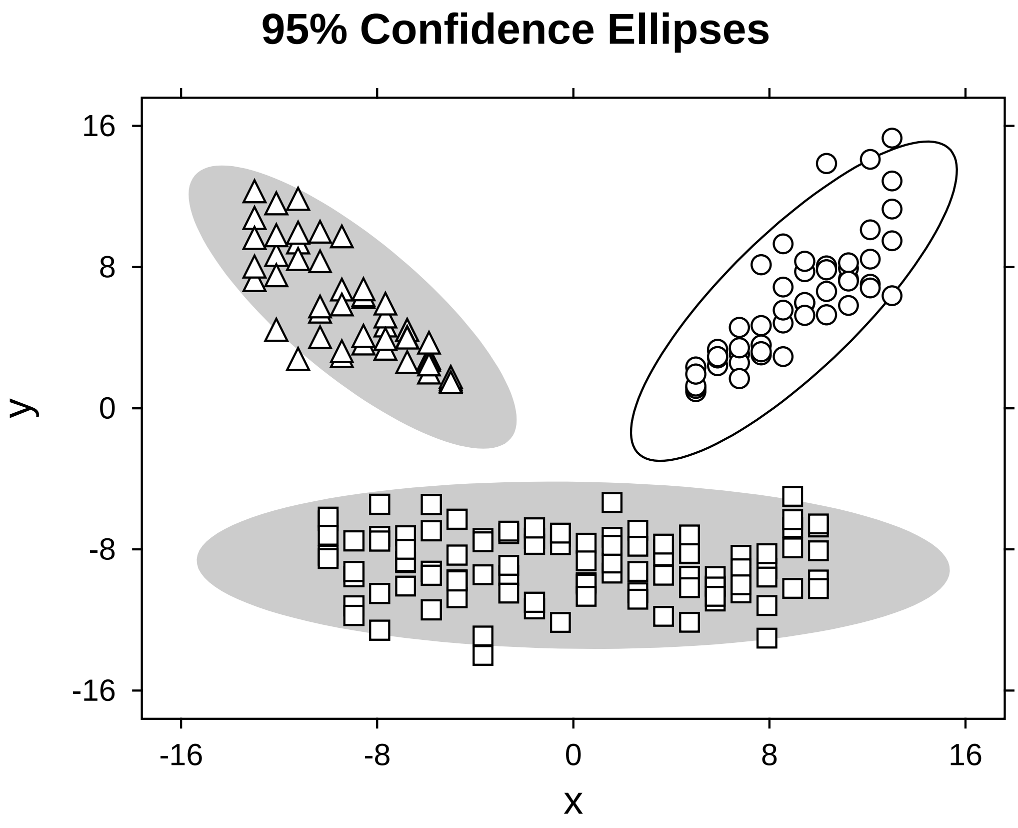


Figure 13.5: 95% confidence regions

The following steps were used to create figure 13.5 and can be easily adapted for any number of sets of two dimensional group coordinates.

- ❍ For each group a file of values for *x* and *y* coordinates in the projected space was saved.
- ❍ Each file was analyzed for correlation using the SIMFIT correlation analysis procedure.
- ❍ After each correlation analysis, the option to create a 95% confidence ellipse for the data was selected, and the ellipse coordinates were saved to file.
- ❍ A library file was created with the ellipse coordinates as the first three files, and the groups data files as the next three files.
- ❍ The library file was read into **simplot**, then colors and symbols were chosen.

Note that, because the ellipse coordinates are read in as the first coordinates to be plotted, the option to plot lines as closed polygons can be used to represent the confidence ellipses as colored background regions.

13.3 Correlation: nonparametric (Kendall tau and Spearman rank)

These nonparametric procedures can be used when the data matrix does not consist of columns of normally distributed measurements, but may contain counts or categorical variables, etc. so that the conditions for Pearson product moment correlation are not satisfied and ranks have to be used. Suppose, for instance, that the data matrix has n rows (observations) and m columns (variables) with $n > 1$ and $m > 1$, then the x_{ij} are replaced by the corresponding column-wise ranks y_{ij}, where groups of tied values are replaced by the average of the ranks that would have been assigned in the absence of ties. Kendall's tau τ_{jk} for variables j and k is then defined as

$$\tau_{jk} = \frac{\sum_{h=1}^{n}\sum_{i=1}^{n} f(y_{hj} - y_{ij}) f(y_{hk} - y_{ik})}{\sqrt{[n(n-1) - T_j][n(n-1)T_k]}},$$

$$\begin{aligned} \text{where } f(u) &= 1 \text{ if } u > 0,\\ &= 0 \text{ if } u = 0,\\ &= -1 \text{ if } u < 0,\\ \text{and } T_j &= \sum t_j(t_j - 1). \end{aligned}$$

Here t_j is the number of ties at successive tied values of variable j, and the summation is over all tied values. For large samples τ_{jk} is approximately normally distributed with

$$\mu = 0$$
$$\sigma^2 = \frac{4n + 10}{9n(n-1)}$$

which can be used as a test for the absence of correlation. Another alternative is to calculate Spearman's rank coefficient c_{jk}, defined as

$$c_{jk} = \frac{n(n^2 - 1) - 6\sum_{i=1}^{n}(y_{ij} - y_{ik})^2 - (T_j + T_k)/2}{\sqrt{[n(n^2 - 1) - T_j][n(n^2 - 1)T_k]}}$$

$$\text{where now } T_j = \sum t_j(t_j^2 - 1)$$

and a test can be based on the fact that, for large samples, the statistic

$$t_{jk} = c_{jk}\sqrt{\frac{n-2}{1 - c_{jk}^2}}$$

is approximately t-distributed with $n - 2$ degrees of freedom.

For example, read in and analyze the test file `npcorr.tfl` as previously to obtain Table 13.3. To be more precise, matrices A and B in table 13.3 are to be interpreted as follows. In the first matrix A, for $j > i$ in the upper triangle, then $a_{ij} = c_{ij} = c_{ji}$ are Spearman correlation coefficients, while for $i > j$ in the lower triangle $a_{ij} = \tau_{ij} = \tau_{ji}$ are the corresponding Kendall coefficients. In the second matrix B, for $j > i$ in the upper triangle, then $b_{ij} = p_{ij} = p_{ji}$ are two-tail probabilities for the corresponding c_{ij} coefficients, while for $i > j$ in the lower triangle $b_{ij} = p_{ij} = p_{ji}$ are the corresponding two-tail probabilities for the corresponding τ_{ij}. Note that, from these matrices, τ_{jk}, c_{jk} and p_{jk} values are given for all possible correlations j, k. Also, note that these nonparametric correlation tests are tests for monotonicity rather that linear correlation but, as with the previous parametric test, the columns of data must be of the same length and the values must be ordered according to some correlating influence such as multiple responses on the same animals. If the number of categories is small or there are many ties, then Kendall's Tau is to be preferred and conversely. Since you are not testing for linear correlation you should not add regression lines when plotting such correlations.

```
Nonparametric correlation results

Matrix A: Upper triangle = Spearman's, Lower = Kendall's tau
  .....  0.2246  0.1186
 0.0294   .....  0.3814
 0.1176  0.2353   .....

Matrix B: Two tail p-values
  .....  0.5613  0.7611
 0.9121   .....  0.3112
 0.6588  0.3772   .....
```

Table 13.3: Correlation: Kendall-tau and Spearman-rank

13.4 Correlation: partial

Partial correlations are useful when it is believed that some subset of the variables in a multivariate data set can realistically be regarded as normally distributed random variables, and correlation analysis is required for this subset of variables, conditional upon the remaining variables being regarded as fixed at their current values. This is most easily illustrated in the case of three variables, and table 13.4 illustrates the calculation of partial correlation coefficients, together with significance tests and confidence limits for the correlation matrix in `pacorr.tf1`. Assuming a multivariate normal distribution and linear correlations, the partial correlations

```
Partial correlation data: 1=Intelligence, 2=Weight, 3=Age
1.0000  0.6162  0.8267
        1.0000  0.7321
                1.0000
No. variables = 3, sample size = 30
r(1,2) = 0.6162
r(1,3) = 0.8267
r(2,3) = 0.7321
......
r(1,2|3) = 0.0286 (95%c.l. = -0.3422,  0.3918)
t = 0.1488, ndof = 27, p = 0.8828
......
r(1,3|2) = 0.7001 (95%c.l. =  0.4479,  0.8490)
t = 5.094, ndof = 27, p = 0.0000 Reject H0 at 1% sig.level
......
r(2,3|1) = 0.5025 (95%c.l. =  0.1659,  0.7343)
t = 3.020, ndof = 27, p = 0.0055 Reject H0 at 1% sig.level
```

Table 13.4: Correlation: partial

between any two variables from the set i, j, k conditional upon the third can be calculated using the usual correlation coefficients as

$$r_{i,j|k} = \frac{r_{ij} - r_{ik}r_{jk}}{\sqrt{(1 - r_{ik}^2)(1 - r_{jk}^2)}}.$$

If there are p variables in all but $p - q$ are fixed then the sample size n can be replaced by $n - (p - q)$ in the usual significance tests and estimation of confidence limits, e.g. $n - (p - q) - 2$ for a t test. From table 13.4 it is clear that when variable 3 is regarded as fixed, the correlation between variables 1 and 2 is not significant but, when either variable 1 or variable 2 are regarded as fixed, there is evidence for significant correlation between the other variables.

The situation is more involved when there are more than three variables, say n_x X variables which can be regarded as fixed, and the remaining n_y Y variables for which partial correlations are required conditional on the fixed variables. Then the variance-covariance matrix Σ can be partitioned as in

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}$$

when the variance-covariance of Y conditional upon X is given by

$$\Sigma_{y|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy},$$

while the partial correlation matrix R is calculated by normalizing as

$$R = \mathrm{diag}(\Sigma_{y|x})^{-\frac{1}{2}}\,\Sigma_{y|x}\,\mathrm{diag}(\Sigma_{y|x})^{-\frac{1}{2}}.$$

This analysis requires a technique for indicating that a full correlation matrix is required for all the variables, but then in a subsequent step some variables are to be regarded as X variables, and others as Y variables. All this can be done interactively but SIMFIT provides a convenient method for doing this directly from the data file. For instance, at the end of the test file `g02byf.tf1`, which is a full data set, not a correlation matrix, will be found the additional lines

```
begin{indicators}
-1 -1  1
end{indicators}
```

and the indicator variables have the following significance. A value of -1 indicates that the corresponding variable is to be used in the calculation of the full correlation matrix, but then this variable is to be regarded as a Y variable when the partial correlation matrix is calculated. A value of 1 indicates that the variable is to be included in the calculation of the full correlation matrix, then regarded as an X variable when the partial correlation matrix is to be calculated. Any values of 0 indicate that the corresponding variables are to be suppressed. Table 13.5 illustrates the successive results for test file `g02byf.tf1` when Pearson correlation is performed, followed by partial correlation with variables 1 and 2 regarded as Y and variable 3 regarded as X. Exactly as for the full correlation matrix, the strict upper triangle of the output from the partial correlation

```
Matrix A, Pearson product moment correlation results:
Upper triangle = r, Lower = corresponding two-tail p values
  ..... 0.7560 0.8309
 0.0011  ..... 0.9876
 0.0001 0.0000  .....
Test for absence of any significant correlations
H0: correlation matrix is the identity matrix
Determinant          =  0.003484
Test statistic (TS) =  68.86
Degrees of freedom   =  3
P(chi-sq >= TS)      =  0.0000 Reject H0 at 1% sig.level

Matrix B, partial correlation results for variables: yyx
Upper triangle: partial r, Lower: corresponding 2-tail p values
    ...-0.7381
 0.0026    ...
```

Table 13.5: Correlation: partial correlation matrix

analysis contains the partial correlation coefficients r_{ij}, while the strict lower triangle holds the corresponding two tail probabilities p_{ij} where

$$p_{ij} = P\left(t_{n-n_x-2} \le -|r_{ij}|\sqrt{\frac{n-n_x-2}{1-r_{ij}^2}}\right) + P\left(t_{n-n_x-2} \ge |r_{ij}|\sqrt{\frac{n-n_x-2}{1-r_{ij}^2}}\right).$$

To be more precise, the values a_{ij} and b_{ij} in the matrices A and B of table 13.5 are interpreted as now described. In the first matrix A, for $j > i$ in the upper triangle, then $a_{ij} = r_{ij} = r_{ji}$ are full correlation coefficients, while for $i > j$ in the lower triangle $a_{ij} = p_{ij} = p_{ji}$ are the corresponding two-tail probabilities. In the second matrix B, for $j > i$ in the upper triangle, then $b_{ij} = r_{ij} = r_{ji}$ are partial correlation coefficients, while for $i > j$ in the lower triangle $b_{ij} = p_{ij} = p_{ji}$ are the corresponding two-tail probabilities.

13.5 Correlation: canonical

This technique is employed when a n by m data matrix includes at least two groups of variables, say n_x variables of type X, and n_y variables of type Y, measured on the same n subjects, so that $m \ge n_x + n_y$. The idea is to find two transformations, one for the X variables to generate new variables V, and one for the Y variables to generate new variables U, with l components each for $l \le \min(n_x, n_y)$, such that the canonical variates u_1, v_1 calculated from the data using these transformations have maximum correlation, then u_2, v_2, and so on. Now the variance-covariance matrix of the X and Y data can be partitioned as

$$\begin{pmatrix} S_{xx} & S_{xy} \\ S_{yx} & S_{yy} \end{pmatrix}$$

and it is required to find transformations that maximize the correlations between the X and Y data sets. Actually, the equations

$$(S_{xy}S_{yy}^{-1}S_{yx} - R^2 S_{xx})a = 0$$
$$(S_{yx}S_{xx}^{-1}S_{xy} - R^2 S_{yy})b = 0$$

have the same nonzero eigenvalues as the matrices $S_{xx}^{-1}S_{xy}S_{yy}^{-1}S_{yx}$ and $S_{yy}^{-1}S_{yx}S_{xx}^{-1}S_{xy}$, and the square roots of these eigenvalues are the canonical correlations, while the eigenvectors of the two above equations define the canonical coefficients, i.e. loadings. Table 13.6 shows the results from analyzing data in the test file `g03adf.tf1`, which has 9 rows and 4 columns. Users of this technique should note that the columns of

```
Variables: yxxy
Number of x = 2, Number of y = 2, Number unused = 0
Minimum of rank of x and rank of y = 2
Correlations  Eigenvalues  Proportions      Chi-sq.  NDOF     p
      0.9570      0.91591       0.8746       14.391     4  0.0061
      0.3624      0.13133       0.1254      0.77438     1  0.3789
CVX: Canonical coefficients for centralized X
 -0.4261  1.034
 -0.3444 -1.114
CVY: Canonical coefficients for centralized Y
 -0.1415  0.1504
 -0.2384 -0.3424
```

Table 13.6: Correlation: canonical

the data matrix must be indicated by setting a n by 1 integer vector with values of 1 for X, 0 for variable suppressed, or -1 for Y. Such variable indicators can be initialized from the trailing section of the data file, by using the special token `begin{indicators}`, as will be seen at the end of `g03adf.tf1`, where the following indicator variables are appended

```
begin{indicators}
  -1  1  1  -1
end{indicators}
```

indicating that variables 1 and 4 are to be considered as Y variables, while variables 2 and 3 are to be regarded as X variables. However, the assignment of data columns to groups of type X, suppressed, or Y can also be adjusted interactively if required. Note that the eigenvalues are proportional to the correlation explained by the corresponding canonical variates, so a scree diagram can be plotted to determine the minimum number of canonical variates needed to adequately represent the data. This diagram plots the eigenvalues together with the average eigenvalue, and the canonical variates with eigenvalues above the average should be retained. Alternatively, assuming multivariate normality, the likelihood ratio test statistics

$$-2 \log \lambda = -(n - (k_x + k_y + 3)/2) \sum_{j=i+1}^{l} \log(1 - R_j^2)$$

can be calculated for $i = 0, 1, \ldots, l-1$, where $k_x \le n_x$ and $k_y \le n_y$ are the ranks of the X and Y data sets and $l = \min(k_x, k_y)$. These are asymptotically chi-square distributed with $(k_x - i)(k_y - i)$ degrees of freedom, so that the case $i = 0$ tests that none of the l correlations are significant, the case $i = 1$ tests that none of the remaining $l - 1$ correlations are significant, and so on. If any of these tests in sequence are not significant, then the remaining tests should, of course, be ignored.

Figure 13.6 illustrates two possible graphical displays for the canonical variates defined by `matrix.tf5`, where

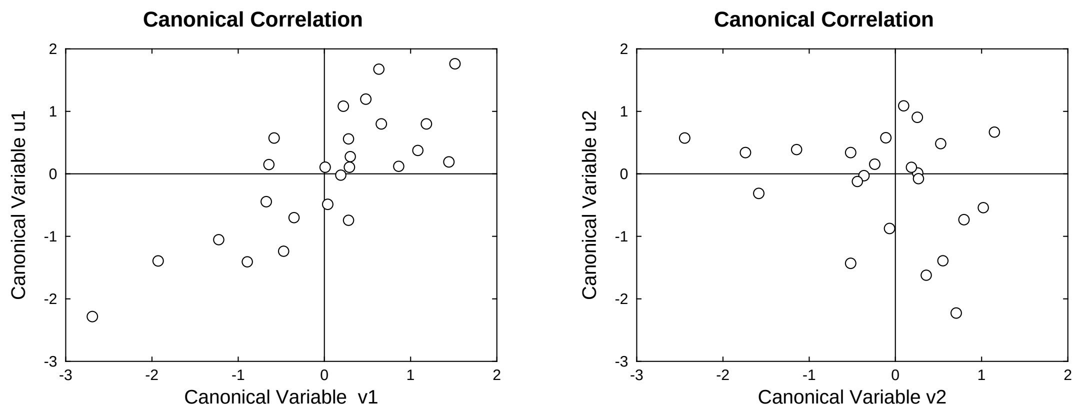


Figure 13.6: Canonical correlations for two groups

columns 1 and 2 are designated the Y sub-matrix, while columns 3 and 4 hold the X matrix. The canonical variates for X are constructed from the n_x by n_{cv} loading or coefficient matrix CVX, where $CVX(i, j)$ contains the loading coefficient for the ith x variable on the jth canonical variate u_j. Similarly $CVY)$ is the n_y by n_{cv} loading coefficient matrix for the ith y variable on the jth canonical variate v_j. More precisely, if cvx_j is column j of CVX, and cvy_j is column j of CVY, while $x(k)$ is the vector of centralized X observations for case k, and $y(k)$ is the vector of centralized Y observations for case k, then the components $u(k)_j$ and $v(k)_j$ of the n vector canonical variates u_j and v_j are

$$v(k)_j = cvx_j^T x(k), \quad k = 1, 2, \ldots, n$$
$$u(k)_j = cvy_j^T y(k), \quad k = 1, 2, \ldots, n.$$

It is important to realize that the canonical variates for U and V do not represent any sort of regression of Y on X, or X on Y, they are just new coordinates chosen to present the existing correlations between the original X and Y in a new space where the correlations are then ordered for convenience as

$$R^2(u_1, v_1) \ge R^2(u_2, v_2) \ge \ldots \ge R^2(u_l, v_l).$$

Clearly, the left hand plot shows the highest correlation, that is, between u_1 and v_1, whereas the right hand plot illustrates weaker correlation between u_2 and v_2. Note that further linear regression and correlation analysis can also be performed on the canonical variates if required, and also the loading matrices can be saved to construct canonical variates using the SimFIT matrix multiplication routines, and vectors of canonical variates can be saved directly from plots like those in figure 13.6.

13.6 Cluster analysis: calculating a distance matrix

The idea is, as in data mining, where you have a n by m matrix a_{ij} of m variables (columns) for each of n cases (rows) and wish to explore clustering, that is groupings together of like entities. To do this, you choose an appropriate pre-analysis transformation of the data, a suitable distance measure, a meaningful scaling procedure, and a sensible linkage function. SimFIT will then calculate a distance matrix, or a similarity matrix, and plot the clusters as a dendrogram. As an example, analyze the test file `cluster.tf1` giving the results displayed in table 13.7.

```
Variables included:-
 1 2 3 4 5 6 7 8
Transformation = Untransformed
Distance = Euclidean distance
Scaling = Unscaled
Linkage = Group average
Weighting [weights r not used]
Distance matrix (strict lower triangle) is:-
  2) 22.0
  3) 36.2 28.8
  4) 22.9 29.7 36.6
  5) 19.5 16.6 31.1 24.5
  6) 39.8 32.7 40.6 31.8 26.1
  7) 21.7 28.3 38.2 21.3 19.3 36.2
  8) 14.1 24.1 42.6 18.8 18.9 34.2 18.5
  9) 32.7 23.0 45.4 44.9 23.6 38.7 36.6 33.4
 10) 31.6 23.9 37.2 41.0 22.2 43.9 33.5 33.9 (+)
 10) 24.7
 11) 32.2 24.4 39.1 41.8 20.2 41.4 31.3 33.4 (+)
 11) 19.9 8.25
 12) 29.9 22.7 37.7 39.0 17.2 38.4 29.2 31.4 (+)
 12) 18.1 11.4 6.24
```

Table 13.7: Cluster analysis: distance matrix

The distance d_{jk} between objects j and k is just a chosen variant of the weighted L_p norm

$$d_{jk} = \{\sum_{i=1}^{m} w_{ijk} D(a_{ji}/s_i, a_{ki}/s_i)\}^p, \text{ for some } D, \text{ e.g.,}$$

(a) The Euclidean distance $D(\alpha,\beta) = (\alpha - \beta)^2$ with $p = 1/2$ and $w_{ijk} = 1$

(b) The Euclidean squared difference $D(\alpha,\beta) = (\alpha - \beta)^2$ with $p = 1$ and $w_{ijk} = 1$

(c) The absolute distance $D = |\alpha - \beta|$ with $p = 1$ and $w_{ijk} = 1$, otherwise known as the Manhattan or city block metric.

However, as the values of the variables may differ greatly in size, so that large values would dominate the analysis, it is usual to subject the data to a preliminary transformation or to apply a suitable weighting. Often it is best to transform the data to standardized $(0, 1)$ form before constructing the dendrogram, or at least to use some sort of scaling procedure such as:

(i) use the sample standard deviation as s_i for variable i,

(ii) use the sample range as s_i for variable i, or

(iii) supply precalculated values of s_i for variable i.

Bray-Curtis dissimilarity uses the absolute distance except that the weighting factor is given by

$$w_{ijk} = \frac{1}{\sum_{i=1}^{m}(a_{ji}/s_i + a_{ki}/s_i)}$$

which is independent of the variables i and only depends on the cases j and k, and distances are usually multiplied by 100 to represent percentage differences. Bray-Curtis similarity is the complement, i.e., 100 minus the dissimilarity. The Canberra distance measure, like the Bray-Curtis one, also derives from the absolute distance except that the weighting factor is now

$$w_{ijk} = \frac{1}{\lambda(a_{ji}/s_i + a_{ki}/s_i)}.$$

There are various conventions for defining λ and deciding what to do when values or denominators are zero with the Bray-Curtis and Canberra distance measures, and the scheme used by SIMFIT is as follows.

- If any values are negative the calculation is terminated.
- If any Bray-Curtis denominator is zero the calculation is terminated.
- If there are no zero values, then λ is equal to the number of variables in the Canberra measure.
- If both members of a pair are zero, then λ is decreased by one for each occurrence of such a pair, and the pairs are ignored.
- If one member of a pair is zero, then it is replaced by the smallest non-zero value in the data set divided by five, then scaled if required.

13.7 Cluster analysis: nearest neighbors

Once a distance matrix has been calculated, it is sometimes useful to calculate the nearest neighbors, as illustrated in table 13.8 for the data in table 13.7.

In this table, column 1 refers to the objects in logical order, column 2 indicates the object that is closest, i.e., the nearest neighbor, while column 3 records these minimum distances. Clearly, the nearest neighbors will depend upon the parameters used to configure the calculation of the distance matrix.

13.8 Cluster analysis: dendrograms

The test file `cluster.tf1` should be examined to see how to provide labels, as in figure 13.7, and further details about plotting dendrograms will now be discussed.

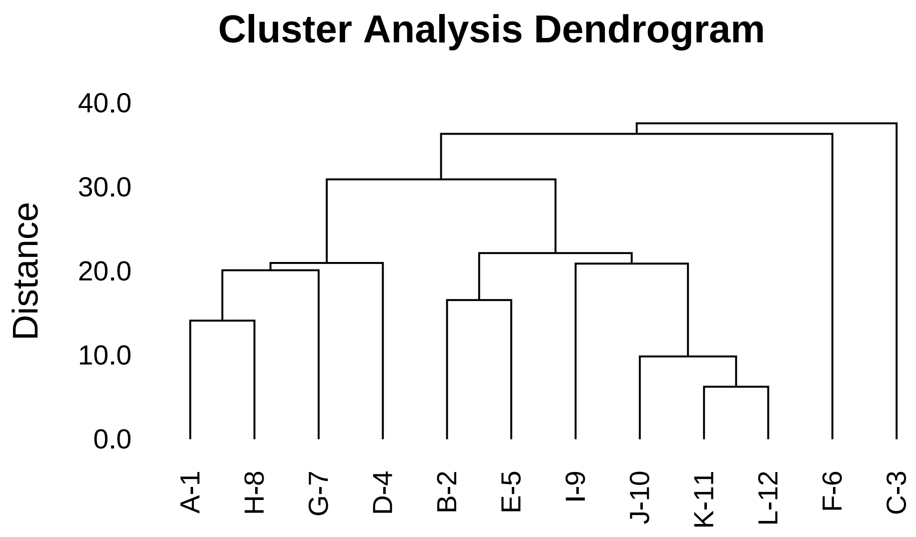


Figure 13.7: Dendrograms and multivariate cluster analysis

The shape of the dendrogram depends on the choice of analytical techniques and the order of objects plotted is arbitrary: groups at a given fixed distance can be rotated and displayed in either orientation. Another choice which will affect the dendrogram shape is the method used to recalculate distances after each merge has occurred. Suppose there are three clusters i, j, k with n_i, n_j, n_k objects in each cluster and let clusters j and k be merged to give cluster jk. Then the distance from cluster i to cluster jk can be calculated in several ways.

[1] Single link: $d_{i,jk} = \min(d_{ij}, d_{ik})$

[2] Complete link: $d_{i,jk} = \max(d_{ij}, d_{ik})$

Object	Nearest	Distance
1	8	14.1067
2	5	16.5529
3	2	28.7576
4	8	18.7617
5	2	16.5529
6	5	26.0960
7	8	18.4932
8	1	14.1067
9	12	18.1384
10	11	8.24621
11	12	6.24500
12	11	6.24500

Table 13.8: Cluster analysis: nearest neighbors

[3] Group average: $d_{i,jk} = (n_j d_{ij} + n_k d_{ik})/(n_j + n_k)$

[4] Centroid: $d_{i,jk} = (n_j d_{ij} + n_k d_{ik} - n_j n_k d_{jk}/(n_j + n_k))/(n_j + n_k)$

[5] Median: $d_{i,jk} = (d_{ij} + d_{ik} - d_{jk}/2)/2$

[6] Minimum variance: $d_{i,jk} = \{(n_i + n_j)d_{ij} + (n_i + n_k)d_{ik} - n_i d_{jk}\}/(n_i + n_j + n_k)$

An important application of distance matrices and dendrograms is in partial clustering. Unlike the situation with full clustering where we start with n groups, each containing a single case, and finish with just one group containing all the cases, in partial clustering the clustering process is not allowed to be completed. There are two distinct ways to arrest the clustering procedure.

1. A number, K, between 1 and $n - 1$ is chosen, and clustering is allowed to proceed until just K subgroups have been formed. It may not always be possible to satisfy this requirement, e.g. if there are ties in the data.

2. A threshold, D, is set somewhere between the first clustering distance and the last clustering distance, and clustering terminates when this threshold is reached. The position of such clustering thresholds will be plotted on the dendrogram, unless D is set equal to zero.

As an example of this technique consider the results in table 13.9. This resulted from analysis of the famous

```
Fisher's Iris data, 3 groups, All 4 variables, Untransformed,
Euclidean distance, Unscaled, Group average linkage,
[weights r not used], Dendrogram sub-clusters for K = 3
Odd rows: data ... Even rows: corresponding group number
  1    2    3    4    5    6    7    8    9   10   11   12
  1    1    1    1    1    1    1    1    1    1    1    1
 13   14   15   16   17   18   19   20   21   22   23   24
  1    1    1    1    1    1    1    1    1    1    1    1
 25   26   27   28   29   30   31   32   33   34   35   36
  1    1    1    1    1    1    1    1    1    1    1    1
 37   38   39   40   41   42   43   44   45   46   47   48
  1    1    1    1    1    1    1    1    1    1    1    1
 49   50   51   52   53   54   55   56   57   58   59   60
  1    1    2    2    2    2    2    2    2    2    2    2
 61   62   63   64   65   66   67   68   69   70   71   72
  2    2    2    2    2    2    2    2    2    2    2    2
 73   74   75   76   77   78   79   80   81   82   83   84
  2    2    2    2    2    2    2    2    2    2    2    2
 85   86   87   88   89   90   91   92   93   94   95   96
  2    2    2    2    2    2    2    2    2    2    2    2
 97   98   99  100  101  102  103  104  105  106  107  108
  2    2    2    2    2*   2*   3    2*   2*   3    2*   3
109  110 111    112 113  114  115  116  117  118  119  120
  2*     3      2*    2*   2*   2*   2*   2*   2*   3    3    2*
121    122    123   124   25  126  127  128  129  130  131  132
  2*     2*     3     2*   2*   3    2*   2*   2*   3    3    3
133    134    135   136   37  138  139  140  141  142  143  144
  2*     2*     2*    3    2*   2*   2*   2*   2*   2*   2*   2*
145    146    147   148   49  150
  2*     2*     2*    2*   2*   2*
```

Table 13.9: Cluster analysis: partial clustering for Iris data

Fisher iris data set in `iris.tf1` when $K = 3$ subgroups were requested. We note that groups 1 (setosa) and 2 (versicolor) contained the all the cases from the known classification, but most of the known group 3 (virginica) cases (those identified by asterisks) were also assigned to subgroup 2. This table should also be compared to table 13.12 resulting from K-means clustering analysis of the same data set. From the SimFIT dendrogram partial clustering procedure it is also possible to create a SimFIT MANOVA type file for any type of subsequent MANOVA analysis and, to aid in the use of dendrogram clusters as training sets for allocating new observations to groups, the subgroup centroids are also appended to such files. Alternatively a file ready for K-means cluster analysis can be saved, with group centroids appended to serve as starting estimates. Finally, attention should be drawn to the advanced techniques provided for plotting dendrogram thresholds and subgroups illustrated next.

13.8.1 Plotting dendrograms: standard format

Dendrogram shape is arbitrary in two ways; the x axis order is arbitrary as clusters can be rotated around any clustering distance leading to 2^{n-1} different orders, and the distance matrix depends on the settings used. For instance, a square root transformation, Bray-Curtis similarity, and a group average link generates the second dendrogram in figure 13.8 from the first. The y plotted are dissimilarities, while labels are $100 - y$, which should be remembered when changing the y axis range.

Users should not manipulate dendrogram parameters to create a dendrogram supporting some preconceived clustering scheme. You can set a label threshold and translation distance from the [X-axis] menu so that, if the number of labels exceeds the threshold, even numbered labels are translated, and font size is decreased.

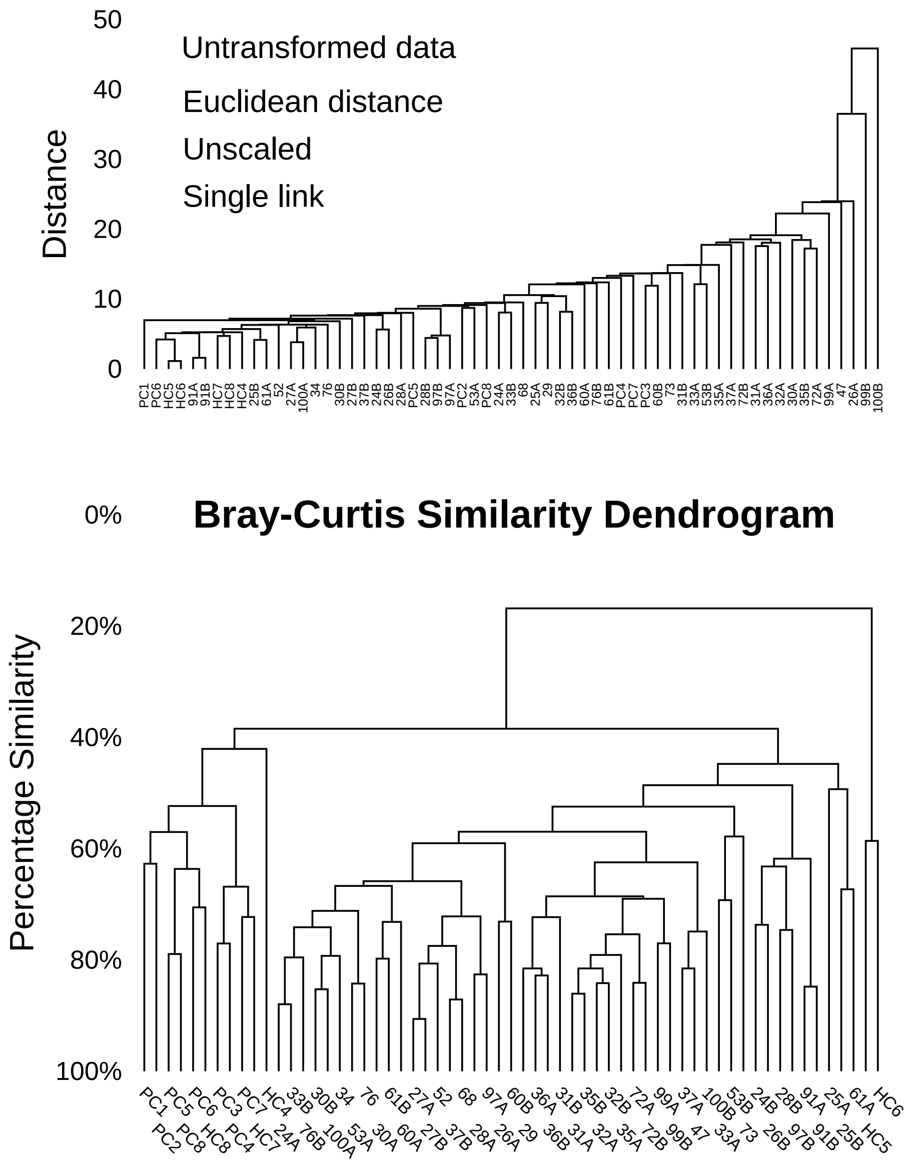


Figure 13.8: Plotting dendrograms: standard format

13.8.2 Plotting dendrograms: stretched format

Sometimes dendrograms are more readable if the white space is stretched without distorting the labels. So SɪᴍFɪT PostScript graphs have a very useful feature: you can stretch or compress the white space between plotted lines and symbols without changing the line thickness, symbol size, or font size and aspect ratio. For instance, stretching, clipping and sliding procedures are valuable in graphs which are crowded due to overlapping symbols or labels, as in figure 13.8. If such dendrograms are stretched retrospectively using **editps**, the labels will not separate as the fonts will also be stretched so letters become ugly due to altered aspect ratios. SɪᴍFɪT can increase white space between symbols and labels while maintaining correct aspect ratios for the fonts in PostScript hardcopy and, to explain this, the creation of figure 13.9 will be described.

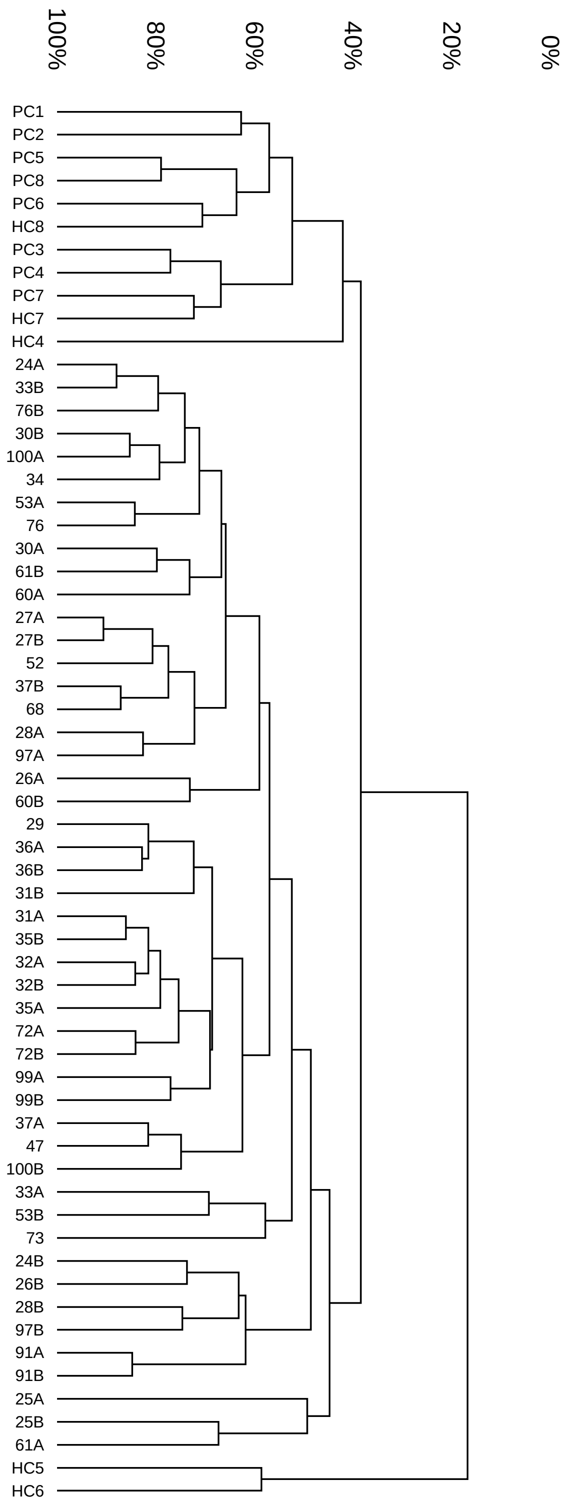


Figure 13.9: Plotting dendrograms: stretched format

The title, legend and double x labeling were suppressed, and landscape mode with stretching, clipping and sliding was selected from the PostScript control using the [Shape] then [Landscape +] options, with an x stretching factor of two. Stretching increases the space between each symbol, or the start of each character string, arrow or other graphical object, but does not turn circles into ellipses or distort letters. As graphs are often stretched to print on several sheets of paper, sub-sections of the graph can be clipped out, then the clipped sub-sections can be slid to the start of the original coordinate system to facilitate printing.

If stretch factors greater than two are used, legends tend to become detached from axes, and empty white space round the graph increases. To remedy the former complication, the default legends should be suppressed or replaced by more closely positioned legends while, to cure the later effect, GSview can be used to calculate new BoundingBox coordinates (by transforming .ps to .eps). If you select the option to plot an opaque background even when white (by mistake), you may then find it necessary to edit the resulting .eps file in a text editor to adjust the clipping coordinates (identified by %#clip in the .eps file) and background polygon filling coordinates (identified by %#pf in the .ps file) to trim away unwanted white background borders that are ignored by GSview when calculating BoundingBox coordinates. Another example of this technique is on page 189, where it is also pointed out that creating transparent backgrounds by suppressing the painting of a white background obviates the need to clip away extraneous white space.

13.8.3 Plotting dendrograms: subgroups

The procedure described on page 210 can also be used to improve the readability of dendrograms where subgroups have been assigned by partial clustering (page 207). Figure 13.10 shows a graph from `iris.tf1` when three subgroups are requested, or a threshold is set corresponding to the horizontal dotted line. Figure 13.11 was created by these steps.
First the title was suppressed, the y-axis range was changed to $(0, 4.25)$ with 18 tick marks, the (x, y) offset was canceled as this suppresses axis moving, the label font size was increased from 1 to 3, and the x-axis was translated to 0.8.
Then the PostScript stretch/slide/clip procedure was used with these parameters

$$
\begin{aligned}
x_{\text{stretch}} &= 1.5\\
y_{\text{stretch}} &= 2.0\\
x_{\text{clip}} &= 0.15, 0.95\\
y_{\text{clip}} &= 0.10, 0.60.
\end{aligned}
$$

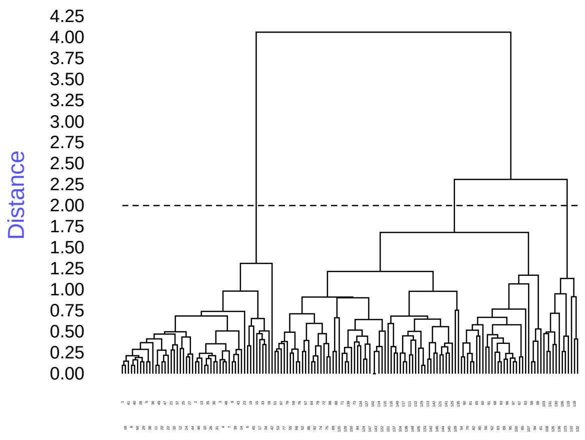


Figure 13.10: Plotting dendrograms: thresholds

Windows users without PostScript printing facilities must create a `*.eps` file using this technique, then use the SɪᴍFɪT procedures to create a graphics file they can use, e.g. `*.jpg`. Use of a larger font and increased x-stretching would be required to read the labels, of course.

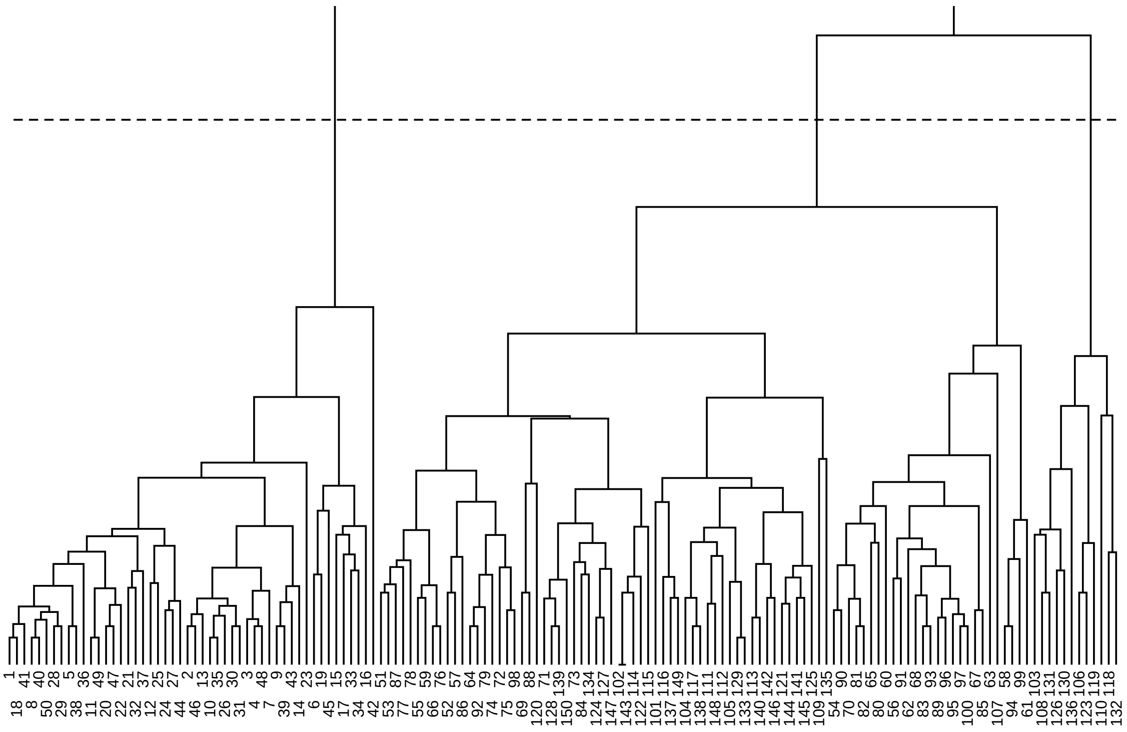


Figure 13.11: Plotting dendrograms: subgroups

13.9 Cluster analysis: classical metric scaling, MDS

Scaling techniques provide various alternatives to dendrograms for visualizing distances between cases, so facilitating the recognition of potential groupings in a space of lower dimension than the number of cases. For instance, once a distance matrix $D = (d_{ij})$ has been calculated for n cases with m variables, as described for dendrograms (page 206), it may be possible to calculate principal coordinates. This involves constructing a matrix E defined by

$$e_{ij} = -\tfrac{1}{2}(d_{ij}^2 - d_{i.}^2 - d_{.j}^2 + d_{..}^2),$$

where $d_{i.}^2$ is the average of d_{ij}^2 over the suffix j, etc., in the usual way. The idea is to choose an integer k, where $1 < k << n-1$, so that the data can be represented approximately in a space of dimension less than the number of cases, but in such a way that the distance between the points in that space correspond to the distances represented by the d_{ij} of the distance matrix as far as possible. If E is positive definite, then the ordered eigenvalues $\lambda_i > 0$ of E will be nonnegative and the proportionality expression

$$P = \sum_{i=1}^{k} \lambda_i \Big/ \sum_{i=1}^{n-1} \lambda_i$$

will show how well the cases of dimension n are represented in this subspace of dimension k. The most useful case is when $k = 2$, or $k = 3$, and the d_{ij} satisfy

$$d_{ij} \le d_{ik} + d_{jk},$$

so that a two or three dimensional plot will display distances corresponding to the d_{ij}. If this analysis is carried out but some relatively large negative eigenvalues result, then the proportion P may not adequately represent the success in capturing the values in distance matrix in a subspace of lower dimension that can be plotted meaningfully. It should be pointed out that the principal coordinates will actually be the same as the principal components scores when the distance matrix is based on Euclidean norms. Further, where metrical scaling succeeds, the distances between points plotted in say two or three dimensions will obey the triangle inequality and so correspond reasonably closely to the distances in the dissimilarity matrix, but if it fails it could be useful to proceed to non-metrical scaling, which is discussed next.

13.10 Cluster analysis: non-metric (ordinal) scaling

Often a distance matrix is calculated where some or all of the variables are ordinal, so that only the relative order is important, not the actual distance measure. Non-metric (i.e. ordinal) scaling is similar to the metric scaling previously discussed, except that the representation in a space of dimension $1 < k << n-1$ is sought in such a way as to attempt to preserve the relative orders, but not the actual distances. The closeness of a fitted distance matrix to the observed distance matrix can be estimated as either $STRESS$, or $SSTRESS$, given by

$$STRESS = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1}(\hat{d}_{ij} - \tilde{d}_{ij})^2}{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\hat{d}_{ij}^2}}$$

$$SSTRESS = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1}(\hat{d}_{ij}^2 - \tilde{d}_{ij}^2)^2}{\sum_{i=1}^{n}\sum_{j=1}^{i-1}\hat{d}_{ij}^4}},$$

where $\hat{d}_{ij}$ is the Euclidean squared distance between points i and j, and $\tilde{d}_{ij}$ is the fitted distance when the $\hat{d}_{ij}$ are monotonically regressed on the d_{ij}. This means that $\tilde{d}_{ij}$ is monotonic relative to d_{ij} and is obtained from $\hat{d}_{ij}$ with the smallest number of changes. This is a nonlinear optimization problem which may depend critically on starting estimates, and so can only be relied upon to locate a local, not a global solution. For this reason, starting estimates can be obtained in SimFIT by a preliminary metric scaling, or alternatively the values from such a scaling can be randomly perturbed before the optimization, in order to explore possible alternative solution points. Note that SimFIT can save distance matrices to files, so that dendrogram creation, classical metric, and non-metric scaling can be carried out retrospectively, without the need to generate distance matrices repeatedly from multivariate data matrices. Such distance matrices will be stored as vectors, corresponding to the strict lower triangle of the distance matrix packed by rows, (i.e. the strict upper triangle packed by columns). Table 13.10 tabulates the results from analyzing the distance matrix,

```
Eigenvalues from MDS (divided by the trace of the E matrix)
     0.78713
     0.28085
     0.15963
    0.074761
    0.031624
    0.020654
         0.0
   -0.012186
   -0.013685
   -0.030479
   -0.045469
   -0.056206
   -0.079207
    -0.11741
[Sum 1 to 2]/[sum 1 to 13] =  0.9558 (95.58%) (actual values)
[Sum 1 to 2]/[sum 1 to 13] =  0.6709 (67.09%) (absolute values)

  STRESS = 0.12557 (start = Metric 0%)
S-STRESS = 0.14962 (start = Metric 0%)
```

Table 13.10: Cluster analysis: metric and non-metric scaling

stored in the test file `g03faf.tf1`, by the metric, and also both non-metric techniques. This table first lists the eigenvalues from classical metric scaling, where each eigenvalue has been normalized by dividing by the sum of all the eigenvalues, then the $STRESS$ and $SSTRESS$ values are listed. Note that the type of starting estimates used, together with the percentages of the metric values used in any random starts, are output by SimFIT and it will be seen that, with this distance matrix, there are small but negative eigenvalues, and hence the proportion actually exceeds unity, and in addition two-dimensional plotting could be misleading. However it is usual to consider such small negative eigenvalues as being effectively zero, so that metric scaling in two dimensions is probably justified in this case. Figure 13.12 confirms this by showing considerable agreement between the two dimensional plots from metric scaling, and also non-metric scaling involving the $STRESS$ calculation. Note that the default labels in such plots may be integers corresponding to the case numbers, and not case labels, but such plot labels can be edited interactively, or overwritten from a labels file if required.

13.11 Cluster analysis: K-means clustering

Once a n by m matrix of values a_{ij} for n cases and m variables has been provided, the cases can be sub-divided into K non-empty clusters where $K < n$, provided that a K by m matrix of starting estimates b_{ij} has been specified. The procedure is iterative, and proceeds by moving objects between clusters to minimize the objective function

$$\sum_{k=1}^{K} \sum_{i \in S_k} \sum_{j=1}^{m} w_i (a_{ij} - \bar{a}_{kj})^2$$

where S_k is the set of objects in cluster k and $\bar{a}_{kj}$ is the weighted sample mean for variable j in cluster k. The weighting factors w_i can allow for situations where the objects may not be of equal value, e.g., if replicates have been used to determine the a_{ij}.

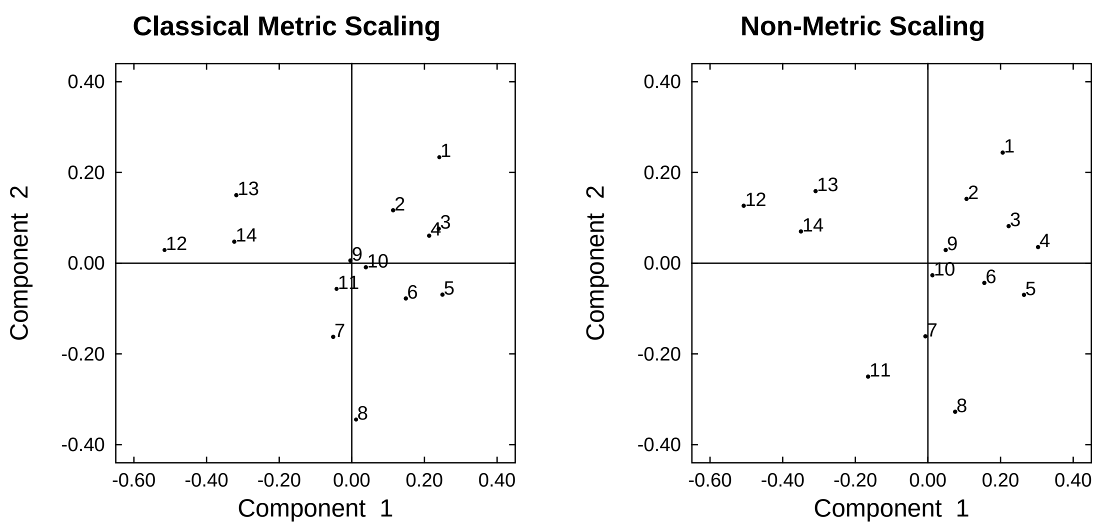


Figure 13.12: Classical metric and non-metric scaling

As an example, analyze the data in test file `g03eff.tf1` using the starting coordinates appended to this file, which are identical to the starting clusters in test file `g03eff.tf2`, to see the results displayed in table 13.11. Note that the final cluster centroids minimizing the objective function, given the starting estimates supplied,

```
Variables included:-
 1 2 3 4 5
Number of clusters = 3
Transformation = Untransformed
Weighting = Unweighted for replicates
Cases (odd rows) and Clusters (even rows)
    1    2    3    4    5    6    7    8    9   10   11   12
    1    1    3    2    3    1    1    2    2    3    3    3
   13   14   15   16   17   18   19   20
    3    3    3    3    3    1    1    3
 Final cluster centroids
      81.183     11.667       7.15       2.05        6.6
      47.867       35.8     16.333        2.4     6.7333
      64.045     25.209     10.745     2.8364     6.6545
```

Table 13.11: Cluster analysis: K-means clustering

are calculated, and the cases are assigned to these final clusters. Plots of the clusters and final cluster centroids can be created as in figure 13.13 for variables x_1 and x_2, with optional labels if these are supplied on the data file (as for dendrograms). With two dimensional data representing actual distances, outline maps can be added and other special effects can be created, as shown on page 218.

Table 13.12 illustrates analysis of the Fisher Iris data set in the file `iris.tf1`, using starting clusters in `iris.tf2`. It should be compared with table 13.9. The data were maintained in the known group order (as in `manova1.tf5`), and the clusters assigned are seen to be identical to the known classification for group 1 (setosa), while limited misclassification has occurred for groups 2 (versicolor, 2 assigned to group 3), and 3 (viginica, 14 assigned to group 2), as shown by the starred values. Clearly group 1 is distinct from groups 2 and 3 which show some similarities to each other, a conclusion also illustrated in figure 13.14, which should be

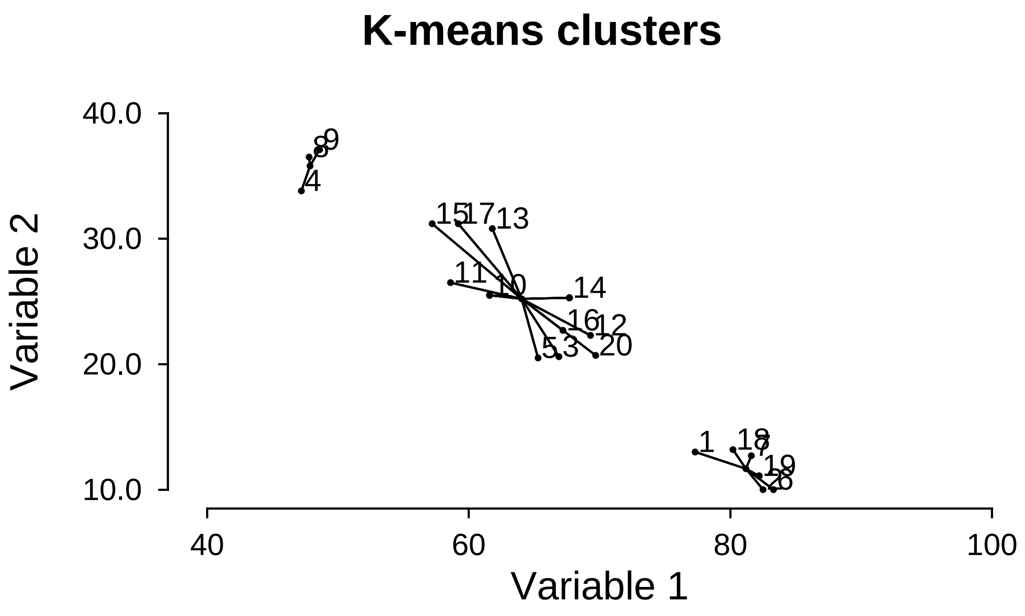


Figure 13.13: Plotting K-means clusters: individual labels

```
1    1    1    1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1    1    1    1
1    1    1    1    1    1    1    1    1    1    1    1
1    1    2    2    3*   2    2    2    2    2    2    2
2    2    2    2    2    2    2    2    2    2    2    2
2    2    2    2    2    3*   2    2    2    2    2    2
2    2    2    2    2    2    2    2    2    2    2    2
2    2    2    2    3    2*   3    3    3    3    2*   3
3    3    3    3    3    2*   2*   3    3    3    3    2*
3    2*   3    2*   3    3    2*   2*   3    3    3    3
3    2    3    3    3    3    2*   3    3    3    2*   3
3    3    2*   3    3    2*
Cluster    Size        WSSQ    Sum of weights
     1      50        15.15          50.0
     2      62        39.82          62.0
     3      38        23.88          38.0
Final cluster centroids
  5.006  3.4280  1.4620   0.246
 5.9016  2.7484  4.3935  1.4339
   6.85  3.0737  5.7421  2.0711
```

Table 13.12: K-means clustering for Iris data

compared with figure 13.24 using principal components (page 223) and canonical variates (page 233). It must be emphasized that in figure 13.14 the groups were generated by K-means clustering, while figure 13.24 was created using pre-assigned groups. Another difference is that the graph from clusters generated by K-means

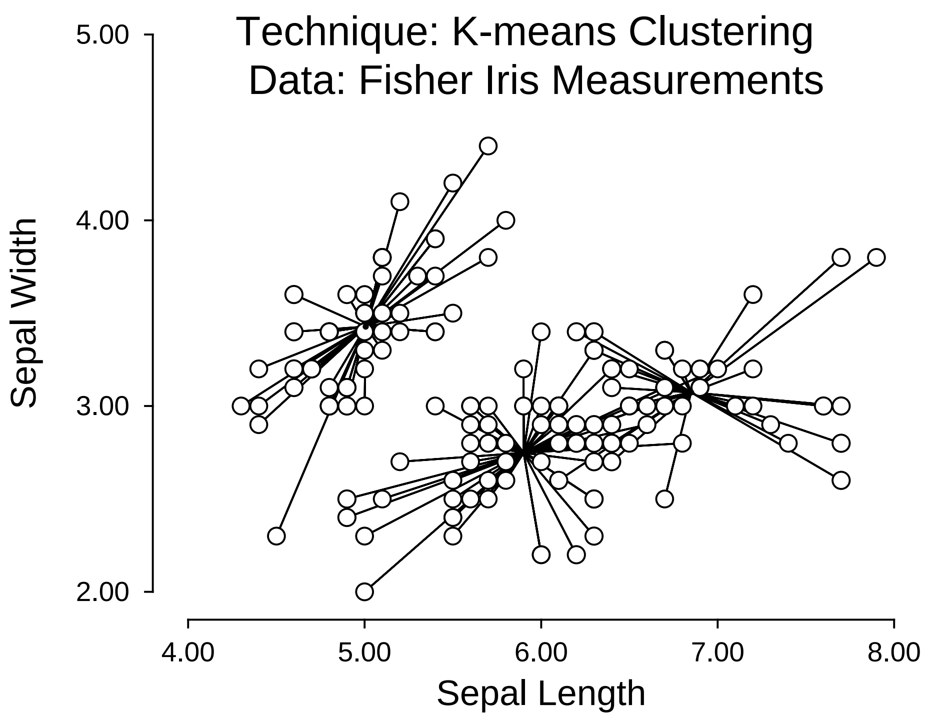


Figure 13.14: Plotting K-means clusters: Fisher Iris data

clustering is in the actual coordinates (or a transformation of the original coordinates) while the graphs in principal components or canonical variates are not in the original variables, but special linear combinations of the physical variables, chosen to emphasize features of the total data set.

Also note that in figure 13.14 there are data assigned to groups with centroids that are not nearest neighbors in the space plotted. This is because the clusters are assigned using the distances from cluster centroids when all dimensions are taken into account, not just the two plotted, which explains this apparent anomaly. Certain other aspects of the SimFIT implementation of K-means clustering should be made clear.

1. If variables differ greatly in magnitude, data should be transformed before cluster analysis but note that, if this is done interactively, the same transformation will be applied to the starting clusters. If a transformation cannot be applied to data, clustering will not be allowed at all, but if a starting estimate cannot be transformed (e.g., square root of a negative number), then that particular value will remain untransformed.

2. If, after initial assignment of data to the starting clusters some are empty, clustering will not start, and a warning will be issued to decrease the number of clusters requested, or edit the starting clusters.

3. Clustering is an iterative procedure, and different starting clusters may lead to different final cluster assignments. So, to explore the stability of a cluster assignment, you can perturb the starting clusters by adding or multiplying by a random factor, or you can even generate a completely random starting set. For instance, if the data have been normalized to zero mean and unit variance, then choosing uniform random starting clusters from $U(-1, 1)$, or normally distributed values from $N(0, 1)$ might be considered.

4. After clusters have been assigned you may wish to pursue further analysis, say using the groups for canonical variate analysis, or as training sets for allocation of new observations to groups. To do this,

you can create a SimFIT MANOVA type file with group indicator in column 1. Such files also have the centroids appended, and these can be overwritten by new observations (not forgetting to edit the extra line counter following the last line of data) for allocating to the groups as training sets.

5. If weighting, variable suppression, or interactive transformation is used when assigning K-means clusters, all results tables, plots and MANOVA type files will be expressed in coordinates of the transformed space.

13.11.1 Plotting K-Means clusters: UK airports

Stretching and clipping are also valuable when graphs have to be re-sized to achieve geometrically correct aspect ratios, as in the map shown in figure 13.15, which can be generated by the K-means clustering procedure using program **simstat** (see page 213) as follows.

- Input `ukmap.tf1` with coordinates for UK airports.
- Input `ukmap.tf2` with coordinates for starting centroids.
- Calculate centroids then transfer the plot to advanced graphics.
- Read in the UK coastal outline coordinates as an extra file from `ukmap.tf3`.
- Suppress axes, labels, and legends, then clip away extraneous white space.
- Stretch the PS output using the [Shape] then [Portrait +] options, and save the stretched eps file.

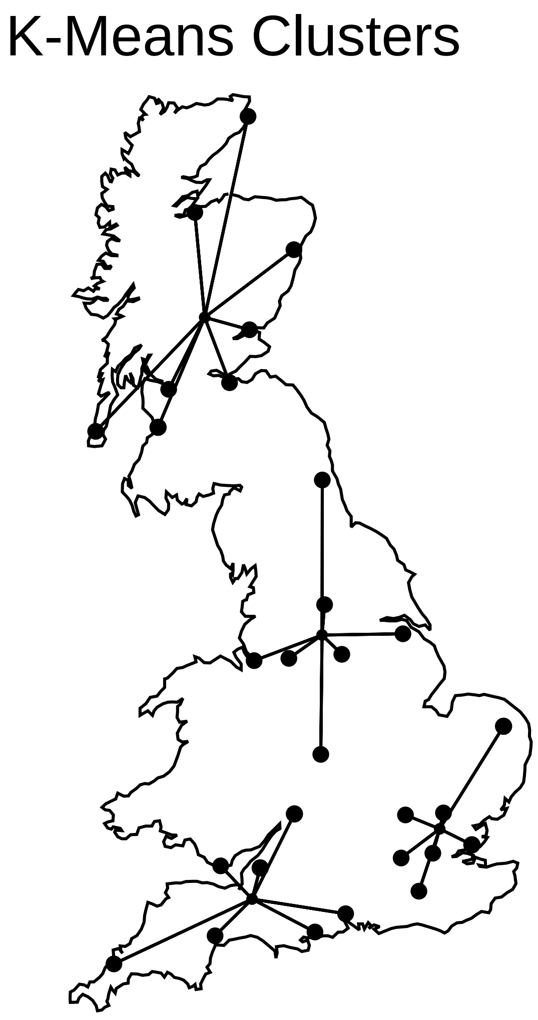


Figure 13.15: Plotting K-means clusters: UK airports

13.11.2 Plotting K-Means clusters: highlighting centroids

It is frequently useful to be able highlight groups of data points in a two dimensional swarm, as in figure 13.16.

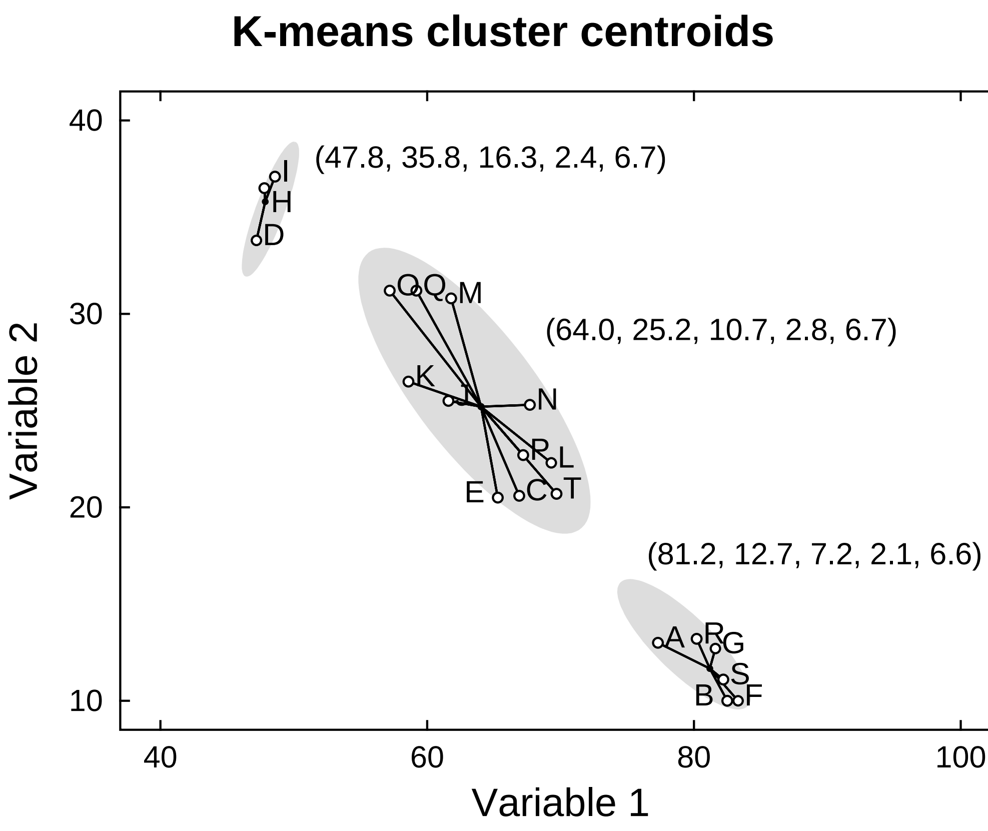


Figure 13.16: Plotting K-means clusters: highlighting centroids

In this case a partition into three groups has been done by K-means clustering, and to appreciate how to use this technique, note that figure 13.16 can be generated by the K-means clustering procedure using program **simstat** (see page 213) as follows.

- Input the K-means clustering test file `kmeans.tf1`.
- Calculate the centroids, using the starting estimates appended to the test file. View them, which then adds them to the results file, then record the centroid coordinates from the results file.
- Select to plot the groups with associated labels, but then it will prove necessary to move several of the labels by substituting new labels, or shifting the *x* or *y* coordinates to clarify the graph, as described on page 222.
- Add the solid background ellipses using the lines/arrows/boxes option because both head and tail coordinate must be specified using the red arrow, as well as an eccentricity value for the ellipses. Of course, any filled shapes such as circles, squares, or triangles can be chosen, and any size or color can be used.
- Add the centroid coordinates as extra text strings.

Of course, this technique can be used to highlight or draw attention to any subsets of data points, for instance groups in principal component analysis, to be discussed shortly.

13.11.3 Plotting K-Means clusters: variables or scores

Figure 13.17 illustrates the results from `kmeans.tf1`. Note that, in the upper figure, symbols F, S, and P have

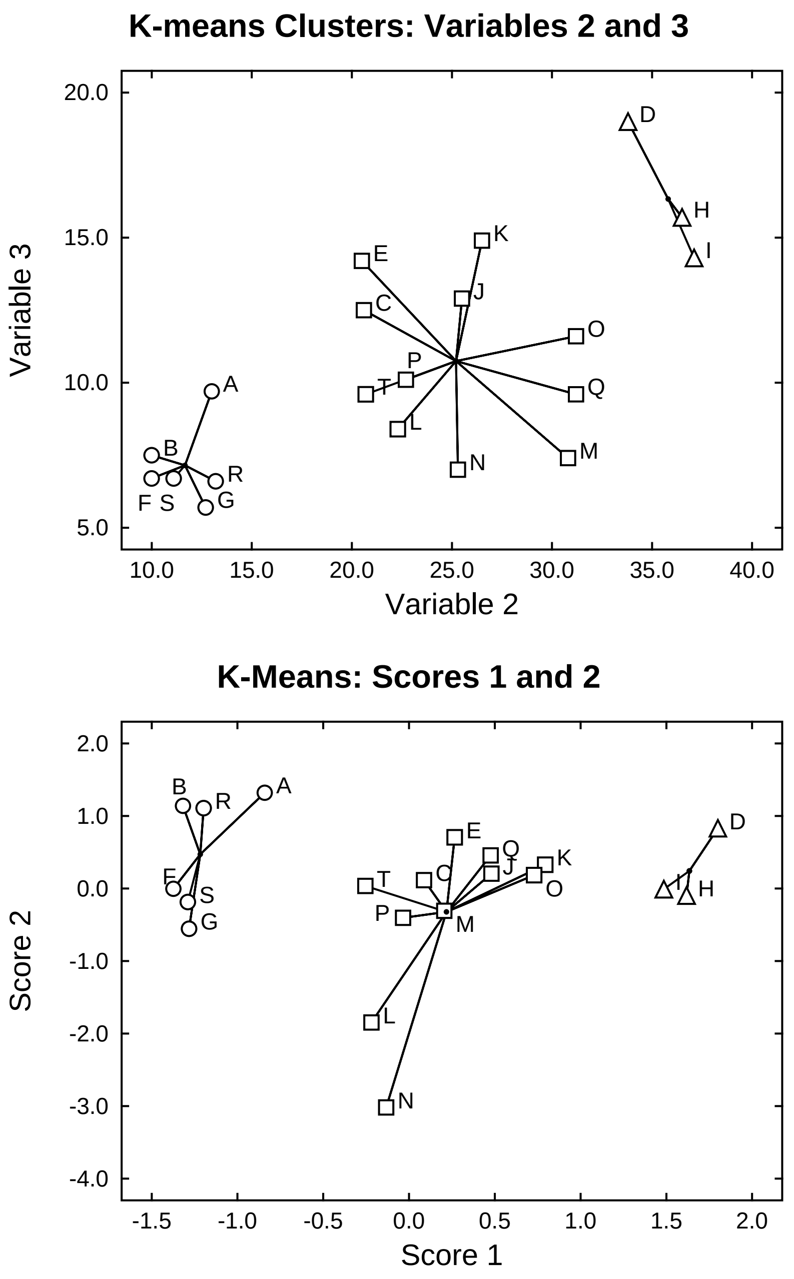


Figure 13.17: Plotting K-means clusters: variables or scores

been translated for clarity, and it should be compared to figure 13.13, for the same data with variables 1 and 2. This highlights an important point when plotting clusters for more than 2 variables: the plot shape depends on the variables chosen. So, for a more representative plot when there are more than 2 variables it is better to plot principal component scores instead of variables, which can be done interactively using the correlation matrix technique.

In the lower figure, symbols B, O, M, and P have been translated for clarity, but now the principal component scores 1 and 2 have been plotted, which will usually be a better representation of the clustering, as the shape of the plot is not so strongly influenced by the variables chosen.

13.12 Labeling multivariate plots

Labels are text strings (with associated template strings) that do not have arbitrary positions, but are plotted to identify the data. Some examples would be as follows.

- Labels adjacent to segments in a pie chart.
- Labels on the X axis to indicate groups in bar charts.
- Labels on the X axis to identify clusters in dendrograms (page 209).
- Labels plotted alongside symbols in 2D plots, such as principal components (page 222).

Test files such as `cluster.tf1` illustrate the usual way to supply labels appended to data files in order to over-ride the defaults set from the configuration options, but sometimes it is convenient to supply labels interactively from a file, or from the clipboard, and not all procedures in SIMFIT use the labels supplied appended to data files. Figure 13.18 illustrates this. Test file `cluster.tf1` was input into the procedure

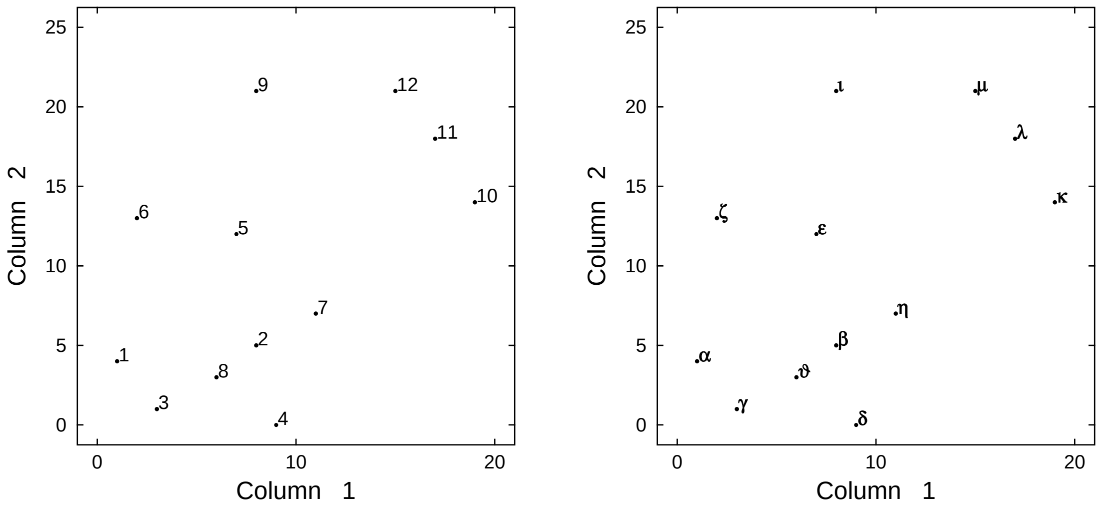
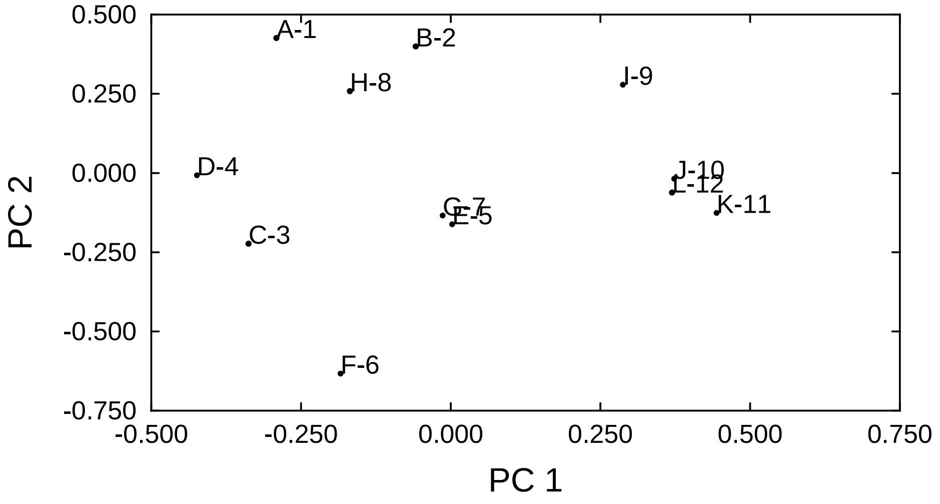


Figure 13.18: Labelling statistical graphs

for exhaustive analysis of a matrix in **simstat**, and the option to plot columns as an advanced 2D plot was selected. This created the left hand figure, where default integer labels indicate row coordinates. Then the option to add labels from a file was chosen, and test file `labels.txt` was input. This is just lines of characters in alphabetical order to overwrite the default integers. Then the option to read in a template was selected, and test file `templates.txt` was input. This just contains a succession of lines containing 6, indicating that alphabetical characters are to be plotted as bold maths symbols, resulting in the right hand figure. To summarize, the best way to manipulate labels in SIMFIT plots is as follows.

1. Write the column of case labels, or row of variable labels, from your data-base or spread-sheet program into an ASCII text file.
2. This file should just consist of one label per line and nothing else (like `labels.txt`)
3. Paste this file at the end of your SIMFIT data matrix file, editing the extra line counter (as in `cluster.tf1`) as required.
4. If there are n lines of data, the extra line counter (after the data but before the labels) must be at least n to use this label providing technique (See page 508).
5. Alternatively use the more versatile `begin{labels} ... end{labels}` technique (See page 508).



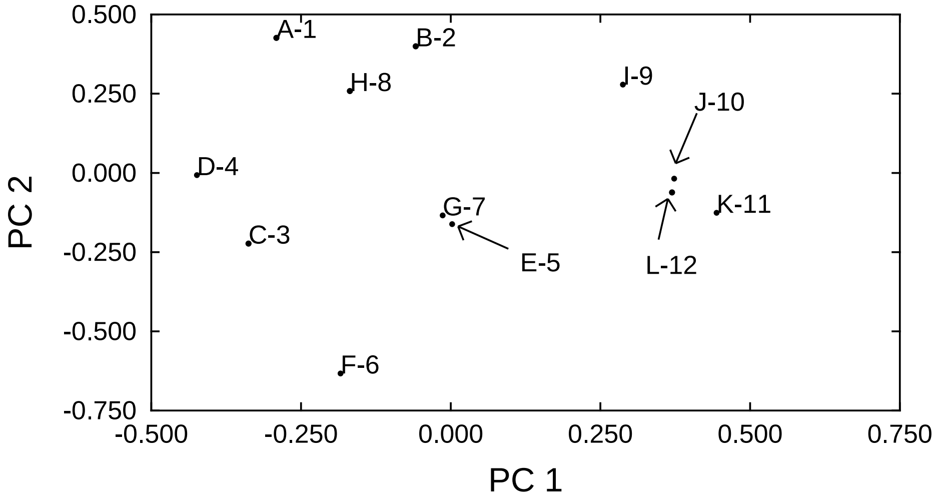


Figure 13.19: Principal components

6. Archive the labels file if interactive use is anticipated as in figure 13.18.
7. If Special symbols or accents are required, a corresponding templates file with character display codes (page 377) can be prepared.

13.13 Adjusting multivariate plot labels

Principal components for multivariate data can be explored by plotting scree diagrams and scattergrams after using the calculations options in program **simstat**. If labels are appended to the data file, as with `cluster.tf2`, they can be plotted, as in figure 13.19.
The labels that are usually plotted along the *x* axis are used to label the points, but moved to the side of the plotting symbol. Colors are controlled from the [Colour] options as these are linked to the color of the symbol plotted, even if the symbol is suppressed. The font is the one that would be used to label the *x* axis if labels were plotted instead of numbers. Clearly arbitrary labels cannot be plotted at the same time on the *x* axis.

Often it is required to move the labels because of clashes, as above. This is done by using the x axis editing function, setting labels that clash equal to blanks, then using the normal mechanism for adding arbitrary text and arrows to label the coordinates in the principal components scattergram. To facilitate this process, the default text font is the same as the axes numbering font. Alternatively, the plotting menus provide the option to move labels by defining parameters to shift individual labels horizontally or vertically.

13.14 Principal components analysis

In the principal components analysis of a n by m data matrix, new coordinates y are selected by rotation of the original coordinates x so that the proportion of the variance projected onto the new axes decreases in the order $y_1, y_2, \ldots, y_m$. The hope is that most of the variance can be accounted for by a subset of the data in y coordinates, so reducing the number of dimensions required for data analysis. Basing principal components analysis on the correlation matrix rather than the covariance or sum of squares and cross product matrices is often recommended as it prevents the analysis being unduly dominated by variables with large values, thus eliminating the need to centralize and scale the data. Remember that the loadings and scores are not unique but will be consistent subject to the parameters used for the analysis. To calculate scores from loadings the data matrix required must be mean centered and either unscaled if the covariance matrix is invoked, scaled by standard deviations if the correlation matrix method is used, or multiplied by $\sqrt{(n-1)}$ if the sum of squares and cross-products matrix is selected. The data format for principal components analysis is exactly the same as for cluster analysis; namely a data matrix with n rows (cases) and m columns (variables).

If the data matrix is X with covariance, correlation or scaled sum of squares and cross products matrix S, then the quadratic form

$$a_1^T S a_1$$

```
Variables included:-
 1 2 3
Transformation: Untransformed
Matrix type: Variance-covariance matrix
Score type: Score variance = eigenvalue
Replicates: Unweighted for replicates
Eigenvalues Proportion Cumulative   chi-sq   DOF   p
   8.274      0.6515     0.6515      8.613    5  0.1255
   3.676      0.2895     0.9410      4.118    2  0.1276
  0.7499      0.0590     1.0000        0.0    0  0.0000
Principal Component loadings (by column)
 -0.138  0.699   0.702
  -0.25  0.661  -0.707
  0.958  0.273 -0.0842
Principal Component scores (by column)
     -2.15     -0.173    -0.107
       3.8      -2.89     -0.51
     0.153     -0.987    -0.269
     -4.71        1.3    -0.652
      1.29       2.28    -0.449
       4.1      0.144     0.803
     -1.63      -2.23    -0.803
      2.11       3.25     0.168
    -0.235      0.373    -0.275
     -2.75      -1.07      2.09
```

Table 13.13: Principal components analysis

is maximized subject to the normalization $a_1^T a_1 = 1$ to give the first principal component

$$c_1 = \sum_{i=1}^{m} a_{1i} x_i.$$

Similarly, the quadratic form

$$a_2^T S a_2$$

is maximized, subject to the normalization and orthogonality conditions $a_2^T a_2 = 1$ and $a_2^T a_1 = 0$, to give the second principal component

$$c_2 = \sum_{i=1}^{m} a_{2i} x_i$$

and so on. The vectors a_i are the eigenvectors of S with eigenvalues λ_i^2, where the proportion of the variation accounted for by the ith principal component can be estimated as

$$\lambda_i^2 / \sum_{j=1}^{m} \lambda_j^2.$$

Actually SimFIT uses a singular value decomposition (SVD) of a centered and scaled data matrix, say $X_s = (X - \bar{X})/\sqrt{(n-1)}$ as in

$$X_s = V \Lambda P^T$$

to obtain the diagonal matrix Λ of singular values, the matrix of left singular vectors V as the n by m matrix of scores, and the matrix of right singular vectors P as the m by m matrix of loadings.

Table 13.13 shows analysis of the data in `g03aaf.tf1`, where column j of the loading matrix contains the coefficients required to express y_j as linear function of the variables $x_1, x_2, \ldots, x_m$, and row i of the scores matrix contains the values for row i of the original data expressed in variables $y_1, y_2, \ldots, y_m$. In this instance the data were untransformed and the variance covariance matrix was used to illustrate the statistical test for relevant eigenvalues but, where different types of variables are used with widely differing means and variances it is more usual to analyze the correlation matrix, instead of the covariance matrix, and many other scaling options and weighting options are available for more experienced users.

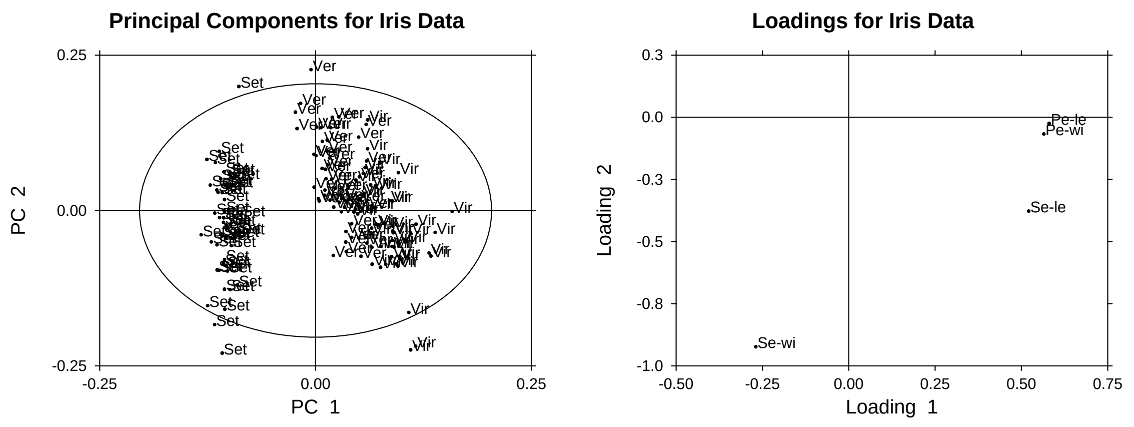


Figure 13.20: Principal component scores and loadings

Figure 13.20 shows the scores and loadings for the data in test file `iris.tf1` plotted as a scattergram after analyzing the correlation matrix. The score plot displays the score components for all samples using the selected principal components, so some may prefer to label the legends as principal components instead of scores, and this plot is used to search for possible groupings among the sample. The loading plot displays

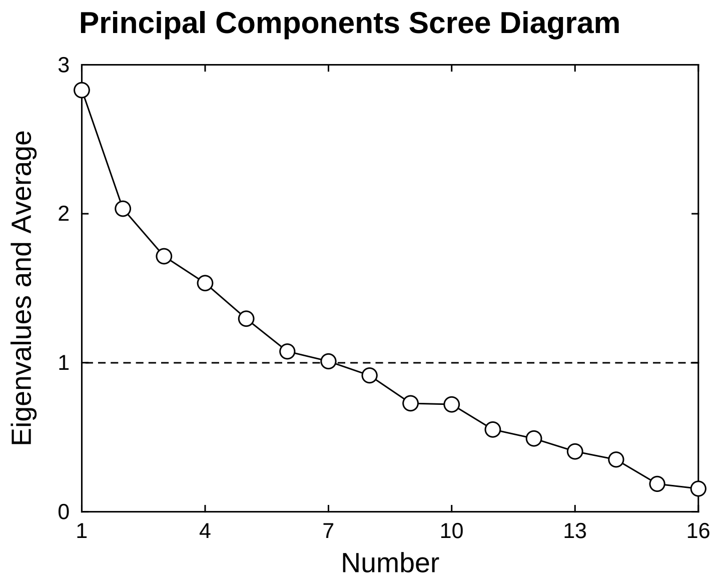


Figure 13.21: Principal components scree diagram

the coefficients that express the selected principal components y_j as linear functions of the original variables $x_1, x_2, \ldots, x_m$, so this plot is used to observe the contributions of the original variables x to the new ones y. Note that a 95% confidence Hotelling T^2 ellipse is also plotted, which assumes a multivariate normal distribution for the original data and uses the F distribution. The confidence ellipse is based on the fact that, if $\bar{y}$ and S are the estimated mean vector and covariance matrix from a sample of size n and, if x is a further independent sample from an assumed $p-$variate normal distribution, then

$$(x - \bar{y})^T S^{-1} (x - \bar{y}) \sim \frac{p(n^2 - 1)}{n(n - p)} F_{p,n-p},$$

where the significance level for the confidence region can be altered interactively. The components can be labeled using any labels supplied at the end of the data, but this can cause confusion where, as in the present case, the labels overlap leading to crowding. A method for moving labels to avoid such confusion is provided, as illustrated on page 222. However, with such dense labels it is best to just plot the scores using different symbols for the three groups, as shown for comparison with canonical variates analysis of the same data set in figure 13.24. Note that figure 13.20 also illustrates an application of the SimFIT technique for adding extra data interactively to create the cross-hairs intersecting at $(0, 0)$, and it also shows how labels can be added to identify the variables in a loadings plot. It should be noted that, as the eigenvectors are of indeterminate sign and only the relative magnitudes of coefficients are important, the scattergrams can be plotted with either the scores calculated from the SVD, or else with the scores multiplied by minus one, which is equivalent to reversing the direction of the corresponding axis in a scores or loadings plot.

An important topic in this type of analysis is deciding how to choose a sufficient number of principal components to represent the data adequately. As the eigenvalues are proportional to the fractions of variance along the principal component axes, a table of the cumulative proportions is calculated, and some users may find it useful to include sufficient principal components to account for a given amount of the variance, say 70%. Figure 13.21 shows how scree plots can be displayed to illustrate the number of components needed to represent the data adequately. For instance, in this case, it seems that approximately half of the principal components are required. A useful rule of thumb for selecting the minimum number of components is to observe where the scree diagram crosses the average eigenvalue or becomes flattened indicating that all subsequent eigenvalues contribute to a comparable extent. In cases where the correlation matrix is not used, a chi-square test statistic is also provided along with appropriate probability estimates to make the decision

```
X-data for rotation: g03bcf.tf1
Y-data for target: g03bcf.tf2
No. of rows 3, No. of columns 2
Type: To origin then Y-centroid
Scaling: Scaling
Alpha = 1.5563E+00
Residual sum of squares = 1.9098E-02
Residuals from Procrustes rotation
  0.096444
  0.084554
  0.051449
Rotation matrix from Procrustes rotation
     0.96732     0.25357
    -0.25357     0.96732
Y-hat matrix from Procrustes rotation
   -0.093442    0.023872
      1.0805    0.025918
    0.012959      1.9502
```

Table 13.14: Procrustes analysis

more objective. In this case, if k principal components are selected, the chi-square statistic

$$(n-1-(2m+5)/6)\left\{-\sum_{i=k+1}^{m}\log(\lambda_i^2)+(m-k)\log\left(\sum_{i=k+1}^{m}\lambda_i^2/(m-k)\right)\right\}$$

with $(m-k-1)(m-k+2)/2$ degrees of freedom can be used to test for the equality of the remaining $m-k$ eigenvalues. If one of these test statistics, say the $k+1$th, is not significant then it is usual to assume k principal components should be retained and the rest regarded as of little importance. So, if it is concluded that the remaining eigenvalues are of comparable importance, then a decision has to be made whether to eliminate all or preserve all. For instance, from the last column of p values referring to the above chi-square test in table 13.13, it might be concluded that a minimum of two components are required to represent this data set adequately. The common practise of always using two or three components just because these can be visualized is to be deplored.

13.15 Procrustes analysis

This technique is useful when there are two matrices X and Y with the same dimensions, and it wished to see how closely the X matrix can be made to fit the target matrix Y using only distance preserving transformations, like translation and rotation. For instance, X could be a matrix of loadings, and the target matrix Y could be a reference matrix of loadings from another data set. Table 13.14 illustrates the outcome from analyzing data in the test files `g03bcf.tf1` with X data to be rotated, and `g03bcf.tf2` containing the target matrix Y. First the centroids of X and Y are translated to the origin to give X_c and Y_c. Then the matrix of rotations R that minimize the sum of squared residuals is found from the singular value decomposition as

$$X_c^T Y_c = UDV^T$$
$$R = UV^T,$$

and after rotation a dilation factor α can be estimated by least squares, if required, to give the estimate

$$\hat{Y}_c = \alpha X_c R.$$

Additional options include normalizing both matrices to have unit sums of squares, normalizing the X matrix to have the same sum of squares as the Y matrix, and translating to the original Y centroid after rotation. Also,

```
Number of rows 10, Number of columns 3
Type: Unstandardised, Scaling: Varimax, Gamma = 1
Data for G03BAF
      0.7881      -0.152      -0.352
       0.874       0.381       0.041
       0.814      -0.043      -0.213
       0.798       -0.17      -0.204
       0.641        0.07      -0.042
       0.755      -0.298       0.067
       0.782      -0.221       0.028
       0.767      -0.091       0.358
       0.733      -0.384       0.229
       0.771      -0.101       0.071
Rotation matrix ... Varimax
  0.63347 -0.53367 -0.56029
  0.75803  0.57333  0.31095
  0.15529 -0.62169  0.76772
Rotated matrix ... Varimax
  0.32929  -0.2.8884  -0.75901
  0.84882  -0.2.7348  -0.33974
  0.44997  -0.3.2664  -0.63297
  0.34496  -0.3.9651  -0.65659
  0.45259  -0.2.7584  -0.36962
  0.26278  -0.6.1542  -0.46424
  0.33219  -0.5.6144  -0.48537
  0.47248  -0.6.8406  -0.18319
  0.20881  -0.7.5370  -0.35429
  0.42287  -0.5.1350  -0.40888
```

Table 13.15: Varimax rotation

as well as displaying the residuals, the sum of squares, the rotation and best fit matrices, options are provided to plot arbitrary rows or columns of these matrices.

13.16 Varimax and Quartimax rotation

Generalized orthomax rotation techniques can be used to simplify the interpretation of loading matrices, e.g. from canonical variates or factor analysis. These are only unique up to rotation so, by applying rotations according to stated criteria, different contributions of the original variables can be assessed. Table 13.15 illustrates how this analysis is performed using the test file `g03baf.tf1`. The input loading matrix Λ has m rows and k columns and results from the analysis of an original data matrix with n rows (i.e. cases) and m columns (i.e. variables), where k factors have been calculated for $k \le m$. If the input loading matrix is not standardized to unit length rows, this can be done interactively. The rotated matrix Λ^* is calculated so that the elements λ^*_{ij} are either relatively large or small. This involves maximizing the function

$$V = \sum_{j=1}^{k}\sum_{i=1}^{m}(\lambda^*_{ij})^4 - \frac{\gamma}{m}\sum_{j=1}^{k}\left[\sum_{i=1}^{m}(\lambda^*_{ij})^2\right]^2$$

for one of several cases as follows

- Varimax rotation: $\gamma = 1$
- Quartimax rotation: $\gamma = 0$.

- Equamax rotation: $\gamma = k/2$.
- Parsimax rotation: $\gamma = p(k-1)/(p+k+2)$.
- User chosen rotation: γ input.

The resulting rotation matrix R satisfies $\Lambda^* = \Lambda R$ and, when the matrices have been calculated they can be viewed, written to the results log file, saved to a text file, or plotted.

13.17 Multivariate analysis of variance (MANOVA)

Sometimes a designed experiment is conducted in which more than one response is measured at each treatment, so that there are two possible courses of action.

1. Do a separate ANOVA analysis for each variable.
   The disadvantages of this approach are that it is tedious, and also it relies upon the questionable assumption that each variable is statistically independent of every other variable, with a fixed variance for each variable. The advantages are that the variance ratio tests are intuitive and unambiguous, and also there is no requirement that sample size per group should be greater than the number of variables.

2. Do an overall MANOVA analysis for all variables simultaneously.
   The disadvantages of this technique are that it relies on the assumption of a multivariate normal distribution with identical covariance matrices across groups, it requires a sample size per group greater than the number of variables, and also there is no unique and intuitive best test statistic. Further, the power will tend to be lower than the power of the corresponding ANOVA. The advantages are that analysis is compact, and several useful options are available which simplify situations like the analysis of repeated measurements.

Central to a MANOVA analysis are the assumptions that there are n observations of a random m dimensional vector divided into g groups, each with n_i observations, so that $n = \sum_{i=1}^{g} n_i$ where $n_i \ge m$ for $i = 1, 2, \ldots, g$. If y_{ij} is the m vector for individual j of group i, then the sample mean $\bar{y}_i$, corrected sum of squares and products matrix C_i, and covariance matrix S_i for group i are

$$\bar{y}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$$

$$C_i = \sum_{j=1}^{n_i}(y_{ij} - \bar{y}_i)(y_{ij} - \bar{y}_i)^T$$

$$S_i = \frac{1}{n_i - 1} C_i .$$

For each ANOVA design there will be a corresponding MANOVA design in which corrected sums of squares and product matrices replace the ANOVA sums of squares, but where other test statistics are required in place of the ANOVA F distributed variance ratios. This will be clarified by dealing with typical MANOVA procedures, such as testing for equality of means and equality of covariance matrices across groups.

MANOVA example 1. Testing for equality of all means

Source of variation	d.f.	ssp matrix
Between groups	$g-1$	B
Within groups	$n-g$	W
Total	$n-1$	T

Table 13.16: MANOVA example 1a. Typical one way MANOVA layout

If all groups have the same multivariate normal distribution, then estimates for the mean μ and covariance matrix Σ can be obtained from the overall sample statistics $\hat{\mu} = \bar{y}$ and $\hat{\Sigma}$

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{g}\sum_{j=1}^{n_i} y_{ij}$$

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{g}\sum_{j=1}^{n_i}(y_{ij}-\hat{\mu})(y_{ij}-\hat{\mu})^T$$

obtained by ignoring group means $\bar{y}_i$ and summing across all groups. Alternatively, the pooled between-groups B, within-groups W, and total sum of squares and products matrices T can be obtained along with the within-groups covariance matrix S using the group mean estimates $\bar{y}_i$ as

$$\begin{aligned} B &= \sum_{i=1}^{g} n_i(\bar{y}_i-\bar{y})(\bar{y}_i-\bar{y})^T \\ W &= \sum_{i=1}^{g}\sum_{j=1}^{n_i}(y_{ij}-\bar{y}_i)(y_{ij}-\bar{y}_i)^T \\ &= \sum_{i=1}^{g}(n_i-1)S_i \\ &= (n-g)S \\ T &= B+W \\ &= (n-1)\hat{\Sigma}. \end{aligned}$$

Table 13.16 is typical, and clearly strong differences between groups will be indicated if B is much larger than W. The usual likelihood ratio test statistic is Wilk's lambda defined as

$$\Lambda = \frac{|W|}{|B|+|W|}$$

but other statistics can also be defined as functions of the eigenvalues of BW^{-1}. Unlike B and W separately, the matrix BW^{-1} is not symmetric and positive definite but, if the m eigenvalues of BW^{-1} are θ_i, then Wilk's lambda, Roy's largest root R, the Lawley-Hotelling trace T, and the Pillai trace P can be defined as

$$\begin{aligned} \Lambda &= \prod_{i=1}^{m}\frac{1}{1+\theta_i} \\ R &= \max(\theta_i) \\ T &= \sum_{i=1}^{m}\theta_i \\ P &= \sum_{i=1}^{m}\frac{\theta_i}{1+\theta_i}. \end{aligned}$$

Table 13.17 resulted when `manova1.tf3` was analyzed and the methods used to calculate the significance levels will be outlined. Table 13.18 indicates conditions on the number of groups g, variables m, and total number of observations n that lead to exact F variables for appropriate transforms of Wilk's Λ. For other conditions the asymptotic expression

$$-\left(\frac{2n-2-m-g}{2}\right)\log\Lambda \sim F_{m,g-1}$$

is generally used. The Lawley-Hotelling trace is a generalized Hotelling's T_0^2 statistic, and so the null distribution of this can be approximated as follows.

```
MANOVA H0: all mean vectors are equal
Number of groups       = 3
Number of variables    = 2
Number of observations = 15

Statistic              Value  Transform  Deg. Freedom     p
Wilks lambda           0.1917   7.062       4   22      0.0008 Reject H0 at 1%
Roys largest root       2.801
Lawley-Hotelling T      3.173   8.727       4   11      0.0017 Reject H0 at 1%
Pillais trace      0.00000001
```

Table 13.17: MANOVA example 1b. Test for equality of all means

Parameters	F statistic	Degrees of freedom
$g = 2,$ any m	$\dfrac{(2g-m-1)(1-\Lambda)}{m\Lambda}$	$m, 2g-m-1$
$g = 3,$ any m	$\dfrac{(3g-m-2)(1-\sqrt{\Lambda})}{m\sqrt{\Lambda}}$	$2m, 2(n-m-2)$
$m = 1,$ any g	$\dfrac{(n-g)(1-\Lambda)}{(g-1)\Lambda}$	$g-1, n-g$
$m = 2,$ any g	$\dfrac{(n-g-1)(1-\sqrt{\Lambda})}{(g-1)\sqrt{\Lambda}}$	$2(g-1), 2(n-g-1)$

Table 13.18: MANOVA example 1c. The distribution of Wilk's Λ

Defining the degrees of freedom and multiplying factors α and β by

$$\begin{aligned}
\nu_1 &= g-1\\
\nu_2 &= n-g\\
\nu &= \frac{m\nu_1(\nu_2-m)}{\nu_1+\nu_2-m\nu_1-1}\\
\alpha &= \frac{(\nu_2-1)(\nu_1+\nu_2-m-1)}{(\nu_2-m)(\nu_2-m-1)(\nu_2-m-3)}\\
\beta &= \frac{m\nu_1}{\nu_2-m+1},
\end{aligned}$$

then the case $\nu > 0$ leads to the approximation

$$T \sim \beta F_{\nu,\nu_2-m+1},$$

otherwise the alternative approximation

$$T \sim \alpha\chi^2_f$$

is employed, where $f = m\nu_1/\{\alpha(\nu_2-m-1)\}$. The null distributions for Roy's largest root and Pillai's trace are more complicated to approximate, which is one reason why Wilk's Λ is the most widely used test statistic.

MANOVA example 2. Testing for equality of selected means

Table 13.19 resulted when groups 2 and 3 were tested for equality, another example of a Hotelling's T^2 test. The first result uses the difference vector $d_{2,3}$ between the means estimated from groups 2 and 3 with the matrix $W = (n-g)S$ estimated using the pooled sum of squares and products matrix to calculate and test T^2

```
MANOVA H0: selected group means are equal
First group             = 2  (5 cases)
Second group            = 3  (5 cases)
Number of observations  = 15 (to estimate CV)
Number of variables     = 2
Hotelling T^2           = 12.0
Test statistic S        = 5.498
Numerator DOF           = 2
Denominator DOF         = 11
p = P(F >= S)           = 0.0221 Reject H0 at 5% sig.level

MANOVA H0: selected group means are equal
First group             = 2  (5 cases)
Second group            = 3  (5 cases)
Number of observations  = 10 (to estimate CV)
Number of variables     = 2
Hotelling T^2           = 15.18
Test statistic S        = 6.64
Numerator DOF           = 2
Denominator DOF         = 7
p = P(F >= S)           = 0.0242 Reject H0 at 5% sig.level
```

Table 13.19: MANOVA example 2. Test for equality of selected means

according to

$$T^2 = \left(\frac{(n-g)n_2n_3}{n_2+n_3}\right) d_{2,3}^T W^{-1} d_{2,3}$$
$$\frac{n-g-m+1}{m(n-g)}T^2 \sim F_{m,n-g-m+1},$$

while the second result uses the data from samples 2 and 3 as if they were the only groups as follows

$$S_{2,3} = \frac{(n_2-1)S_2 + (n_3-1)S_3}{n_2+n_3-2}$$
$$T^2 = \left(\frac{n_2n_3}{n_2+n_3}\right) d_{2,3}^T S_{2,3}^{-1} d_{2,3}$$
$$\frac{n_2+n_3-m-1}{m(n_2+n_3-2)}T^2 \sim F_{m,n_2+n_3-m-1}.$$

The first method could be used if all covariance matrices are equal (see next) but the second might be preferred if it was only likely that the selected covariance matrices were identical.

MANOVA example 3. Testing for equality of all covariance matrices

Table 13.20 shows the results from using Box's test to analyze `manova1.tf2` for equality of covariance matrices. This depends on the likelihood ratio test statistic C defined by

$$C = M\left\{(n-g)\log|S| - \sum_{i=1}^{g}(n_i-1)\log|S_i|\right\},$$

where the multiplying factor M is

$$M = 1 - \frac{2m^2+3m-1}{6(m+1)(g-1)}\left(\sum_{i=1}^{g}\frac{1}{n_i-1} - \frac{1}{n-g}\right)$$

```
MANOVA H0: all covariance matrices are equal

Number of groups         = 3
Number of observations   = 21
Number of. variables     = 2
Test statistic C         = 1.924
Degrees of Freedom       = 6
p = P(chi-sqd. >= C)     = 0.0038 Reject H0 at 1% sig.level
```

Table 13.20: MANOVA example 3. Test for equality of all covariance matrices

and, for large n, C is approximately distributed as χ^2 with $m(m+1)(g-1)/2$ degrees of freedom. Just as tests for equality of variances are not very robust, this test should be used with caution, and then only with large samples, i.e. $n_i >> m$.

MANOVA example 4. Profile analysis

Figure 13.22 illustrates the results from plotting the group means from `manova1.tf1` using the profile analysis

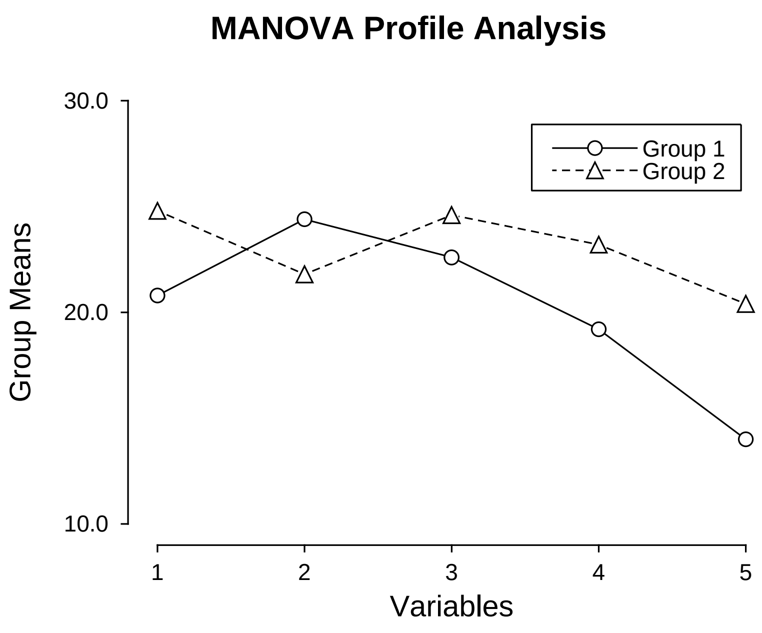


Figure 13.22: MANOVA profile analysis

option, noting that error bars are not added as a multivariate distribution is assumed, while table 13.21 shows the results of the statistical analysis. Profile analysis attempts to explore a common question that often arises in repeated measurements ANOVA namely, can two profiles be regarded as parallel. This amounts to testing if the sequential differences between adjacent means for groups i and j are equal, that is, if the slopes between adjacent treatments are constant across the two groups, so that the two profiles represent a common shape. To

```
MANOVA H0: selected group profiles are equal

First group             = 1  (5 cases)
Second group            = 2  (5 cases)
Number of observations  = 10 (to estimate CV)
Number of variables     = 5
Hotelling T^2           = 35.65
Test statistic S        = 5.57
Numerator DOF           = 4
Denominator DOF         = 5
p = P(F >= S)           = 0.0438 Reject H0 at 5% sig.level
```

Table 13.21: MANOVA example 4. Profile analysis

do this, we first define the $m - 1$ by m transformation matrix K by

$$K = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & \dots \\ 0 & 1 & -1 & 0 & 0 & \dots \\ 0 & 0 & 1 & -1 & \dots & \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}.$$

Then a Hotelling's T^2 test is conducted using the pooled estimate for the covariance matrix $S_{ij} = [(n_i - 1)S_i + (n_j - 1)S_j]/(n_i + n_j - 2)$ and mean difference vector $d_{ij} = \bar{y}_i - \bar{y}_j$ according to

$$T^2 = \left(\frac{n_i n_j}{n_i + n_j}\right)(Kd_{ij})^T(KS_{ij}K^T)^{-1}(Kd_{ij})$$

and comparing the transformed statistic

$$\frac{n_i + n_j - m}{(n_i + n_j - 2)(m - 1)}T^2 \sim F_{m-1, n_1+n_2-m}$$

to the corresponding F distribution. Clearly, from table 13.21, the profiles are not parallel for the data in test file `manova1.tf1`.

13.18 Comparing groups: canonical variates (discriminant functions)

If MANOVA investigation suggests that at least one group mean vector differs from the the rest, it is usual to proceed to canonical variates analysis, although this technique can be also be used for data exploration when the assumption of multivariate normality with equal covariance matrices is not justified. Transforming multivariate data using canonical variates is a technique for highlighting differences between groups. Table 13.22 shows the results from analyzing data in the test file `manova1.tf4` which has three groups, each of size three. The most useful application of this technique is to plot the group means together with the data and 95% confidence regions in canonical variate space in order to visualize how close or how far apart the groups are. This is done for the first two canonical variates in figure 13.23, which requires some explanation. First of all, note that canonical variates, unlike principal components, are not simply obtained by a distance preserving rotation: the transformation is non-orthogonal and best represents the Mahalanobis distance between groups. In figure 13.23 we see the group means identified by the filled symbols labelled as 1, 2 and 3, each surrounded by a 95% confidence region, which in this case is circular as equally scaled physical distances are plotted along the axes. The canonical variates are uncorrelated and have unit variance so, assuming normality, the $100(1 - \alpha)$% confidence region for the population mean is a circle radius

$$r = \sqrt{\chi^2_{\alpha,2}/n_i},$$

```
Rank = 3
Correlations Eigenvalues Proportions     Chi-sq.  NDOF    p
      0.8826      3.5238      0.9795      7.9032     6  0.2453
      0.2623      0.0739      0.0205      0.3564     2  0.8368
Canonical variate means
     0.9841     0.2797
      1.181    -0.2632
     -2.165   -0.01642
Canonical coefficients
     -1.707     0.7277
     -1.348     0.3138
     0.9327       1.22
```

Table 13.22: Comparing groups: canonical variates

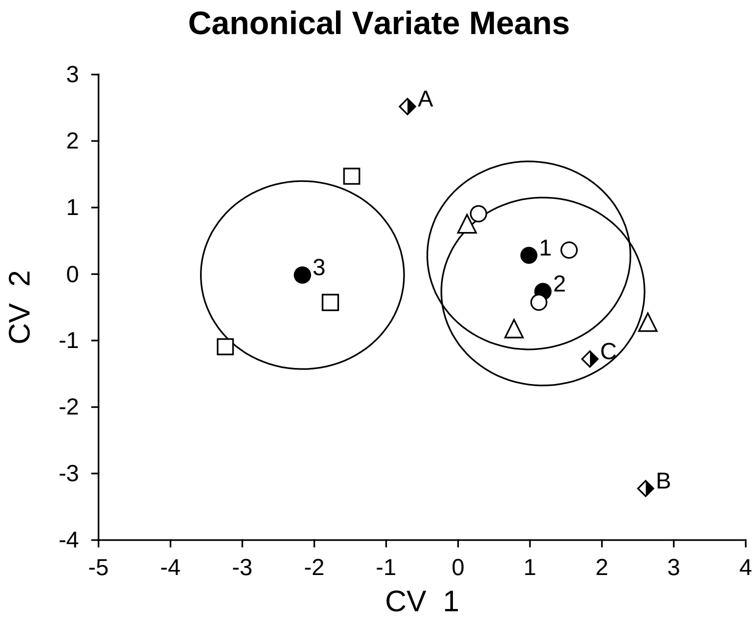


Figure 13.23: Comparing groups: canonical variates and confidence regions

where group i has n_i observations and $\chi^2_{\alpha,2}$ is the value exceeded by $100\alpha\%$ of a chi-square distribution with 2 degrees of freedom. Note that a circle radius $\sqrt{\chi^2_{\alpha,2}}$ defines a tolerance region, i.e. the region within which $100(1-\alpha)\%$ of the whole population is expected to lie. Also, the test file `manova1.tf4` has three other observations appended which are to be compared with the main groups in order to assign group membership, that is, to see to which of the main groups 1, 2 and 3 the extra observations should be assigned. The half-filled diamonds representing these are identified by the labels A, B and C which, like the identifying numbers 1, 2, and 3, are plotted automatically by SɪᴍFɪT to identify group means and extra data. In this case, as the data sets are small, the transformed observations from groups 1, 2 and 3 are also shown as circles, triangles and squares respectively, which is easily done by saving the coordinates from the plotted transforms of the observations in ASCII text files which are then added interactively as extra data files to the means plot.

The aim of canonical variate analysis is to find the transformations a_i that maximize F_i, the ratios of B (the between group sum of squares and products matrices) to W (the within-group sum of squares and products matrix), i.e.

$$F_i = \frac{a_i^T B a_i/(g-1)}{a_i^T W a_i/(n-g)}$$

where there are g groups and n observations with m covariates each, so that $i = 1, 2, \ldots, l$ where l is the lesser of the number of groups minus one and the rank of the data matrix. The canonical variates are obtained by solving the symmetric eigenvalue problem

$$(B - \lambda^2 W)x = 0,$$

where the eigenvalues λ_i^2 define the ratios F_i, and the eigenvectors a_i corresponding to the λ_i^2 define the transformations. So, just as with principal components, a scree diagram of the eigenvalues in decreasing order indicates the proportion of the ratio of between-group to within-group variance captured by the canonical variates. Note that table 13.22 lists the rank k of the data matrix, the number of canonical variates $l = \min(k, g-1)$, the eigenvalues λ_i^2, the canonical correlations $\lambda_i^2/(1+\lambda_i^2)$, the proportions $\lambda_i^2/\sum_{j=1}^{l}\lambda_j^2$, the group means, the loadings, and the results of a chi-square test. If the data are assumed to be from a common multivariate distribution, then to test for a significant dimensionality greater than some level i, the statistic

$$\chi^2 = (n - 1 - g - (k-g)/2)\sum_{j=i+1}^{l}\log(1+\lambda_j^2)$$

has an asymptotic chi-square distribution with $(k-i)(g-1-i)$ degrees of freedom. If the test is not significant for some level h, then the remaining tests for $i > h$ should be ignored. It should be noted that the group means and loadings are calculated for data after column centering and the canonical variates have within group variance equal to unity. Also, if the covariance matrices $\beta = B/(g-1)$ and $\omega = W/(n-g)$ are used, then $\omega^{-1}\beta = (n-g)W^{-1}B/(g-1)$, so eigenvectors of $W^{-1}B$ are the same as those of $\omega^{-1}\beta$, but eigenvalues of $W^{-1}B$ are $(g-1)/(n-g)$ times the corresponding eigenvalues of $\omega^{-1}\beta$.

Figure 13.24 illustrates the famous Fisher Iris data set contained in `manova1.tf5` and shown in table 13.12,

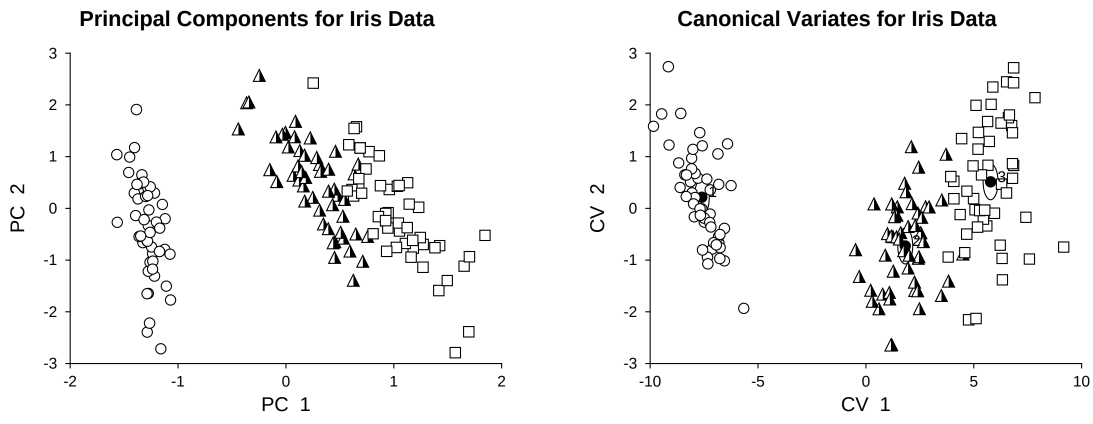


Figure 13.24: Comparing groups: principal components and canonical variates

using the first two principal components and also the first two canonical variates. In this instance there are only two canonical variates, so the canonical variates diagram is fully representative of the data set, and both techniques illustrate the distinct separation of group 1 (circles = setosa) from groups 2 (triangles = versicolor) and 3 (squares = virginica), and the lesser separation between groups 2 and 3. Users of these techniques should always remember that, as eigenvectors are only defined up to an arbitrary scalar multiple and different matrices may be used in the principal component calculation, principal components and canonical variates may have to be reversed in sign and re-scaled to be consistent with calculations reported using software other than SIMFIT. To see how to compare extra data to groups involved in the calculations, the test file `manova1.tf4` should be examined.

13.18.1 Comparing groups: Mahalanobis distances (discriminant analysis)

Discriminant analysis can be performed for grouped multivariate data as in table 13.23 for test file g03daf.tf1,

```
D^2 for all groups assuming unequal CV
        0.0         9.557        51.974
     8.5140           0.0        25.297
     25.121        4.7114           0.0

D^2 for samples/groups assuming unequal CV
     3.3393       0.75213      50.928
     20.777        5.6559    0.059653
     21.363        4.8411      19.498
    0.71841        6.2803      124.73
       55.0         88.86      71.785
      36.17        15.785      15.749
```

Table 13.23: Comparing groups: Mahalanobis distances

by calculating Mahalanobis distances between group means, or between group means and samples. The squared Mahalanobis distance D^2_{ij} between two group means $\bar{x}_i$ and $\bar{x}_j$ can be defined as either

$$D^2_{ij} = (\bar{x}_i - \bar{x}_j)^T S^{-1}(\bar{x}_i - \bar{x}_j)$$
$$\text{or } D^2_{ij} = (\bar{x}_i - \bar{x}_j)^T S_j^{-1}(\bar{x}_i - \bar{x}_j)$$

depending on whether the covariance matrices are assumed to be equal, when the pooled estimate S is used, or unequal when the group estimate S_j is used. This distance is a useful quantitative measure of similarity between groups, but often there will be extra measurements which can then be appended to the data file, as with manova1.tf2, so that the distance between measurement k and group j can be calculated as either

$$D^2_{kj} = (x_k - \bar{x}_j)^T S^{-1}(x_k - \bar{x}_j)$$
$$\text{or } D^2_{kj} = (x_k - \bar{x}_j)^T S_j^{-1}(x_k - \bar{x}_j).$$

From table 13.20 on page 232 we see that, for these data, the covariances must be regarded as unequal, so from table 13.23 we conclude that the groups are similarly spaced but, whereas extra data points 1 to 4 seem to belong to group 2, extra data points 5 and 6 can not be allocated so easily.

13.18.2 Comparing groups: Assigning new observations

Assigning new observations to groups defined by training sets can be made more objective by employing Bayesian techniques than by simply using distance measures, but only if a multivariate normal distribution can be assumed. For instance, table 13.24 displays the results from assigning the six observations appended to g03dcf.tf1 to groups defined by using the data as a training set, under the assumption of unequal variance-covariance matrices and equal priors. The calculation is for g groups, each with n_j observations on m variables, and it is necessary to make assumptions about the identity or otherwise of the variance-covariance matrices, as well as assigning prior probabilities. Then Bayesian arguments lead to expressions for posterior probabilities q_j, under a variety of assumptions, given prior probabilities π_j as follows.

- Estimative with equal variance-covariance matrices (Linear discrimination)

$$\log q_j \propto -\tfrac{1}{2}D^2_{kj} + \log \pi_j$$

- Estimative with unequal variance-covariance matrices (Quadratic discrimination)

$$\log q_j \propto -\tfrac{1}{2}D^2_{kj} + \log \pi_j - \tfrac{1}{2}\log|S_j|$$

```
Size of training set = 21
Number of groups = 3
Method: Predictive
CV-mat: Unequal
Priors: Equal
Observation Group-allocated
     1             2
     2             3
     3             2
     4             1
     5             3
     6             3
 Posterior probabilities
  0.0939  0.9046  0.0015
  0.0047  0.1682  0.8270
  0.0186  0.9196  0.0618
  0.6969  0.3026  0.0005
  0.3174  0.0130  0.6696
  0.0323  0.3664  0.6013
 Atypicality indices
  0.5956  0.2539  0.9747
  0.9519  0.8360  0.0184
  0.9540  0.7966  0.9122
  0.2073  0.8599  0.9929
  0.9908  0.9999  0.9843
  0.9807  0.9779  0.8871
```

Table 13.24: Comparing groups: Assigning new observations

- Predictive with equal variance-covariance matrices

$$q_j \propto \frac{\pi_j}{((n_j+1)/n_j)^{m/2}\{1+[n_j/((n-g)(n_j+1))]D_{kj}^2\}^{(n-g+1)/2}}$$

- Predictive with unequal variance-covariance matrices

$$q_j \propto \frac{\pi_j \Gamma(n_j/2)}{\Gamma((n_j-m)/2)((n_j^2-1)/n_j)^{m/2}|S_j|^{1/2}\{1+(n_j/(n_j^2-1))D_{kj}^2\}^{n_j/2}}$$

Subsequently the posterior probabilities are normalized so that $\sum_{j=1}^{g} q_j = 1$ and the new observations are assigned to the groups with the greatest posterior probabilities. In this analysis the priors can be assumed to be all equal, proportional to sample size, or user defined. Also, atypicality indices I_j are computed to estimate how well an observation fits into an assigned group. These are

- Estimative with equal or unequal variance-covariance matrices

$$I_j = P(D_{kj}^2/2, m/2)$$

- Predictive with equal variance-covariance matrices

$$I_j = R(D_{kj}^2/(D_{kj}^2 + (n-g)(n_j-1)/n_j), m/2, (n-g-m+1)/2)$$

- Predictive with unequal variance-covariance matrices

$$I_j = R(D_{kj}^2/(D_{kj}^2 + (n_j^2-1)/n_j), m/2, (n_j-m)/2),$$

where $P(x, \alpha)$ is the incomplete gamma function (page 426), and $R(x, \alpha, \beta)$ is the incomplete beta function (page 424). Values of atypicality indices close to one for all groups suggest that the corresponding new observation does not fit well into any of the training sets, since one minus the atypicality index can be interpreted as the probability of encountering an observation as or more extreme than the one in question given the training set. As before, observations 5 and 6 do not seem to fit into any of the groups.

Note that extra observations can be edited interactively or supplied independently in addition to the technique of appending to the data file as with `manova.tf2`. However, the assignment of extra observations to the training sets depends on the data transformation selected and variables suppressed or included in the analysis, and this must be considered when supplying extra observations interactively. Finally, once extra observations have been assigned, you can generate an enhanced training set, by creating a SIMFIT MANOVA type file in which the new observations have been appended to the groups to which they have been assigned.

13.18.3 Plotting training sets and assigned observations

Figure 13.25 displays the training set from `manoval.tf2`, together with the assignment of the extra observa-

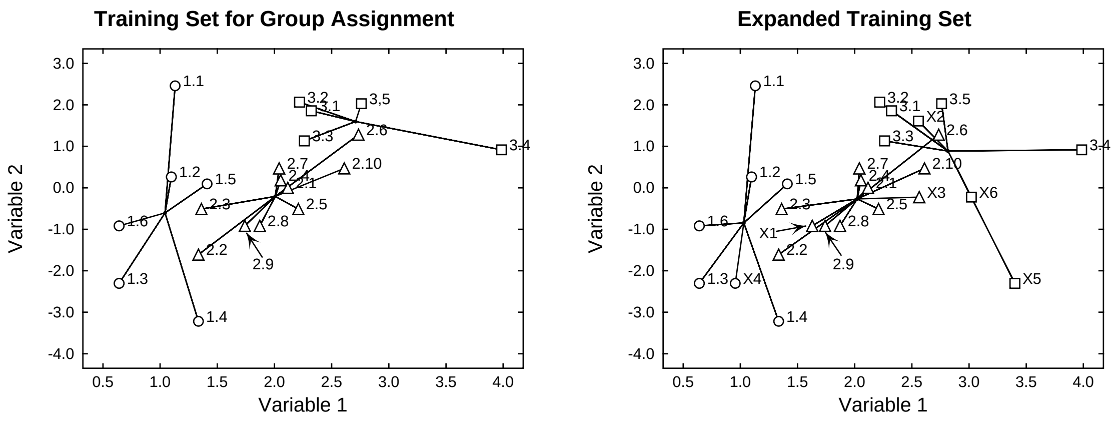


Figure 13.25: Training sets and groups assigned

tions appended to `manoval.tf2` just described. Note that observation 9 in group 1, labeled 2.9, and the extra observation number 1, labeled X1, have been displaced for clarity.

13.19 Factor analysis

This technique is used when it is wished to express a multivariate data set in m manifest, or observed variables, in terms of k latent variables, where $k < m$. Latent variables are variables that by definition are unobservable, such as social class or intelligence, and thus cannot be measured but must be inferred by estimating the relationship between the observed variables and the supposed latent variables. The statistical treatment is based upon a very restrictive mathematical model that, at best, will only be a very crude approximation and, most of the time, will be quite inappropriate. For instance, Krzanowski (in Principles of Multivariate Analysis, Oxford, revised edition, 2000) explains how the technique is used in the psychological and social sciences, but then goes on to state

At the extremes of, say, Physics or Chemistry, the models become totally unbelievable. p477
It should only be used if a positive answer is provided to the question, "Is the model valid?" p503

However, despite such warnings, the technique is now widely used, either to attempt to explain observables in terms of hypothetical unobservables, or as just another technique for expressing multivariate data sets in a space of reduced dimension. In this respect it is similar to principal components analysis (page223), except that the technique attempts to capture the covariances between the variables, not the variances. If the observed variables x can be represented as a linear combination of the unobservable variables or factors f, so that the partial correlation $r_{ij.l}$ between x_i and x_j with f_l fixed is effectively zero, then the correlation between x_i and x_j can be said to be explained by f_l. The idea is to estimate the coefficients expressing the dependence of x on f in such a way that the the residual correlation between the x variables is a small as possible, given the value of k.

The assumed relationship between the mean-centered observable variables x_i and the factors is

$$x_i = \sum_{j=1}^{k} \lambda_{ij} f_j + e_i \text{ for } i = 1, 2, \dots, m, \text{ and } j = 1, 2, \dots, k$$

where λ_{ij} are the loadings, f_i are independent normal random variables with unit variance, and e_i are independent normal random variables with variances ψ_i. If the variance covariance matrix for x is Σ, defined as

$$\Sigma = \Lambda\Lambda^T + \Psi,$$

where Λ is the matrix of factor loadings λ_{ij}, and Ψ is the diagonal matrix of variances ψ_i, while the sample covariance matrix is S, then maximum likelihood estimation requires the minimization of

$$F(\Psi) = \sum_{j=k+1}^{m} (\theta_j - \log \theta_j) - (m - k),$$

where θ_j are eigenvalues of $S^* = \Psi^{-1/2} S \Psi^{-1/2}$. Finally, the estimated loading matrix $\hat{\Lambda}$ is given by

$$\hat{\Lambda} = \Psi^{1/2} V (\Theta - I)^{1/2},$$

where V are the eigenvectors of S^*, Θ is the diagonal matrix of θ_i, and I is the identity matrix.

Table 13.25 illustrates the analysis of data in `g03caf.tf`, which contains a correlation matrix for $n = 211$ and $m = 9$. The proportion of variation for each variable x_i accounted for by the k factors is the communality $\sum_{j=1}^{k} \lambda_{ij}^2$, the Psi-estimates are the variance estimates, and the residual correlations are the off-diagonal elements of

$$C - (\Lambda\Lambda^T + \Psi)$$

where C is the sample correlation matrix. If a good fit has resulted and sufficient factors have been included, then the off-diagonal elements of the residual correlation matrix should be small with respect to the diagonals (listed with arbitrary values of unity to avoid confusion). Subject to the normality assumptions of the model, the minimum dimension k can be estimated by fitting sequentially with $k = 1$, $k = 2$, $k = 3$, and so on, until the likelihood ratio test statistic

$$TS = [n - 1 - (2m + 5)/6 - 2k/3] F(\hat{\Psi})$$

is not significant as a chi-square variable with $[(m - k)^2 - (m + k)]/2$ degrees of freedom. Note that data for factor analysis can be input as a general n by m multivariate matrix, or as either a m by m covariance or correlation matrix. However, if a square covariance or correlation matrix is input then there are two further considerations: the sample size must be supplied independently, and it will not be possible to estimate or plot the sample scores in factor space, as the original sample matrix will not be available.

It remains to explain the estimation of scores, which requires the original data of course, and not just the covariance or correlation matrix. This involves the calculation of a m by k factor score coefficients matrix Φ, so that the estimated vector of factor scores $\hat{f}$, given the x vector for an individual can be calculated from

$$\hat{f} = x^T \Phi.$$

```
Number of variables   = 9, Transformation = Untransformed
Matrix type           = Input correlation/covariance matrix
Number of factors     = 3, Replicates = Unweighted
F(Psi-hat)            = 0.035017
Test stat TS          = 7.1494
Degrees of Freedom    = 12 (Number of cases = 211)
p = P(chisq >= TS)    = 0.8476
   Eigenvalues Communalities Psi-estimates
        15.968       0.54954       0.45046
        4.3577       0.57293       0.42707
        1.8475       0.38345       0.61655
         1.156       0.78767       0.21233
         1.119       0.61947       0.38053
        1.0271       0.82308       0.17692
       0.92574       0.60046       0.39954
       0.89508       0.53846       0.46154
        0.8771       0.76908       0.23092
Residual correlations
  0.0004
 -0.0128  0.0220
  0.0114 -0.0053  0.0231
 -0.0100 -0.0194 -0.0162  0.0033
 -0.0046  0.0113 -0.0122 -0.0009 -0.0008
  0.0153 -0.0216 -0.0108  0.0023  0.0294 -0.0123
 -0.0011 -0.0105  0.0134  0.0054 -0.0057 -0.0009  0.0032
 -0.0059  0.0097 -0.0049 -0.0114  0.0020  0.0074  0.0033 -0.0012
Factor loadings by columns
  0.6.6421 -0.3.2087    0.073519
  0.6.8883 -0.2.4714    -0.19328
  0.4.9262 -0.3.0216    -0.22243
  0.8.3720  0.2.9243   -0.035395
  0.7.0500  0.3.1479    -0.15278
  0.8.1870  0.3.7667     0.10452
  0.6.6150 -0.3.9603   -0.077747
  0.4.5793 -0.2.9553     0.49135
  0.7.6567 -0.4.2743   -0.011701
```

Table 13.25: Factor analysis 1: calculating loadings

However, when calculating factor scores from the factor score coefficient matrix in this way, the observable variables x_i must be mean centered, and also scaled by the standard deviations if a correlation matrix has been analyzed. The regression method uses

$$\Phi = \Psi^{-1}\Lambda(I + \Lambda^T\Psi^{-1}\Lambda)^{-1},$$

while the Bartlett method uses

$$\Phi = \Psi^{-1}\Lambda(\Lambda^T\Psi^{-1}\Lambda)^{-1}.$$

Table 13.26 shows the analysis of `g03ccf.tf1`, a correlation matrix for 220 cases, 6 variables and 2 factors, but a further possibility should be mentioned. As the factors are only unique up to rotation, it is possible to perform a Varimax or Quartimax rotation (page 227) to calculate a rotation matrix R before working out the score coefficients, which may simplify the interpretation of the observed variables in terms of the unobservable variables.

```
TS = 2.3346, p = P(chisq >= TS) = 0.6745,
DOF = 4 (n = 220, m = 6, k = 2)
   Eigenvalues Communalities Psi-estimates
        5.6142       0.48983       0.51017
        2.1428       0.40593       0.59407
        1.0923       0.35627       0.64373
        1.0264       0.62264       0.37736
       0.99082       0.56864       0.43136
       0.89051       0.37179       0.62821
Factor loadings by columns
     0.55332    -0.42856
      5.6816    -0.28832
      3.9218    -0.44996
      7.4042      0.2728
      7.2387     0.21131
      5.9536     0.13169
Factor score coefficients, Method: Regression, Rotation: None
     0.19318    -0.39203
     0.17035    -0.22649
     0.10852    -0.32621
     0.34950     0.33738
     0.29891     0.22861
     0.16881    0.097831
```

Table 13.26: Factor analysis 2: calculating factor scores

13.20 Biplots

The biplot is used to explore relationships between the rows and columns of any arbitrary matrix, by projecting the matrix onto a space of smaller dimensions using the singular value decomposition (SVD Page 309). It is based upon the fact that, as a n by m matrix X of rank k can be expressed as a sum of k rank 1 matrices as follows

$$X = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T,$$

then the best fit rank r matrix Y with $r < k$ which minimizes the objective function

$$\begin{aligned} S &= \sum_{i=1}^{m}\sum_{j=1}^{n}(x_{ij} - y_{ij})^2 \\ &= \text{trace}[(X - Y)(X - Y)^T] \end{aligned}$$

is the sum of the first r of these rank 1 matrices. Further, such a least squares approximation results in the minimum value

$$S_{min} = \sigma_{r+1}^2 + \sigma_{r+2}^2 + \cdots + \sigma_k^2$$

so that the rank r least squares approximation Y accounts for a fraction

$$\frac{\sigma_1^2 + \cdots + \sigma_r^2}{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_k^2}$$

of the total variance, where k is less than or equal to the smaller of n and m, k is greater than or equal to r, and $\sigma_i = 0$ for $i > k$.

Figure 13.26 illustrates a biplot for the data in test file `houses.tf1`. The technique is based upon creating one

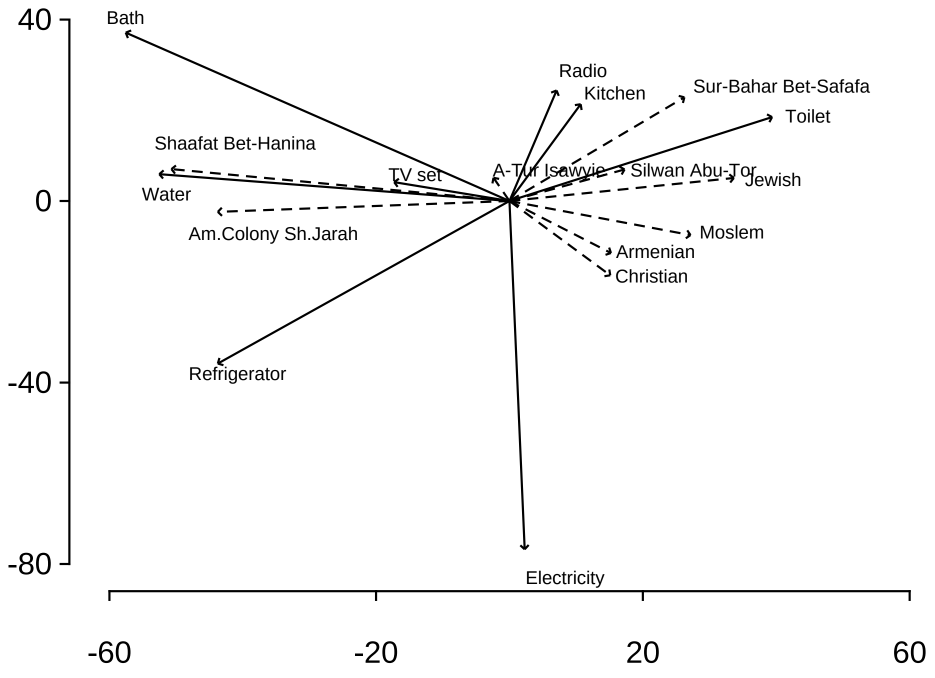


Figure 13.26: Two dimensional biplot for East Jerusalem Households

of several possible rank-2 representations of of a n by m matrix X with rank k of at least two as follows. Let the SVD of X be

$$\begin{aligned} X &= U\Sigma V^T \\ &= \sum_{i=1}^{k} \sigma_i u_i v_i^T \end{aligned}$$

so that the best fit rank-2 matrix Y to the original matrix X will be

$$Y = \begin{pmatrix} u_{11} & u_{21} \\ u_{12} & u_{22} \\ \vdots & \vdots \\ u_{1n} & u_{2n} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1m} \\ v_{21} & v_{22} & \dots & v_{2m} \end{pmatrix}.$$

Then Y can be written in several ways as GH^T, where G is a n by 2 matrix and H is a m by 2 matrix as follows.

1. General representation

$$Y = \begin{pmatrix} u_{11}\sqrt{\sigma_1} & u_{21}\sqrt{\sigma_2} \\ u_{12}\sqrt{\sigma_1} & u_{22}\sqrt{\sigma_2} \\ \vdots & \vdots \\ u_{1n}\sqrt{\sigma_1} & u_{2n}\sqrt{\sigma_2} \end{pmatrix} \begin{pmatrix} v_{11}\sqrt{\sigma_1} & v_{12}\sqrt{\sigma_1} & \dots & v_{1m}\sqrt{\sigma_1} \\ v_{21}\sqrt{\sigma_2} & v_{22}\sqrt{\sigma_2} & \dots & v_{2m}\sqrt{\sigma_2} \end{pmatrix}$$

2. Representation with row emphasis

$$Y = \begin{pmatrix} u_{11}\sigma_1 & u_{21}\sigma_2 \\ u_{12}\sigma_1 & u_{22}\sigma_2 \\ \vdots & \vdots \\ u_{1n}\sigma_1 & u_{2n}\sigma_2 \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1m} \\ v_{21} & v_{22} & \dots & v_{2m} \end{pmatrix}$$

3. Representation with column emphasis

$$Y = \begin{pmatrix} u_{11} & u_{21} \\ u_{12} & u_{22} \\ \vdots & \vdots \\ u_{1n} & u_{2n} \end{pmatrix} \begin{pmatrix} v_{11}\sigma_1 & v_{12}\sigma_1 & \dots & v_{1m}\sigma_1 \\ v_{21}\sigma_2 & v_{22}\sigma_2 & \dots & v_{2m}\sigma_2 \end{pmatrix}$$

4. User-defined representation

$$Y = \begin{pmatrix} u_{11}\sigma_1^\alpha & u_{21}\sigma_2^\alpha \\ u_{12}\sigma_1^\alpha & u_{22}\sigma_2^\alpha \\ \vdots & \vdots \\ u_{1n}\sigma_1^\alpha & u_{2n}\sigma_2^\alpha \end{pmatrix} \begin{pmatrix} v_{11}\sigma_1^\beta & v_{12}\sigma_1^\beta & \dots & v_{1m}\sigma_1^\beta \\ v_{21}\sigma_2^\beta & v_{22}\sigma_2^\beta & \dots & v_{2m}\sigma_2^\beta \end{pmatrix}$$

   where $0 < \alpha < 1$, and $\beta = 1 - \alpha$.

To construct a biplot we take the n row effect vectors g_i and m column effect vectors h_j as vectors with origin at $(0, 0)$ and defined in the general representation as

$$\begin{aligned} g_i^T &= (u_{1i}\sqrt{\sigma_1}, u_{2i}\sqrt{\sigma_2}) \\ h_j^T &= (v_{1j}\sqrt{\sigma_1}, v_{2j}\sqrt{\sigma_2}) \end{aligned}$$

with obvious identities for the alternative row emphasis and column emphasis factorizations. The biplot consists of n vectors with end points at $(u_{1i}\sqrt{\sigma_1}, u_{2i}\sqrt{\sigma_2})$ and m vectors with end points at $(v_{1j}\sqrt{\sigma_1}, v_{2j}\sqrt{\sigma_2})$ so that interpretation of the biplot is then in terms of the inner products of vector pairs. That is, vectors with the same direction correspond to proportional rows or columns, while vectors approaching right angles indicate near orthogonality, or small contributions. Another possibility is to display a difference biplot in which a residual matrix R is first created by subtracting the best fit rank-1 matrix so that

$$\begin{aligned} R &= X - \sigma_1 u_1 v_1^T \\ &= \sum_{i=2}^{k} \sigma_i u_i v_i^T \end{aligned}$$

and this is analyzed, using appropriate vectors calculated with σ_2 and σ_3 of course. Again, the row vectors may dominate the column vectors or vice versa whatever representation is used and, to improve readability, additional scaling factors may need to be introduced. For instance, figure 13.26 used the residual matrix and scaling factors of -100 for rows and -1 for columns to reflect and stretch the vectors until comparable size was attained. To do this over-rides the default autoscaling option, which is to scale each set of vectors so that the largest row and largest column vector are of unit length, whatever representation is chosen.

Biplots are most useful when the number of rows and columns is not too large, and when the rank-2 approximation is satisfactory as an approximation to the data or residual matrix. Note that biplot labels should be short, and they can be appended to the data file as with `houses.tf1`, or pasted into the plot as a table of label values. Fine tuning to re-position labels was necessary with figure 13.26, and this can be done by editing the PostScript file in a text editor (page 366), or by using the same techniques described for scattergrams with labels (page 222).

Sometimes, as with figure 13.27, it is useful to inspect biplots in three dimensions. This has the advantage that three singular values can be used, but the plot may have to be viewed from several angles to get a good idea of which vectors of like type are approaching a parallel orientation (indicating proportionality of rows or columns) and which pairs of vectors i, j of opposite types are orthogonal (i.e., at right angles, indicating small contributions to x_{ij})
As with other projection techniques, such as principal components, it is necessary to justify that the number of singular values used to display a biplot does represent the data matrix adequately. To do this, consider table 13.27 from the singular value decomposition of `houses.tf1`.

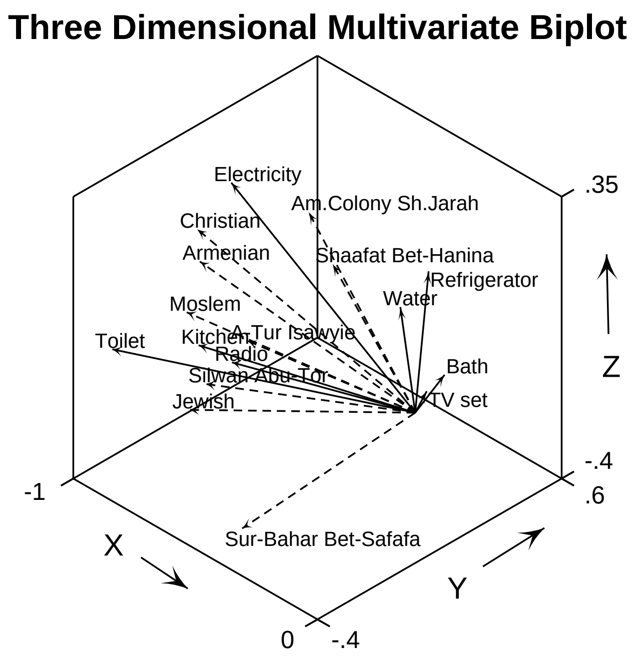


Figure 13.27: Three dimensional biplot for East Jerusalem Households

```
rank = 8
Index     Sigma(i) Fraction Cumulative   Sigma(i)^2 Fraction Cumulative
    1      499.39   0.7486     0.7486     249394.0   0.9631     0.9631
    2      88.348   0.1324     0.8811      7805.36   0.0301     0.9933
    3     33.6666   0.0505     0.9315      1133.44   0.0044     0.9977
    4     17.8107   0.0267     0.9582      317.222   0.0012     0.9989
    5     12.8584   0.0193     0.9775      165.339   0.0006     0.9995
    6     10.4756   0.0157     0.9932      109.738   0.0004     1.0000
    7     3.37372   0.0051     0.9983       11.382   0.0000     1.0000
    8     1.15315   0.0017     1.0000      1.32974   0.0000     1.0000
```

Table 13.27: Singular values for East Jerusalem Households

In this example, it is clear that the first two or three singular values do represent the data adequately, and this is further reinforced by figure 13.28 where the percentage variance represented by the successive singular values is plotted as a function of the singular value index. Here we see plotted the cumulative variance $CV(i)$

$$CV(i) = \frac{100 \sum_{j=1}^{i} \sigma_j^2}{\sum_{j=1}^{k} \sigma_j^2}$$

plotted as a function of the index i, and such tables or plots should always be inspected to make sure that $CV(i)$ is greater than some minimum value (say 70 percent, for instance) for $i = 2$ or $i = 3$ as appropriate.

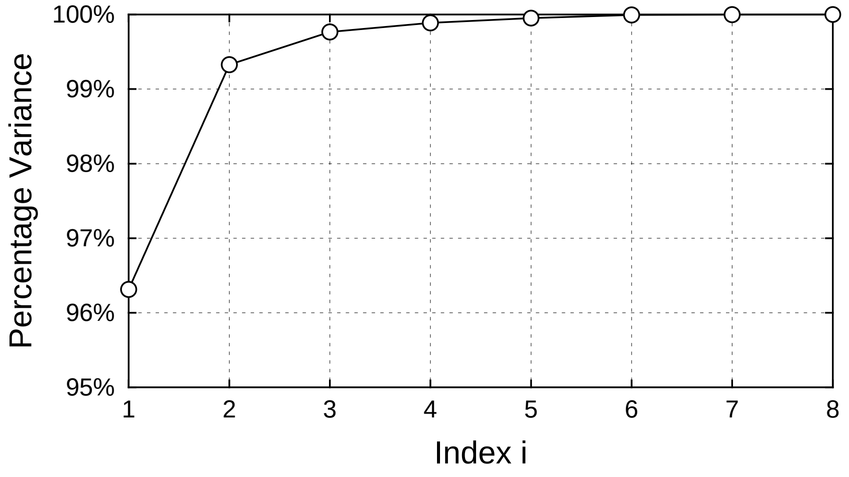


Figure 13.28: Percentage variance from singular value decomposition

Part 14

Time series

14.1 Introduction

A time series is a vector $x(t)$ of $n > 1$ observations x_i obtained at a sequence of points t_i, e.g., times, distances, etc., at fixed intervals Δ, i.e.

$$\Delta = t_{i+1} - t_i, \text{ for } i = 1, 2, \ldots, n-1,$$

and it is assumed that there is some seasonal variation, or other type of autocorrelation to be estimated. A linear trend can be removed by first order differencing

$$\nabla x_t = x_t - x_{t-1},$$

while seasonal patterns of seasonality s can be eliminated by first order seasonal differencing

$$\nabla_s x_t = x_t - x_{t-s}.$$

14.2 Time series data smoothing

Sometimes it is useful to be able to smooth a time series in order to suppress outliers and reveal trends more clearly. In extreme cases it may even be better to create a smoothed data set for further correlation analysis or model fitting. The obvious way to do this is to apply a moving average of span n, which replaces the data values by the average of n adjacent values to create a smooth set. When n is odd, it is customary to set the new smooth point equal to the mean of the original value and the $(n-1)/2$ values on either side but, when n is even, the Hanning filter is used that is, double averaging or alternatively using an appropriately weighted mean of span $(n+1)$. Because such moving averages could be unduly influenced by outliers, running medians can also be used, however a very popular smoothing method is the 4253H twice smoother. This starts by applying a span 4 running median centered by 2, followed by span 5 then span 3 running medians, and finally a Hanning filter. The rough (i.e., residuals) are then treated in the same way and the first-pass smooth are re-roughed by adding back the smoothed rough, then finally the rough are re-calculated. Figure 14.1 illustrates the effect of this $T4253H$ smoothing.

14.3 Time series lags and autocorrelations

This procedure should be used to explore a time series before fitting an ARIMA model (page 249). The general idea is to observe the autocorrelations and partial autocorrelations in order to identify a suitable differencing scheme. You input a vector of length NX, which is assumed to represent a time series sequence with fixed differences, e.g., every day, at intervals of 10 centimeters, etc. Then you choose the orders of non-seasonal differencing ND, and seasonal differencing NDS, along with seasonality NS, the maximum number of lags required NK, and the maximum number of partial autocorrelations of interest L. All autocorrelations and partial autocorrelations requested are then worked out, and a statistic S to test for the presence of significant

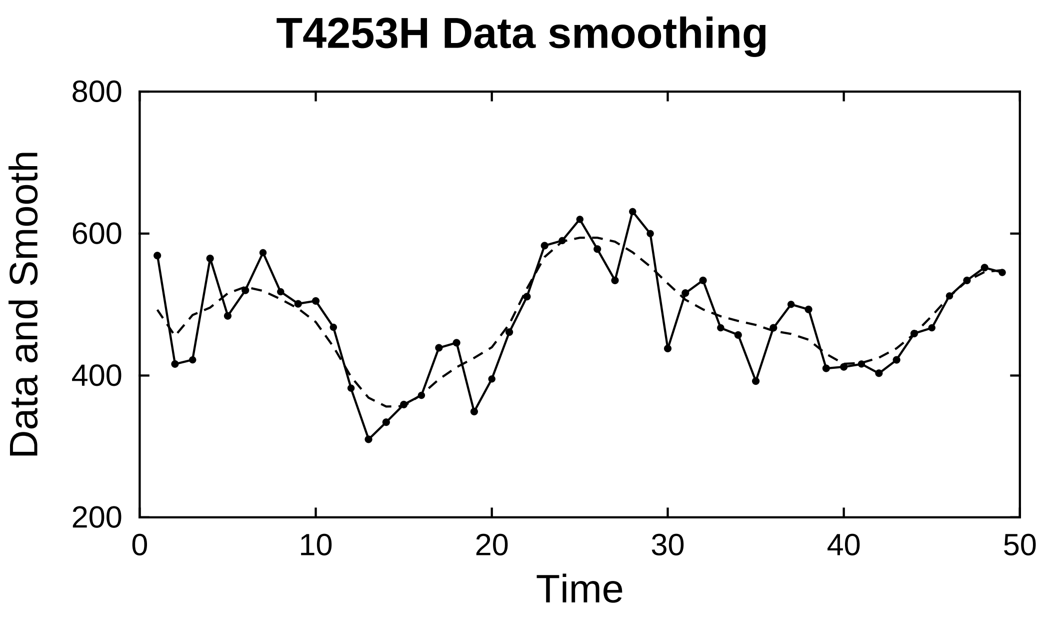


Figure 14.1: The T4253H data smoother

```
Original dimension (NX)    = 100
After differencing (NXD) = 99
Non-seasonal order (ND)    = 1
Seasonal order (NDS)       = 0
Seasonality (NS)           = 0
No. of lags (NK)           = 10
No. of PACF (NVL)          = 10
X-mean (differenced)       = 0.4283
X-variance (differenced) = 0.3152
Statistic (S)              = 83.13
P(chi-sq >= S)             = 0.0000
 Lag       R         PACF        VR          ARP
   1     0.5917     0.5917     0.6498      0.3916
   2     0.5258     0.2703     0.6024      0.3988
   3     0.3087    -0.1299     0.5922    0.001601
   4     0.1536    -0.1440     0.5799     -0.1440
   5     0.0345   -0.05431     0.5782     -0.1365
   6    -0.0297    0.01105     0.5782    -0.04528
   7    -0.0284    0.07109     0.5752      0.1474
   8    -0.0642   -0.04492     0.5741      0.1306
   9    -0.1366    -0.1759     0.5563    -0.06707
  10    -0.2619    -0.2498     0.5216     -0.2498
```

Table 14.1: Autocorrelations and Partial Autocorrelations

autocorrelations in the data is calculated. Table 14.1 shows the results from analysis of `times.tf1`. Note that differencing of orders $d = ND$, $D = NDS$, and seasonality $s = NS$ may be applied repeatedly to a series so that

$$w_t = \nabla^d \nabla_s^D x_t$$

will be shorter, of length $NXD = n - d - D \times s$, and will extend for $t = 1 + d + D \times s, \dots, NX$.

Non-seasonal differencing up to order d is calculated sequentially using

$$\begin{array}{lcll}
\nabla^1 x_i & = & x_{i+1} - x_i & \text{for } i = 1, 2, \dots, n-1 \\
\nabla^2 x_i & = & \nabla^1 x_{i+1} - \nabla^1 x_i & \text{for } i = 1, 2, \dots, n-2 \\
\dots & & & \\
\nabla^d x_i & = & \nabla^{d-1} x_{i+1} - \nabla^{d-1} x_i & \text{for } i = 1, 2, \dots, n-d
\end{array}$$

while seasonal differencing up to order D is calculated by the sequence

$$\begin{array}{lcll}
\nabla^d \nabla_s^1 x_i & = & \nabla^d x_{i+s} - \nabla^d x_i & \text{for } i = 1, 2, \dots, n-d-s \\
\nabla^d \nabla_s^2 x_i & = & \nabla^d \nabla_s^1 x_{i+s} - \nabla^d \nabla_s^1 x_i & \text{for } i = 1, 2, \dots\ n-d-2s \\
\dots & & & \\
\nabla^d \nabla_s^D x_i & = & \nabla^d \nabla_s^{D+1} x_{i+s} - \nabla^d \nabla_s^{D+1} x_i & \text{for } i = 1, 2, \dots, n-d-D \times s.
\end{array}$$

Note that, as indicated in table 14.1, either the original sample X of length NX, or a differenced series XD of length NXD, can be analyzed interactively, by simply adjusting ND, NDS, or NS. Also the maximum number of autocorrelations $NK < NXD$ and maximum number of partial autocorrelations $L \le NK$, can be controlled, although the maximum number of valid partial autocorrelations NVL may turn out to be less than L. Now, defining either $x = X$, and $n = NX$, or else $x = XD$ and $n = NXD$ as appropriate, and using $K = NK$, the mean and variance are recorded, plus the autocorrelation function R, comprising the autocorrelation coefficients of lag k according to

$$r_k = \sum_{i=1}^{n-k} (x_i - \bar{x})(x_{i+k} - \bar{x}) \Big/ \sum_{i=1}^{n} (x_i - \bar{x})^2.$$

If n is large and much larger than K, then the S statistic

$$S = n \sum_{k=1}^{K} r_k^2$$

has a chi-square distribution with K degrees of freedom under the hypothesis of zero autocorrelation, and so it can be used to test that all correlations are zero. The partial autocorrelation function $PACF$ has coefficients at lag k corresponding to $p_{k,k}$ in the autoregression

$$x_t = c_k + p_{k,1} x_{t-1} + p_{k,2} x_{t-2} + \cdots + p_{k,l} x_{t-k} + e_{k,t}$$

where $e_{k,t}$ is the predictor error, and the $p_{k,k}$ estimate the correlation between x_t and x_{t+k} conditional upon the intermediate values $x_{t+1}, x_{t+2}, \dots, x_{t+k-1}$. Note that the parameters change as k increases, and so $k = 1$ is used for $p_{1,1}$, $k = 2$ is used for $p_{2,2}$, and so on. These parameters are determined from the Yule-Walker equations

$$r_i = p_{k,1} r_{i-1} + p_{k,2} r_{i-2} + \cdots + p_{k,k} r_{i-k},\ i = 1, 2, \dots, k$$

where $r_j = r_{|j|}$ when $j < 0$, and $r_0 = 1$. An iterative technique is used and it may not always be possible to solve for all the partial autocorrelations requested. This is because the predictor error variance ratios VR are defined as

$$\begin{aligned}
v_k &= Var(e_{k,t})/Var(x_t) \\
&= 1 - p_{k,1} r_1 - p_{k,2} r_2 - \cdots - p_{k,k} r_k,
\end{aligned}$$

unless $|p_{k,k}| \ge 1$ is encountered at some $k = L_0$, when the iteration terminates, with $NVL = L_0 - 1$. The Autoregressive parameters of maximum order ARP are the final parameters $p_{L,j}$ for $j = 1, 2, \dots, NVL$ where NVL is the number of valid partial autocorrelation values, and L is the maximum number of partial autocorrelation coefficients requested, or else $L = L_0 - 1$ as before in the event of premature termination of the algorithm.

Figure 14.2 shows the data in test file `times.tf1` before differencing and after first order non-seasonal differencing has been applied to remove the linear trend. Note that, to obtain hardcopy of any differenced

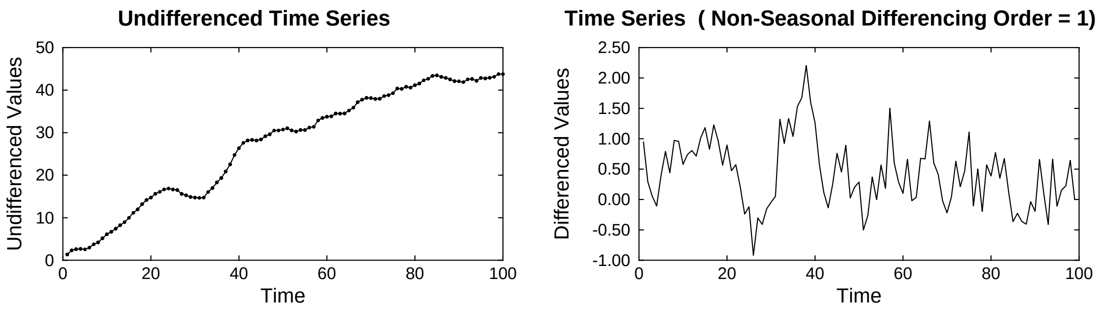


Figure 14.2: Time series before and after differencing

series, a file containing the t values and corresponding differenced values can be saved from the graph as an ASCII coordinate file, then column 1 can be discarded using **editfl**. A valuable way to detect significant autocorrelations is to plot the autocorrelation coefficients, or the partial autocorrelation coefficients, as in figure 14.3. The statistical significance of autocorrelations at specified lags can be judged by plotting the approximate 95% confidence limits or, as in this case, by plotting $2/\sqrt{n}$, where n is the sample size (after differencing, if any). Note that in plotting time series data you can always choose the starting value and the increment between observations, otherwise defaults starting at 1 with an increment of 1 will be assumed.

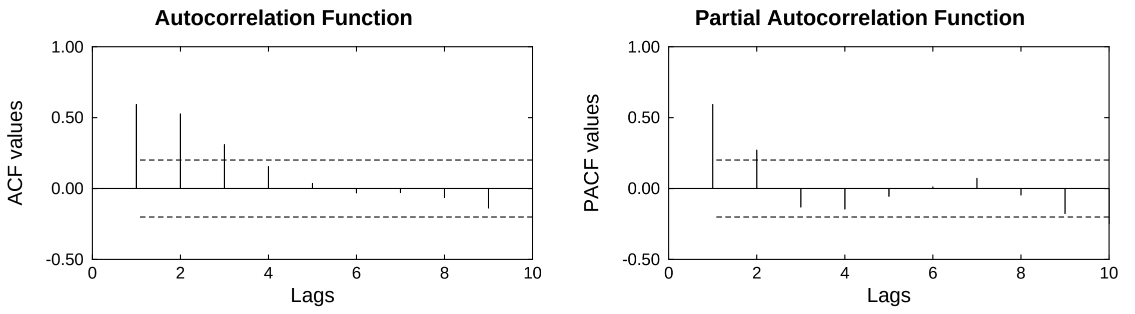


Figure 14.3: Times series autocorrelation and partial autocorrelations

14.4 Autoregressive integrated moving average models (ARIMA)

It must be stressed that fitting an ARIMA model is a very specialized iterative technique that does not yield unique solutions. So, before using this procedure, you must have a definite idea, by using the autocorrelation and partial autocorrelation options (page 246), or by knowing the special features of the data, exactly what differencing scheme to adopt and which parameters to fit. Users can select the way that starting estimates are estimated, they can monitor the optimization, and they can alter the tolerances controlling the convergence, but only expert users should alter the default settings.

It is assumed that the time series data $x_1, x_2, \ldots, x_n$ follow an ARIMA model so that a differenced series given by

$$w_t = \nabla^d \nabla_s^D x_i - c$$

can be fitted, where c is a constant, d is the order of non-seasonal differencing, D is the order of seasonal differencing and s is the seasonality. The method estimates the expected value c of the differenced series in terms of an uncorrelated series a_t and an intermediate series e_t using parameters $\phi, \theta, \Phi, \Theta$ as follows. The seasonal structure is described by

$$w_t = \Phi_1 w_{t-s} + \Phi_2 w_{t-2\times s} + \cdots + \Phi_P w_{t-P\times s} + e_t - \Theta_1 e_{t-s} - \Theta_2 e_{t-2\times s} - \cdots - \Theta_Q e_{t-Q\times s}$$

while the non-seasonal structure is assumed to be

$$e_t = \phi_1 e_{t-1} + \phi_2 e_{t-2} + \cdots + \phi_p e_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q}.$$

The model parameters $\phi_1, \phi_2, \ldots, \phi_p, \theta_1, \theta_2, \ldots, \theta_q$ and $\Phi_1, \Phi_2, \ldots, \Phi_P, \Theta_1\Theta_2, \ldots, \Theta_Q$ are estimated by nonlinear optimization, the success of which is heavily dependent on choosing an appropriate differencing scheme, starting estimates and convergence criteria. After fitting an ARIMA model, forecasts can be estimated along with 95% confidence limits.

For example, table 14.2 shows the results from fitting the data in `times.tf1` with a non-seasonal order of

```
Original dimension (NX)     = 100
After differencing (NXD)    = 99
Non-seasonal order (ND)     = 1
Seasonal order (NDS)        = 0
Seasonality (NS)            = 0
Number of forecasts (NF)    = 3
Number of parameters (NP)   = 1
Number of iterations (ITC)  = 2
Sum of squares (SSQ)     = 0.199E+02
  Parameter    Value     Std. err.     Type
    phi(1)    0.60081    0.08215   Autoregressive
      C(0)    0.42959     0.1124   Constant term
   pred(1)     43.935      0.453   Forecast
   pred(2)     44.208     0.8551   Forecast
   pred(3)     44.544      1.233   Forecast
```

Table 14.2: Fitting an ARIMA model to time series data

one and no seasonality, along with forecasts and associated standard errors, while figure 14.4 illustrates the fit. On the first graph the original time series data are plotted along with forecasts and 95% confidence limits for the predictions. However it should be realized that only the differenced time series has been fitted, that is, after first order differencing to remove the linear trend. So in the second plot the best fit ARIMA model is shown as a continuous line, while the differenced data are plotted as symbols.

14.5 Auto- and cross-correlation matrices

This technique is used when there are two time series, or in fact any series of signals recorded at a sequence of fixed discrete intervals of time or space etc., and a comparison of the two series is required.

Table 14.3 illustrates the results from analyzing test file `g13dmf.tf1` for the first ten nonzero lags using the SIMFIT time series options.
The data must be supplied as two vectors, say X and Y of length n for instance, with X as column 1 of a n by 2 matrix, and Y as column 2. The routine first calculates the sample means $\bar{x}$ and $\bar{y}$, the sample variances V_x and V_y, and sample correlation coefficient r. Then, for a selected number of lags $m = 1, 2, \ldots, k$, the auto-correlations and cross-correlations are output as a sequence of 2 by 2 matrices.

Since $1/\sqrt{n}$ is a rough approximation to the standard errors of these estimates, the approximate significance for the sample cross-correlations is indicated as in table 14.3 using the following labeling scheme.

$$|r(i, j)| > 3.29/\sqrt{n} : * * *$$
$$|r(i, j)| > 2.58/\sqrt{n} : **$$
$$|r(i, j)| > 1.96/\sqrt{n} : *.$$

Finally, the off-diagonal i.e., cross-correlation, coefficient with largest absolute value is indicated. If this value is close to unity it indicates that the series are closely similar, and the value of m at which this occurs indicates the extent to which the series have to be slid past each other to obtain maximum similarity of profiles. Usually, the largest value of m selected for analysis would be for $k \le n/4$.

Defining the denominator D as follows

$$D = \sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

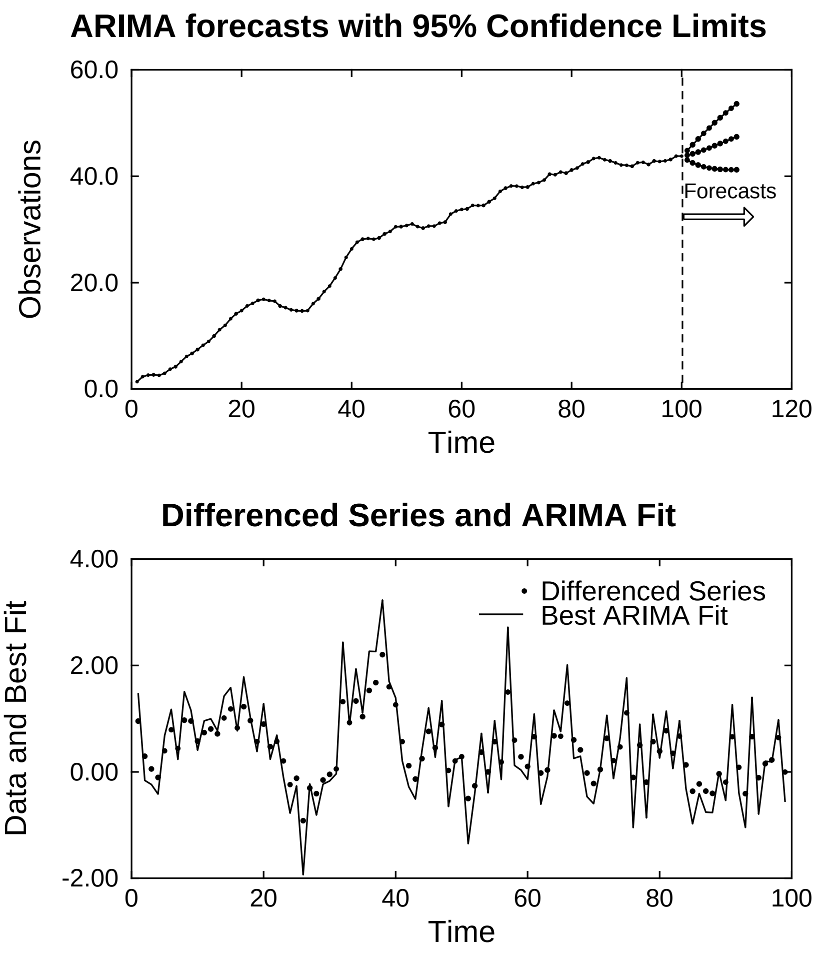


Figure 14.4: Fitting an ARIMA model to time series data

```
Auto- and cross-correlations: n = 48, approx.st.dev. = 1.443E-01
Mean of X = 4.37021E+00,  Mean of Y = 7.86750E+00
For lag m = 0:  sample X,Y Correlation coefficient r = 0.2493
m = 1:  0.7366(***)  0.1743
        0.2114       0.5541(***)

m = 2 :  0.4562(**)   0.0764
         0.0693       0.2602

m = 3 :  0.3795(**)   0.0138
         0.0260      -0.0381

m = 4 :  0.3227(*)    0.1100
         0.0933      -0.2357

m = 5 :  0.3414(*)    0.2694
         0.0872      -0.2499

m = 6 :  0.3634(*)    0.3436(*)
         0.1323      -0.2263

m = 7 :  0.2802       0.4254(**)
         0.2069      -0.1283

m = 8 :  0.2482       0.5217(***)
         0.1970      -0.0845

m = 9 :  0.2400       0.2664
         0.2537       0.0745

m = 10:  0.1621      -0.0197
        0.2667       0.0047
Indicators: p<.005(***), p<.01(**), p<.05(*)
Max. off-diag., m = 8, |C(1,2)| = 0.5217
```

Table 14.3: Auto- and cross-correlation matrices

then the auto-correlations $r(1,1)$ and $r(2,2)$, and the cross-correlations $r(1,2)$ and $r(2,1)$ as functions of m are given by

$$r(1,1) = \frac{1}{D}\sum_{i=1}^{n-m}(x_i - \bar{x})(x_{i+m} - \bar{x})$$
$$r(1,2) = \frac{1}{D}\sum_{i=1}^{n-m}(x_i - \bar{x})(y_{i+m} - \bar{y})$$
$$r(2,1) = \frac{1}{D}\sum_{i=1}^{n-m}(x_{i+m} - \bar{x})(y_i - \bar{y})$$
$$r(2,2) = \frac{1}{D}\sum_{i=1}^{n-m}(y_i - \bar{y})(y_{i+m} - \bar{y})$$

for $m = 1, 2, \ldots, k$.

Part 15

Survival analysis

15.1 Introduction

In the context of survival analysis, the random survival time $T \geq 0$, with density $f(t)$, cumulative distribution function $F(t)$, survivor function $S(t)$, hazard function $h(t)$, and integrated hazard function $H(t)$ are defined by

$$\begin{aligned} S(t) &= 1 - F(t) \\ h(t) &= f(t)/S(t) \\ H(t) &= \int_0^t h(u)\,du \\ f(t) &= h(t)\exp\{-H(t)\}. \end{aligned}$$

There will usually be one or more groups of subjects from which individuals are removed as time goes by. Typically, members of the group are removed by some critical event, such as death, but often individuals leave a group for other reasons, and these are referred to as censored observations. Survival analysis attempts to fit models to such data in order to estimate parameters and thereby predict survival probabilities.

15.2 Fitting one set of survival times

The idea is that you have one or more samples of survival times (page 419) with possible censoring, but no covariates, that you wish to analyze and compare for differences, using **gcfit** in mode 3, or **simstat**. In other words, you observe one or more groups and, at known times, you record the frequency of failure or censoring. You would want to calculate a nonparametric Kaplan-Meier estimate for the survivor function, as well as a maximum likelihood estimate for some supposed probability density function, such as the Weibull distribution. Finally, you would want to compare survivor functions in groups by comparing parameter estimates, or using the Mantel-Haenszel log rank test, or by resorting to some model such as the proportional hazards model, i.e. Cox regression by generalized linear modelling, particularly if covariates have to be taken into account.

For example, figure 15.1 shows the result from analyzing the test file `survive.tf4`, which contains times for both failure and right censoring. Note that more advanced versions of such plots are described on page 262. Also, the parameter estimates from fitting the Weibull model (page 424) by maximum likelihood can be seen in table 15.1. To understand these results, note that if the times t_i are distinct and ordered failure times, i.e. $t_{i-1} < t_i$, and the number in the sample that have not failed by time t_i is n_i, while the number that do fail is d_i, then the estimated probabilities of failure and survival at time t_i are given by

$$\begin{aligned} \hat{p}(\text{failure}) &= d_i/n_i \\ \hat{p}(\text{survival}) &= (n_i - d_i)/n_i. \end{aligned}$$

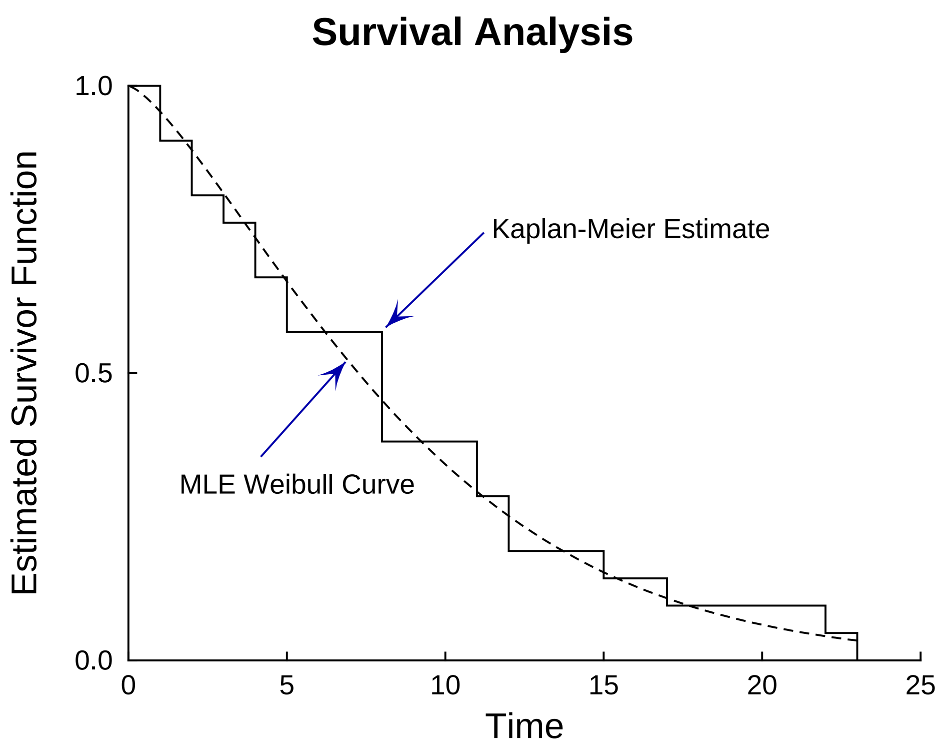


Figure 15.1: Analyzing one set of survival times

```
 Alternative MLE Weibull parameterizations

 S(t) = exp[-{exp(beta)}t^B]
      = exp[-{lambda}t^B]
      = exp[-{A*t}^B]

Parameter     Value     Std. err.    ..95% conf. lim. ..      p
        B      1.371      0.238        0.84       1.90    0.000
     beta     -3.083      0.646       -4.52      -1.64    0.001
   lambda    0.04583     0.0296     -0.0201      0.112    0.153 *
        A     0.1055     0.0177       0.066      0.145    0.000
   t-half      7.257       1.36        4.22       10.3    0.000
 Correlation coefficient(beta,B) = -0.9412
```

Table 15.1: Survival analysis: one sample

The Kaplan-Meier product limit nonparametric estimate of the survivor function (page 419) is defined as a step function which is given in the interval t_i to t_{i+1} by the product of survival probabilities up to time t_i, that is

$$\hat{S}(t) = \prod_{j=1}^{i} \left(\frac{n_j - d_j}{n_j} \right)$$

with variance estimated by Greenwood's formula as

$$\hat{V}(\hat{S}(t)) = \hat{S}(t)^2 \sum_{j=1}^{i} \frac{d_j}{n_j(n_j - d_j)}.$$

It is understood in this calculation that, if failure and censoring occur at the same time, the failure is regarded as having taken place just before that time and the censoring just after it. To understand fitting the Weibull distribution, note that maximum likelihood parameter and standard error estimates are reported for three alternative parameterizations, namely

$$\begin{aligned} S(t) &= \exp(-\exp(\beta)t^B) \\ &= \exp(-\lambda t^B) \\ &= \exp(-(At)^B). \end{aligned}$$

Since the density and survivor function are

$$\begin{aligned} f(t) &= B\lambda t^{B-1}\exp(-\lambda t^B) \\ S(t) &= \exp(-\lambda t^B), \end{aligned}$$

and there are d failures and $n - d$ right censored observations, the likelihood function $l(B, \lambda)$ is proportional to the product of the d densities for the failures in the overall set of n observations and the survivor functions, that is

$$l(B,\lambda) \propto (B\lambda)^d \left(\prod_{i\in D} t_i^{B-1}\right) \exp\left(-\lambda \sum_{i=1}^{n} t_i^B\right)$$

where D is the set of failure times. Actually, the log-likelihood function objective function

$$L(B,\beta) = d\log(B) + d\beta + (B-1)\sum_{i\in D}\log(t_i) - \exp(\beta)\sum_{i=1}^{n} t_i^B$$

with $\lambda = \exp(\beta)$ is better conditioned, so it is maximized and the partial derivatives

$$\begin{aligned} L_1 &= \partial L/\partial\beta \\ L_2 &= \partial L/\partial B \\ L_{11} &= \partial^2 L/\partial\beta^2 \\ L_{12} &= \partial^2 L/\partial B\partial\beta \\ L_{22} &= \partial^2 L/\partial B^2 \end{aligned}$$

are used to form the standard errors and correlation coefficient according to

$$\begin{aligned} \mathrm{se}(\hat{B}) &= \sqrt{-L_{11}/(L_{11}L_{22} - L_{12}^2)} \\ \mathrm{se}(\hat{\beta}) &= \sqrt{-L_{22}/(L_{11}L_{22} - L_{12}^2)} \\ \mathrm{corr}(\hat{B},\hat{\beta}) &= L_{12}/\sqrt{L_{11}L_{22}}. \end{aligned}$$

15.3 Comparing two sets of survival times

As an example of how to compare two data sets, consider the pairwise comparison of the survival times in `survive.tf3` and `survive.tf4`, leading to the results of figure 15.2. Note that you can plot the hazards and the other usual transforms, and do graphical tests for the proportional hazards model. For instance, the transformed Kaplan-Meier nonparametric survivor functions in figure 15.2 should be approximately linear and parallel if the proportional hazards assumption and also the Weibull survival model are justified. To

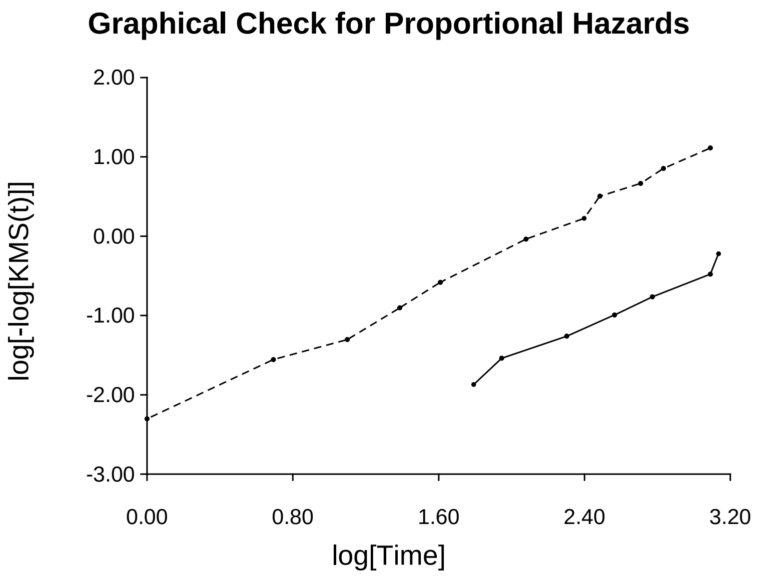


Figure 15.2: Analyzing two sets of survival times

```
Results for the Mantzel-Haenszel (log-rank) test

 H0: h_A(t) = h_B(t)       (equal hazards)
 H1: h_A(t) = theta*h_B(t) (proportional hazards)
 QMH test statistic     = 16.79
 p = P(chi-sq. >= QMH)  = 0.0000 Reject H0 at 1% s-level
 Estimate for theta     = 0.1915
 95% confidence range   = 0.0828, 0.4429
```

Table 15.2: Survival analysis: two samples

prepare your own data you must first browse the test files `survive.tf?` and understand the format (column 1 is time, column 2 is 0 for failure and 1 for right censoring, column 3 is frequency), then use program **makmat**. To understand the graphical and statistical tests used to compare two samples, and to appreciate the results displayed in table 15.2, consider the relationship between the cumulative hazard function $H(t)$ and the hazard function $h(t)$ defined as follows

$$\begin{aligned} h(t) &= f(t)/S(t) \\ H(t) &= \int_0^t h(u)\,du \\ &= -\log(S(t)). \end{aligned}$$

So various graphs can be plotted to explore the form of the cumulative survivor functions for the commonly used models based on the identities

$$\begin{aligned}\text{Exponential} &: H(t) = At\\ \text{Weibull} &: \log(H(t)) = \log A^B + B\log t\\ \text{Gompertz} &: \log(h(t)) = \log B + At\\ \text{Extreme value} &: \log(H(t)) = \alpha(t-\beta).\end{aligned}$$

For instance, for the Weibull distribution, a plot of $\log(-\log(\hat{S}(t))$ against $\log t$, i.e. of the type plotted in figure 15.2, should be linear, and the proportional hazards assumption would merely alter the constant term since, for $h(t) = \theta AB(At)^{B-1}$,

$$\log(-\log(S(t)) = \log\theta + \log A^B + B\log t.$$

Testing for the presence of a constant of proportionality in the proportional hazards assumption amounts to testing the value of θ with respect to unity. If the confidence limits in table 15.2 enclose 1, this can be taken as suggesting equality of the two hazard functions, and hence equality of the two distributions, since equal hazards implies equal distributions. The QMH statistic given in table 15.2 can be used in a chi-square test with one degree of freedom for equality of distributions, and it arises by considering the 2 by 2 contingency tables at each distinct time point t_j of the following type.

	Died	Survived	Total
Group A	d_{jA}	$n_{jA} - d_{jA}$	n_{jA}
Group B	d_{jB}	$n_{jB} - d_{jB}$	n_{jB}
Total	d_j	$n_j - d_j$	n_j

Here the total number at risk n_j at time t_j also includes subjects subsequently censored, while the numbers d_{jA} and d_{jB} actually dying can be used to estimate expectations and variances such as

$$\begin{aligned}E(d_{jA}) &= n_{jA}d_j/n_j\\ V(d_{jA}) &= \frac{d_j(n_j - d_j)n_{jA}n_{jB}}{n_j^2(n_j-1)}.\end{aligned}$$

Now, using the sums

$$\begin{aligned}O_A &= \sum d_{jA}\\ E_A &= \sum E(d_{jA})\\ V_A &= \sum V(d_{jA})\end{aligned}$$

as in the Mantel-Haenszel test, the log rank statistic can be calculated as

$$QMH = \frac{(O_A - E_A)^2}{V_A}.$$

Clearly, the graphical test, the value of θ, the 95% confidence range, and the chi-square test with one degree of freedom support the assumption of a Weibull distribution with proportional hazards in this case. The advanced technique for plotting survival analysis data is described on page 262.

15.4 Survival analysis using generalized linear models

Many survival models can be fitted to n uncensored and m right censored survival times with associated explanatory variables using the GLM technique from **linfit**, **gcfit** in mode 4, or **simstat**. For instance, the simplified interface allows you to read in data for the covariates, x, the variable y which can be either 1 for right-censoring or 0 for failure, together with the times t in order to fit survival models. With a density $f(t)$, survivor function $S(t) = 1 - F(t)$ and hazard function $h(t) = f(t)/S(t)$ a proportional hazards model is assumed for $t \geq 0$ with

$$\begin{aligned}
h(t_i) &= \lambda(t_i)\exp(\sum_j \beta_j x_{ij}) \\
&= \lambda(t_i)\exp(\beta^T x_i) \\
\Lambda(t) &= \int_0^t \lambda(u)\,du \\
f(t) &= \lambda(t)\exp(\beta^T x - \Lambda(t)\exp(\beta^T x)) \\
S(t) &= \exp(-\Lambda(t)\exp(\beta^T x)).
\end{aligned}$$

15.4.1 The exponential survival model

The exponential model has constant hazard and is particularly easy to fit, since

$$\begin{aligned}
\eta &= \beta^T x \\
f(t) &= \exp(\eta - t\exp(\eta)) \\
F(t) &= 1 - \exp(-t\exp(\eta)) \\
\lambda(t) &= 1 \\
\Lambda(t) &= t \\
h(t) &= \exp(\eta) \\
\text{and } E(t) &= \exp(-\eta),
\end{aligned}$$

so this simply involves fitting a GLM model with Poisson error type, a log link, and a calculated offset of $\log(t)$. The selection of a Poisson error type, the log link and the calculation of offsets are all done automatically by the simplified interface from the data provided, as will be appreciated on fitting the test file `cox.tf1`. It should be emphasized that the values for y in the simplified GLM procedure for survival analysis must be either $y = 0$ for failure or $y = 1$ for right censoring, and the actual time for failure t must be supplied paired with the y values. Internally, the SimFIT simplified GLM interface reverses the y values to define the Poisson variables and uses the t values to calculate offsets automatically. Users who wish to use the advanced GLM interface for survival analysis must be careful to declare the Poisson variables correctly and provide the appropriate offsets as offset vectors. Results from the analysis of `cox.tf1` are shown in table 15.3.

15.4.2 The Weibull survival model

Weibull survival is similarly easy to fit, but is much more versatile than the exponential model on account of the extra shape parameter α as in the following equations.

$$\begin{aligned}
f(t) &= \alpha t^{\alpha-1}\exp(\eta - t^\alpha \exp(\eta)) \\
F(t) &= 1 - \exp(-t\exp(\eta)) \\
\lambda(t) &= \alpha t^{\alpha-1} \\
\Lambda(t) &= t^\alpha \\
h(t) &= \alpha t^{\alpha-1}\exp(\eta) \\
E(t) &= \Gamma(1 + 1/\alpha)\exp(-\eta/\alpha).
\end{aligned}$$

However, this time, the offset is $\alpha\log(t)$, where α has to be estimated iteratively and the covariance matrix subsequently adjusted to allow for the extra parameter α that has been estimated. The iteration to estimate α and covariance matrix adjustments are done automatically by the SimFIT simplified GLM interface, and the deviance is also adjusted by a term $-2n\log\hat{\alpha}$.

```
Model: exponential survival
 Number of parameters = 4, Rank = 4, Number of points = 33, NDOF = 29
 Parameter        Value     95% conf. limits      Std.error     p
  Constant       -5.150     -6.201      -4.098      0.5142   0.0000
      B(1)       0.4818     0.1146       0.849      0.1795   0.0119
      B(2)         1.87      0.374       3.367      0.7317   0.0161
      B(3)      -0.3278     -0.831      0.1754      0.2460   0.1931  **
Deviance = 38.55, A = 1.0

Model: Weibull survival
 Number of parameters = 4, Rank = 4, Number of points = 33, NDOF = 29
 Parameter        Value     95% conf. limits      Std.error     p
  Constant       -5.041     -6.182      -3.899       0.558   0.0000
      B(1)       0.4761     0.1079      0.8443        0.18   0.0131
      B(2)        1.841     0.3382       3.344      0.7349   0.0181
      B(3)      -0.3244    -0.8286      0.1798      0.2465   0.1985  **
     Alpha       0.9777      0.889       1.066     0.04336   0.0000
Deviance = 37.06
Deviance - 2n*log[alpha] = 38.55

Model: Cox proportional hazards
 Number of parameters = 3, Number of points = 33, NDOF = 30
 Parameter        Value     95% conf. limits      Std.error     p
      B(1)       0.7325     0.2483       1.217       0.2371   0.0043
      B(2)        2.756     0.7313        4.78       0.9913   0.0093
      B(3)      -0.5792     -1.188     0.02962       0.2981   0.0615   *
 Deviance = 131.5
```

Table 15.3: GLM survival analysis

15.4.3 The extreme value survival model

Extreme value survival is defined by

$$f(t) = \alpha \exp(\alpha t) \exp(\eta - \exp(\alpha t + \eta))$$

which is easily fitted, as it is transformed by $u = \exp(t)$ into Weibull form, and so can be fitted as a Weibull model using t instead of $\log(t)$ as offset. However it is not so useful as a model since the hazard increases exponentially and the density is skewed to the left.

15.4.4 The Cox proportional hazards model

This model assumes an arbitrary baseline hazard function $\lambda_0(t)$ so that the hazard function is

$$h(t) = \lambda_0(t) \exp(\eta).$$

It should first be noted that Cox regression techniques may often yield slightly different parameter estimates, as these will often depend on the starting estimates, and also since there are alternative procedures for allowing for ties in the data. In order to allow for Cox's exact treatment of ties in the data, i.e., more than one failure or censoring at each time point, this model is fitted by the SIMFIT GLM techniques after first calculating the risk sets at failure times t_i, that is, the sets of subjects that fail or are censored at time t_i plus those who survive beyond time t_i. Then the model is fitted using the technique for conditional logistic analysis of stratified data (page 5.5). The model does not involve calculating an explicit constant as that is subsumed into the arbitrary baseline function. However, the model can accommodate strata in two ways. With just a few strata, dummy indicator variables can be defined as in test files `cox.tf2` and `cox.tf3` but, with large numbers of strata, data should be prepared as for `cox.tf4`.

As an example, consider the results shown in table 15.3 from fitting an exponential, Weibull, then Cox model to data in the test file `cox.tf1`. In this case there is little improvement from fitting a Weibull model after an exponential model, as shown by the deviances and half normal residuals plots. The deviances from the full models (exponential, Weibull, extreme value) can be compared for goodness of fit, but they can not be compared directly to the Cox deviance.

15.5 Comprehensive Cox regression

Note that the Cox model can be completed by assuming a baseline hazard function, such as a piecewise exponential function, and the advantage in doing this is so that the survivor functions for the strata can be computed and the residuals can be used for goodness of fit analysis. Table 15.4 shows the results from analysis of `cox.tf4` using the comprehensive Cox regression procedure to calculate parameter scores, residuals, and survivor functions, in addition to parameter estimates.

```
Cox regression: 1, Data set: 1, Offset included: No
Deviance = 1.09.25, Number of time points = 50
B(i)   Estimate     Score   Lower95%cl Upper95%cl Std.error    p
  1    -0.48926  0.00008156    -1.423      0.4447     0.4643 0.2973  ***
  2     0.16086 -0.00002865    -0.7239       1.046     0.4398 0.7162  ***
  3      1.5749   0.0002992     0.5619       2.588     0.5036 0.0030
```

Table 15.4: Cox regression parameters

Also, figure 15.3 illustrates the plots of survivor functions.

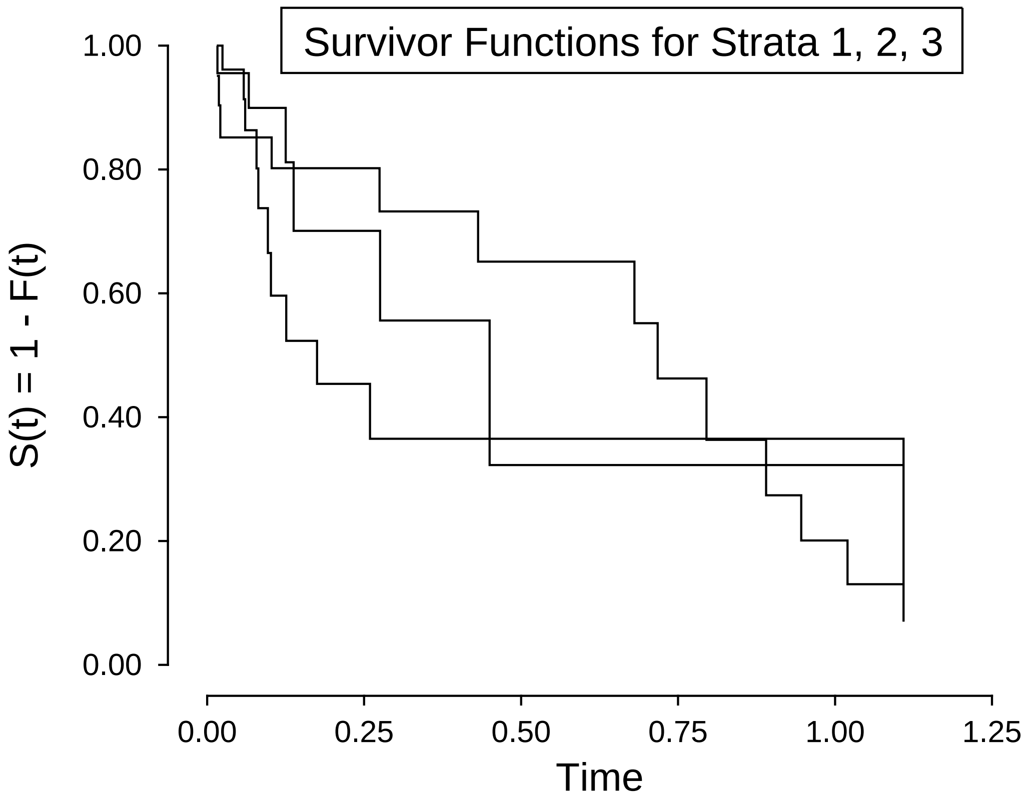


Figure 15.3: Cox regression survivor functions

This data set has three covariates and three strata, hence there are three survivor functions, one for each stratum. It is frequently beneficial to plot the survivor functions in order to visualize the differences in survival between different sub-groups, i.e., strata, and in this case, the differences are clear. It should be pointed out that parameter estimates using the comprehensive procedure may be slightly different from parameter estimates obtained by the GLM procedure if there are ties in the data, as the Breslow approximation for ties may sometimes be used by the comprehensive procedure, unlike the Cox exact method which is employed by the GLM procedures.

Another advantage of the comprehensive procedure is that experienced users can input a vector of offsets, as the assumed model is actually

$$\lambda(t,x) = \lambda_0(t)\exp(\beta^T x + \omega)$$

for parameters β, covariates x and offset ω. Then the maximum likelihood estimates for β are obtained by maximizing the Kalbfleisch and Prentice approximate marginal likelihood

$$L = \prod_{i=1}^{n_d} \frac{\exp(\beta^T s_i + \omega_i)}{[\sum_{l \in R(t_{(i)})} \exp(\beta^T x_l + \omega_l)]^{d_i}}$$

where, n_d is the number of distinct failure times, s_i is the sum of the covariates of individuals observed to fail at $t_{(i)}$, and $R(t_{(i)})$ is the set of individuals at risk just prior to $t_{(i)}$.

In the case of multiple strata, the likelihood function is taken to be the product of such expressions, one for each stratum. For example, with ν strata, the marginal likelihood will be

$$L = \prod_{k=1}^{\nu} L_k.$$

Once parameters have been estimated the survivor function $\exp(-\hat{H}(t_{(i)})$ and residuals $r(t_l)$ are then calculated using

$$\hat{H}(t_{(i)}) = \sum_{t_j \le t_i} \left(\frac{d_j}{\sum_{l \in R(t_{(j)})} \exp(\hat{\beta}^T x_l + \omega_l)} \right)$$
$$r(t_l) = \hat{H}(t_l)\exp(\hat{\beta}^T x_l + \omega_l),$$

where there are d_j failures at t_j.

15.6 Plotting censored survival data

It is often necessary to display survival data for two groups in order to assess the relative hazards. For instance, using **gcfit** in mode 3, or alternatively **simstat**, to analyze the test files `survive.tf5` and `survive.tf6`, which contain survival data for stage 3 (group A) and stage 4 (group B) tumors, yields the result that the samples differ significantly according to the Mantel-Haenszel log rank test, as in table 15.5.

```
Results for the Mantel-Haenszel (log-rank) test

H0: h_A(t) = h_B(t)       (equal hazards)
H1: h_A(t) = theta*h_B(t) (proportional hazards)
QMH test statistic      = 6.71
p = P(chi-sq. >= QMH)   = 0.0096 Reject H0 at 1% s-level
Estimate for theta      = 0.3786
95% confidence range    = 0.1795, 0.7982
```

Table 15.5: Mantel-Haenzel log rank test

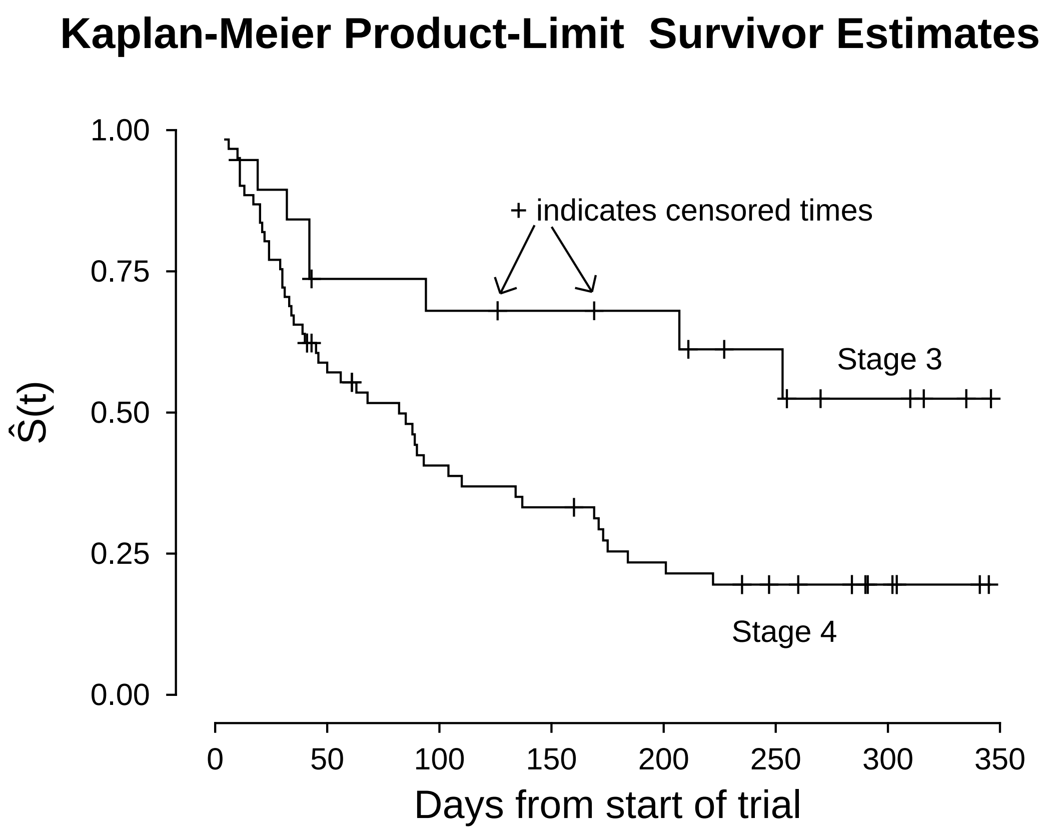


Figure 15.4: Plotting censored survival times

Figure 15.4 illustrates the Kaplan-Meier product limit survivor functions for these data using the advanced survival analysis plotting mode.

In this plot, the survivor functions are extended to the end of the data range for both sets of data, even though no deaths occurred at the extreme end of the time range. There are two features of this graph that should be indicated. The first is that the coordinates supplied for plotting are in the form of a stair step type of survival curve with steps and corners, so that alternative line types and colors can be used. Another point is that censored observations are also plotted, in this case using plus signs as plotting symbols, which results in the censored observations being identified as short vertical lines.

Attention should be drawn to the fact that, in survival analysis, SIMFIT calculates the starting sample size from the sum of all frequencies supplied in the data file either as right censored subjects or failures, so there two possibilities.

1. The data file contains an entry for every case with an indication of either censorship or failure, i.e. every frequency is reported as 1.
2. The data file contains results in compressed format, so that frequencies are provided for the numbers of subjects censored or failing at sampling times.

In the first case all subjects remaining at the end of the observation period must be reported as censored observations, while in the second case a final frequency must be supplied representing all subjects remaining at the end as censored. View the test files `survive.tf5` and `survive.tf6` for further clarification.

Part 16

Areas, slopes, lag times and asymptotes

16.1 Introduction

It frequently happens that measurements of a response y as a function of time t are made in order to measure an initial rate, a lag time, an asymptotic steady state rate, a horizontal asymptote or an area under the curve (AUC). Examples could be the initial rate of an enzyme catalyzed reaction or the transport of labeled solute out of loaded erythrocytes. Stated in equations we have the responses

$$y_i = f(t_i) + \epsilon_i,\ i = 1, 2, \ldots, n$$

given by a deterministic component plus a random error and it is wished to measure the following limiting values

$$\begin{aligned}
\text{the initial rate } &= \frac{df}{dt} \text{ at } t = 0\\
\text{the asymptotic slope } &= \frac{df}{dt} \text{ as } t \longrightarrow \infty\\
\text{the final asymptote } &= f \text{ as } t \longrightarrow \infty\\
\text{the AUC } &= \int_{\alpha}^{\beta} f(t)\,dt.
\end{aligned}$$

There are numerous ways to make such estimates in SIMFIT and the method adopted depends critically on the type of experiment. Choosing the wrong technique can lead to biased estimates, so you should be quite clear which is the correct method for your particular requirements.

16.2 Models used by program inrate

The models used in this program are

$$\begin{aligned}
f_1 &= Bt + C\\
f_2 &= At^2 + Bt + C\\
f_3 &= \alpha[1 - \exp(-\beta t)] + C\\
f_4 &= \frac{Vt^n}{K^n + t^n} + C\\
f_5 &= Pt + Q[1 - \exp(-Rt)] + C
\end{aligned}$$

and there are test files to illustrate each of these. It is usual to assume that $f(t)$ is an increasing function of t with $f(0) = 0$, which is easily arranged by suitably transforming any initial rate data. For instance, if you have measured efflux of an isotope from vesicles you would analyze the rate of appearance in the external solute, that is, express your results as

$$f(t) = \text{initial counts - counts at time } t$$

so that $f(t)$ increase from zero at time $t = 0$. All you need to remember is that, for any constant K,

$$\frac{d}{dt}\{K - f(t)\} = -\frac{df}{dt}.$$

However it is sometimes difficult to know exactly when $t = 0$, e.g., if the experiment involves quenching, so there exists an option to force the best fit curve to pass through the origin with some of the models if this is essential. The models available will now be summarized.

1. f_1: This is used when the data are very close to a straight line and it can only measure initial rates.
2. f_2: This adds a quadratic correction and is used when the data suggest only a slight curvature. Like the previous it can only estimate initial rates.
3. f_3: This model is used when the data rapidly bend to a horizontal asymptote in an exponential manner. It can be used to estimate initial rates and final horizontal asymptotes.
4. f_4: This model can be used with n fixed (e.g., $n = 1$) for the Michaelis-Menten equation or with n varied (the Hill equation). It is not used for initial rates but is sometimes better for estimating final horizontal asymptotes than the previous model.
5. f_5: This is the progress curve equation used in transient enzyme kinetics. It is used when the data have an initial lag phase followed by an asymptotic final steady state. It is not used to estimate initial rates, final horizontal asymptotes or AUC. However, it is very useful for experiments with cells or vesicles which require a certain time before attaining a steady state, and where it is wished to estimate both the length of lag phase and the final steady state rate.

To understand these issues, see what happens the test files. These are, models f_1 and f_2 with `inrate.tf1`, model f_3 with `inrate.tf2`, model f_4 with `inrate.tf3` and model f_5 using `inrate.tf4`.

16.2.1 Estimating initial rates using inrate

A useful method to estimate initial rates when the true deterministic equation is unknown is to fit quadratic $At^2 + Bt + C$, in order to avoid the bias that would inevitably result from fitting a line to nonlinear data. Use **inrate** to fit the test file `inrate.tf1`, and note that, when the model has been fitted, it also estimates the slope at the origin. The reason for displaying the tangent in this way, as in figure 16.1, is to give you some idea of what is involved in extrapolating the best fit curve to the origin, so that you will not accept the estimated initial rate uncritically.

16.2.2 Lag times and steady states using inrate

Use **inrate** to fit $Pt + Q[1 - \exp(-Rt)] + C$ to `inrate.tf4` and observe that the asymptotic line is displayed in addition to the tangent at the origin, as in figure 16.2. However, sometimes a burst phase is appropriate, rather than lag phase, as figure 16.3.

16.3 Model-free fitting using compare

SimFIT can fit arbitrary models, where the main interest is data smoothing by model-free or nonparametric techniques, rather than fitting mathematical models. For instance, **polnom** fits polynomials while **calcurve** fits splines for calibration. For now we shall use **compare** to fit the test files `compare.tf1` and `compare.tf2` as in figure 16.4, where the aim is to compare two data sets by model free curve fitting using the automatic spline knot placement technique described on page 270. Table 16.1 then summarizes the differences reported

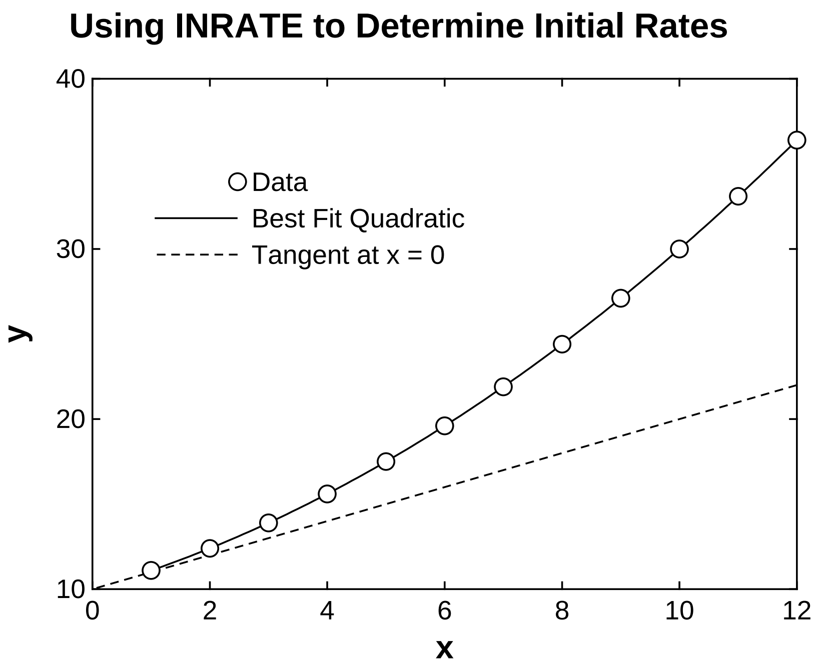


Figure 16.1: Fitting initial rates

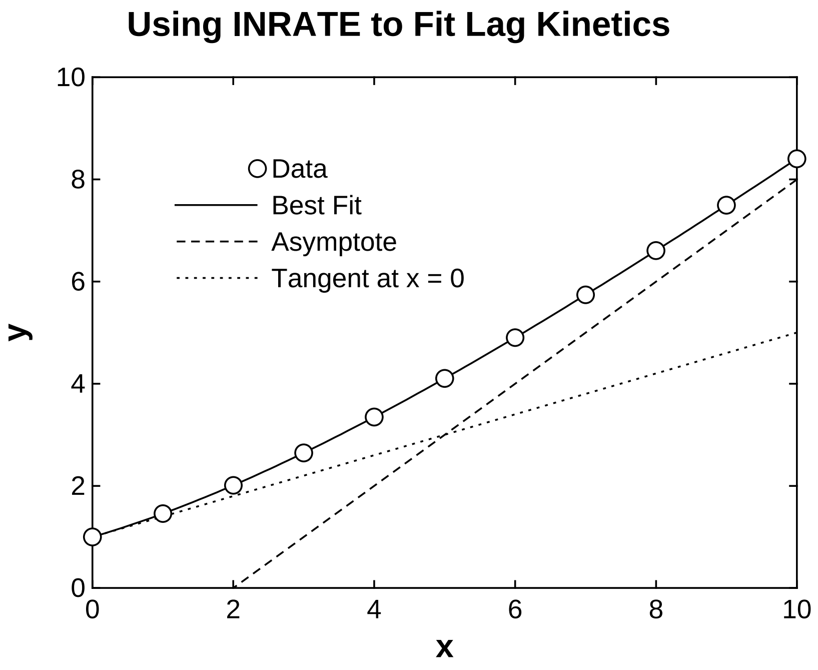


Figure 16.2: Fitting lag times

for the two curves shown in figure 16.4. Program **compare** reads in one or two data sets, calculates means and standard errors of means from replicates, fits constrained splines, and then compares the two fits. You can

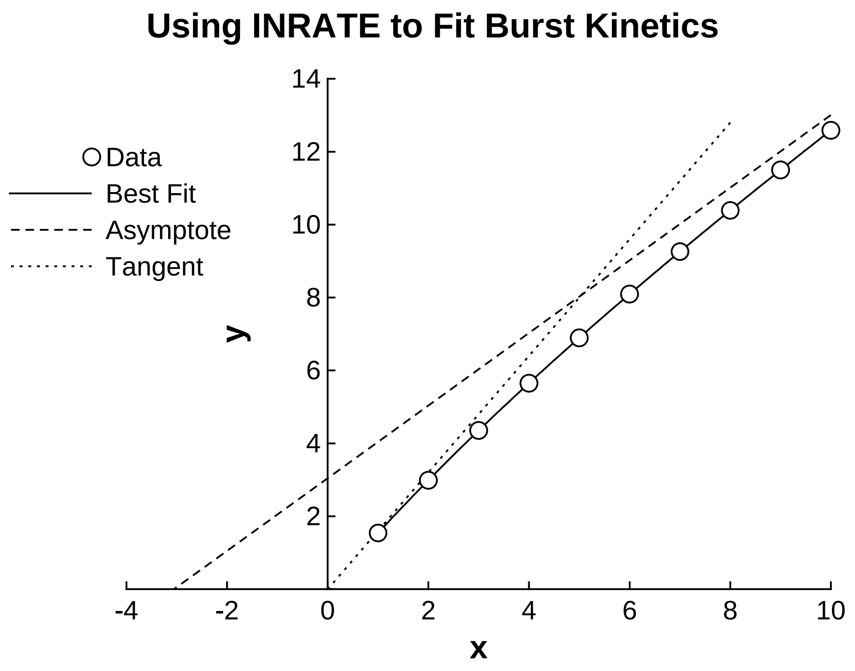


Figure 16.3: Fitting burst kinetics

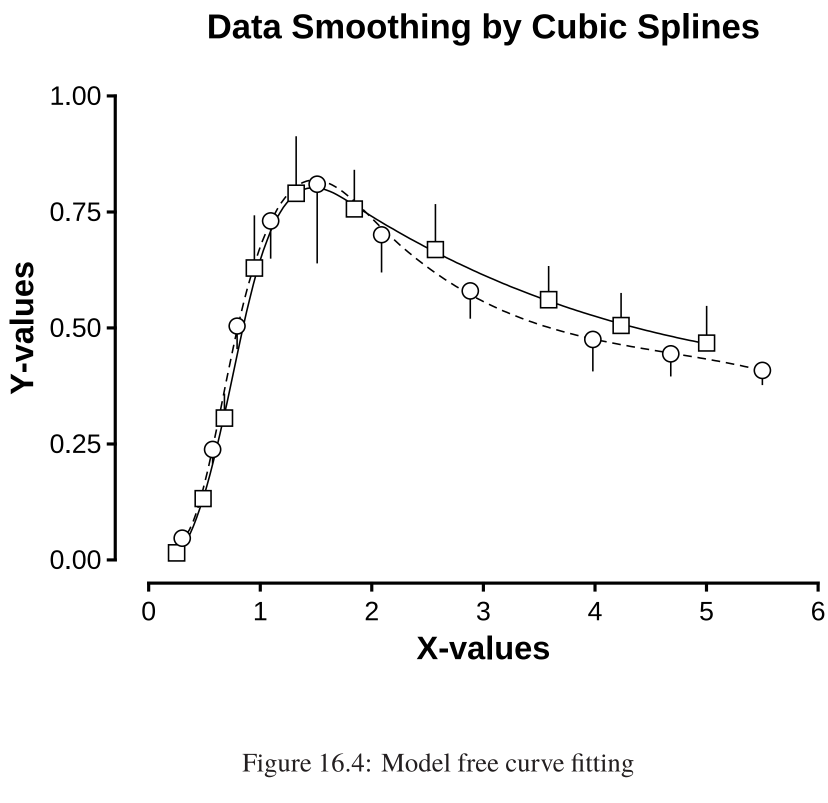


Figure 16.4: Model free curve fitting

change a smoothing factor until the fit is acceptable and you can use the spline coefficients for calculations, or store them for re-use by program **spline**. Using spline coefficients you can plot curve sections, estimate

```
Area under curve 1 (   0.25 < x <   5.0   )       (A1) = 2.69
Area under curve 2 (    0.3 < x <   5.5   )       (A2) = 2.84
For window number 1:    0.3 < x <   5.0,        y_min = 0.0
Area under curve 1 inside window 1              (B1) = 2.69
Area under curve 2 inside window 1              (B2) = 2.63
Integral of |curve1 - curve2| for the x_overlap (AA) = 0.262
For window number 2:    0.3 < x <   5.0,       y_min = 0.028
Area under curve 1 inside window 2              (C1) = 2.56
Area under curve 2 inside window 2              (C2) = 2.50
Estimated percentage differences between the curves:
Over total range of x values: 100|A1 - A2|/(A1 + A2) = 2.63 %
In window 1 (with a zero baseline): 100*AA/(B1 + B2) = 4.92 %
In window 2 (with  y_min baseline): 100*AA/(C1 + C2) = 5.18 %
```

Table 16.1: Comparing two data sets

derivatives and areas, calculate the arc length and total absolute curvature of a curve, or characterize and compare data sets which do not conform to known mathematical models. Comparing raw data sets with profiles as in figure 16.4 is complicated by the fact that there may be different numbers of observations, and observations may not have been made at the same x values. Program **compare** replaces a comparison of two data sets by a comparison of two best-fit curves, chosen by data smoothing. Two windows are defined by the data sets as well as a window of overlap, and these would be identical if both data sets had the same x-range. Perhaps the absolute area between the two curves over the range where the data sets overlap AA is the most useful parameter, which may be easier to interpret as a percentage. Note that, where data points or fitted curves have negative y values, areas are replaced by areas with respect to a baseline in order to remove ambiguity and makes areas positive over any window within the range set by the data extremes. The program also reports the areas calculated by the trapezoidal method, but the calculations reported in table 16.1 are based on numerical integration of the best-fit spline curves.

16.4 Estimating averages and AUC using deterministic equations

Observations y_i are often made at settings of a variable x_i as for a regression, but where the main aim is to determine the area under a best fit theoretical curve AUC rather than any best fit parameters. Frequently also $y_i > 0$, which is the case we now consider, so that there can be no ambiguity concerning the definition of the area under the curve. One example would be to determine the average value f_{average} of a function $f(x)$ for $\alpha \le x \le \beta$ defined as

$$f_{\text{average}} = \frac{1}{\beta - \alpha} \int_\alpha^\beta f(u)\, du.$$

Another example is motivated by the practise of fitting an exponential curve in order to determine an elimination constant k by extrapolation, since

$$\int_0^\infty \exp(-kt)\, dt = \frac{1}{k}.$$

Yet again, given any arbitrary function $g(x)$, where $g(x) \ge 0$ for $\alpha \le x \le \beta$, a probability density function f_T can always be constructed for a random variable T using

$$f_T(t) = \frac{g(t)}{\displaystyle\int_\alpha^\beta g(u)\, du}$$

which can then be used to model residence times, etc. If the data do have a known form, then fitting an appropriate equation is probably the best way to estimate slopes and areas. For instance, in pharmacokinetics you can use program **exfit** to fit sums of exponentials and also estimate areas over the data range and AUC by extrapolation from zero to infinity since

$$\int_0^\infty \sum_{i=1}^n A_i \exp(-k_i t)\, dt = \sum_{i=1}^n \frac{A_i}{k_i}$$

which is calculated as a derived parameter with associated standard error and confidence limits. Other deterministic equations can be fitted using program **qnfit** since, after this program has fitted the requested equation from the library or your own user-supplied model, you have the option to estimate slopes and areas using the current best-fit curve.

16.5 Estimating AUC using average

The main objection to using a deterministic equation to estimate the AUC stems from the fact that, if a badly fitting model is fitted, biased estimates for the areas will result. For this reason, it is frequently better to consider the observations y_i, or the average value of the observations if there are replicates, as knots with coordinates x_i, y_i defining a linear piecewise spline function. This can then be used to calculate the area for any sub range a, b where $A \le a \le b \le B$.

To practise, read `average.tf1` into program **average** and create a plot like figure 16.5. Another use for the

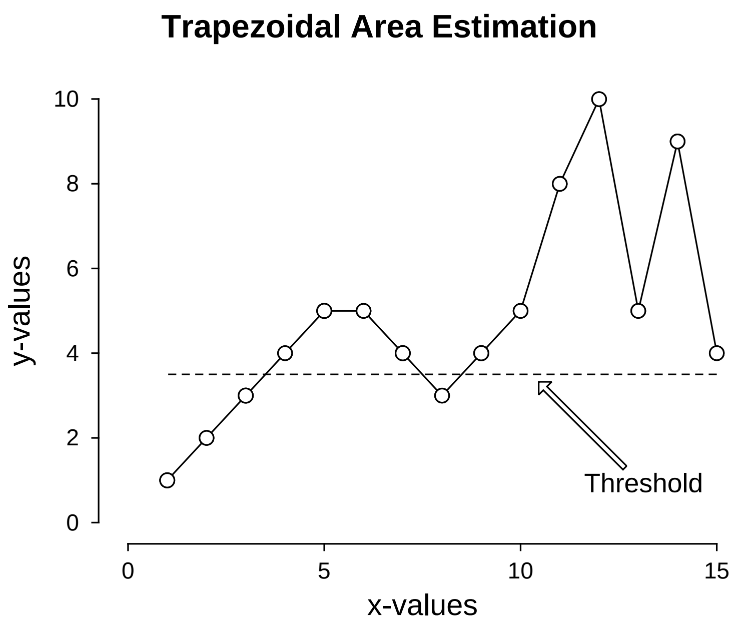


Figure 16.5: Trapezoidal method for areas/thresholds

trapezoidal technique is to calculate areas above or below a baseline, or fractions of the x range above and below a threshold, for example, to record the fraction of a certain time interval that a patients blood pressure was above a baseline value. Note that, in figure 16.5, the base line was set at $y = 3.5$, and program **average** calculates the points of intersection of the horizontal threshold with the linear spline in order to work out fractions of the x range above and below the baseline threshold. For further versatility, you can select the end points of interest, but of course it is not possible to extrapolate beyond the data range to estimate AUC from zero to infinity.

Part 17

Spline smoothing

17.1 Introduction

It often happens that a mathematical model is not available for a given data set because of one of these reasons.

- ❏ The data are too sparse or noisy to justify deterministic model fitting.
- ❏ The fundamental processes involved in generating the data set are unknown.
- ❏ The mathematical description of the data is too complex to warrant model fitting.
- ❏ The error structure of the data is unknown, so maximum likelihood cannot be invoked.
- ❏ The users merely want a smooth representation of the data to display trends, calculate derivatives, or areas, or to use as a standard curve to predict x given y.

The traditional method to model such situations was to fit a polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

by weighted least squares, where the degree n was adjusted to obtain optimum fit. However, such a procedure is seldom adopted nowadays because of the realization that polynomials are too flexible, allowing over-fit, and because polynomials cannot fit the horizontal asymptotes that are often encountered in experimental data, e.g. growth curves, or dose response curves. Because of these restrictions, polynomials are only used in situations where data sets are monotonic and without horizontal asymptotes, and only local modeling is anticipated, with no expectation of meaningful extrapolation beyond the limits of the data.

Where such model free fitting is required, then simple deterministic models, such as the exponential or logistic models, can often be useful. However, here the problem of systematic bias can be encountered, where the fixed curve shape of the simple model can lead to meaningless extrapolation or prediction. To circumvent this problem, piecewise cubic splines can be used, where a certain number of knot positions are prescribed and, between the knot positions, cubic polynomials are fitted, one between each knot, with the desirable property of identical function and derivative value at the knots. Here again it is necessary to impose additional constraints on the splines and knot placements, otherwise under or over fitting can easily result, particularly when the splines attempt to fit outliers leading to undulating best fit curves.

SIMFIT allows users to fit one of four types of spline curve.

1. User-defined fixed knots
2. Automatically calculated knots
3. In between knots: smoothing factor input

4. In between knots: smoothing factor by generalized cross validation

Given n data values x, y, s and m knots, then each type of spline curve fitting technique minimizes an objective function involving the weighted sum of squares $WSSQ$ given by

$$WSSQ = \sum_{i=1}^{n} \left\{ \frac{y_i - f(x_i)}{s_i} \right\}^2$$

where $f(t)$ is the spline curve defined piecewise between the m knots, but each type of spline curve has advantages and limitations, which will be discussed after dealing with the subject of replicates. All x, y values must be supplied, and the s values should either be all equal to 1 for unweighted fitting, or equal to the standard deviation of y otherwise (see page 29).

It frequently happens that data sets contain replicates and, to avoid confusion, SIMFIT automatically compresses data sets with replicates before fitting the splines, but then reports residuals and other goodness of fit criteria in terms of the full data set. If there are groups of replicates, then the sample standard deviations within groups of replicates are calculated interactively for weighting, and the s values supplied are used for single observations. Suppose that there are n observations but N distinct x values x_j and at each of these there are k_j replicates, where all of the replicates have the same s value s_j at $x = x_j$ for weighting. Then we would have

$$n = \sum_{j=1}^{N} k_j$$

$$\bar{y}_j = (1/k_j) \sum_{i=l}^{k_j+l-1} y_i$$

$$WSSQ_N = \sum_{j=1}^{N} \left\{ \frac{\bar{y}_j - f(x_j)}{s_j/k_j} \right\}^2$$

where l is the index of x in the full data set when rearranged into nondecreasing order where a group of replicates start. So, whether users input all n replicates with $s = 1$ or the standard deviation of y, or just N mean values with s_j equal to the standard errors of the means $\bar{y}_j$, the same spline will result. However, incorrect goodness of fit statistics, such as the runs and signs tests or half normal residuals plots, will result if means are supplied instead of all replicates.

17.2 User-defined fixed knots

Here the user must specify the number of interior knots and their spacing in such a way that genuine dips, spikes or asymptotes in the data can be modeled by clustering knots appropriately. Four knots are added automatically to correspond to the smallest x value, and four more are also added to equal the largest x value. If the data are monotonic and have no such spike features, then equal spacing can be resorted to, so users only need to specify the actual number of interior knots. The programs **calcurve** and **csafit** offer users both of these techniques, as knot values can be provided after the termination of the data values in the data file, while program **spline** provides the best interface for interactive spline fitting. Fixed knot splines have the advantage that the effect of the number of knots on the best fit curve is fully intuitive; too few knots lead to under-fit, while too many knots cause over-fit. Figure 17.1 illustrates the effect of changing the number of equally spaced knots when fitting the data in `compare.tf1` by this technique. The vertical bars at the knot positions were generated by replacing the default symbols (dots) by narrow (size 0.05) solid bar-chart type bars. It is clear that the the fit with one interior knot is quite sufficient to account for the shape of the data, while using four gives a better fit at the expense of excessive undulation. To overcome this limitation of fixed knots SIMFIT provides the facility to provide knots that can be placed in specified patterns and, to illustrate this, figure 17.2 illustrates several aspects of the fit to `e02baf.tf1`. The left hand figure shows the result when spline knots were input from the spline file `e02baf.tf2`, while the right hand figure shows how program

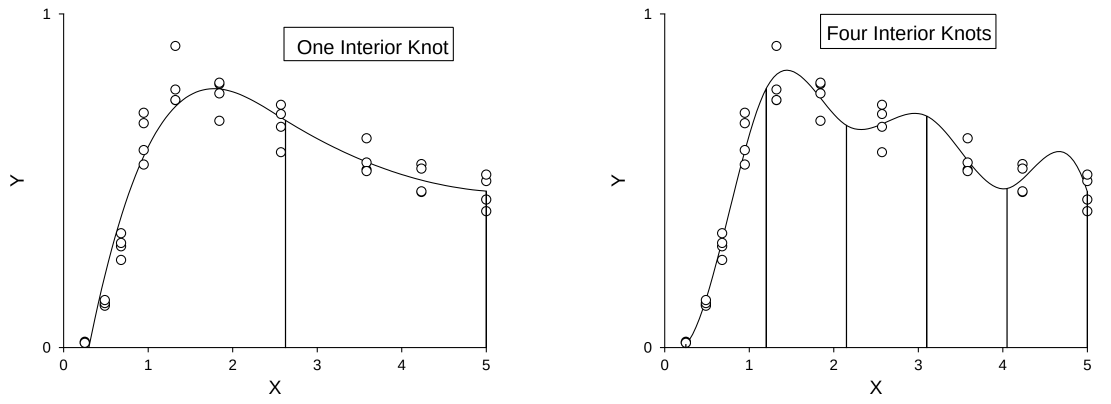


Figure 17.1: Splines: equally spaced interior knots

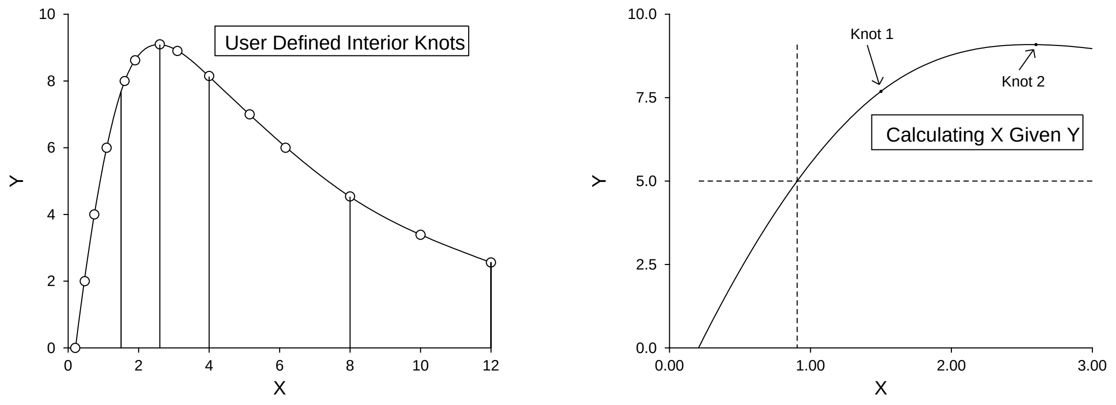


Figure 17.2: Splines: user spaced interior knots

spline can be used to predict X given values of Y. Users simply specify a range of X within the range set by the data, and a value of Y, whereupon the intersection of the dashed horizontal line at the the specified value of Y is calculated numerically, and projected down to the X value predicted by the vertical dashed line. Note that, after fitting `e02baf.tf1` using knots defined in `e02baf.tf2`, the best fit spline curve was saved to the file `spline.tf1` which can then always be input again into program **spline** to use as a deterministic equation between the limits set by the data in `e02baf.tf1`.

17.3 Automatically calculated knots

Here the knots are generated automatically and the spline is calculated to minimize

$$\eta = \sum_{i=5}^{m-5} \delta_i^2,$$

where δ_i is the discontinuity jump in the third derivative of the spline at the interior knot i, subject to the constraint

$$0 \le WSSQ_N \le F$$

where F is user-specified. If F is too large there will be under-fit and best fit curve will be unsatisfactory, but if F is too small there will be over-fit. For example, setting $F = 0$ will lead to an interpolating spline passing through every point, while choosing a large F value will produce a best-fit cubic polynomial with $\eta = 0$ and no internal knots. In weighted least squares fitting $WSSQ$ will often be approximately a chi-square variable with degrees of freedom equal to the number of experimental points minus the number of parameters fitted, so choosing a value for $F \approx N$ will often be a good place to start. The programs **compare** and **spline** provide extensive options for fitting splines of this type. Figure 17.3, for example, illustrates the effect of

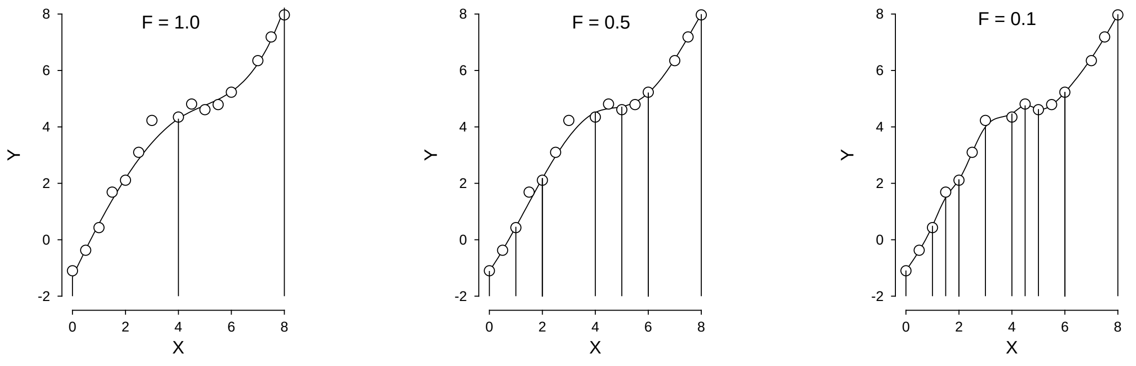


Figure 17.3: Splines: automatically spaced interior knots

fitting `e02bef.tf1` using smoothing factors of 1.0, 0.5, and 0.1.

Table 17.1 presents typical results from the fitting illustrated in figure 17.3.

```
From spline fit with 1 automatic knots, WSSQ = 1.0
From spline fit with 5 automatic knots, WSSQ = 0.5
From spline fit with 8 automatic knots, WSSQ = 0.1

Spline knots and coefficients from fitting the file:
C:\simfit5\temp\e02bef.tf1

 X-value    spline  1st.deriv.  2nd.deriv.  3rd.deriv.
     2.0     2.125       1.462       2.896       12.43
     4.0     4.474      0.5562       1.905       5.211
     6.0     5.224       1.058      0.2932      -1.912

   A       B    Area  s=Arc-length Integral|K|ds  (In degrees)
  0.0     8.0   30.92        12.8          4.569         261.8
  2.0     6.0   16.87       5.574          3.715         212.9
  3.0     5.0    8.97       2.235          2.316         132.7
```

Table 17.1: Spline calculations

17.4 In between knots: ρ input

Here there is one knot between each distinct x value and the spline $f(x)$ is calculated as that which minimizes

$$WSSQ_N + \rho \int_{-\infty}^{\infty} (f''(x))^2 \, dx.$$

As with the automatically generated knots, a large value of the smoothing parameter ρ gives under-fit while $\rho = 0$ generates an interpolating spline, so assigning ρ controls the overall fit and smoothness. As splines are linear in parameters then a matrix H can be found such that

$$\hat{y} = H\bar{y}$$

and the degrees of freedom ν can be defined in terms of the leverages h_{ii} in the usual way as

$$\begin{aligned}\nu &= \text{Trace}(I - H)\\ &= \sum_{i=1}^{N}(1 - h_{ii}).\end{aligned}$$

This leads to two ways to specify the spline coefficients which depend on ρ being fixed by the user or estimated in some way.

The spline can be fixed by specifying the value of ρ. To use this option, the value of ρ is input interactively, and the resulting fit inspected graphically until it is acceptable. This way users have complete control over the amount of smoothing required.

17.5 In between knots: ρ by generalized cross validation

Alternatively, ρ can be estimated by minimizing the generalized cross validation GCV, where

$$GCV = N\left(\frac{\sum_{i=1}^{N} r_i^2}{(\sum_{i=1}^{N}(1 - h_{ii}))^2}\right).$$

As this leads to a unique estimate for ρ, users have no control if the spline leads to either over-smoothing or over-fitting.

17.6 Using splines

As splines are defined by knots and coefficients rather than equations, special techniques are required to re-use best fit spline functions. SIMFIT provides procedures to save spline parameters to a file so that they can be re-used to restore previously fitted spline functions. This is particularly useful when a best-fit spline is to be re-used as reference function or standard curve, as in calibration.

Input a spline file such as `spline.tf1` into program **spline** to appreciate how to re-use a best fit spline stored from **spline**, **calcurve**, or **compare**, to estimate derivatives, areas, curvatures and arc lengths.

SIMFIT spline files of length $k \geq 12$, such as `spline.tf1`, have $(k + 4)/2$ knots, then $(k - 4)/2$ coefficients as follows.

- There must be at least 8 nondecreasing knots
- The first 4 of these knots must all be equal to the lowest x value
- The next $(k - 12)/2$ must be the non-decreasing interior knots
- The next 4 of these knots must all be equal to the highest x value
- Then there must be $(k - 4)/2$ spline coefficients c_i

With $\bar{n}$ spline intervals (i.e. one greater than the number of interior knots), $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are knots corresponding to the lowest x value, $\lambda_5, \lambda_6, \ldots, \lambda_{\bar{n}+3}$ are interior knots, while $\lambda_{\bar{n}+4}, \lambda_{\bar{n}+5}, \lambda_{\bar{n}+6}, \lambda_{\bar{n}+7}$ correspond to the largest x value. Then the best-fit spline $f(x)$ is

$$f(x) = \sum_{i=1}^{\bar{n}+3} c_i B_i(x).$$

where the c_i are the spline coefficients, and the $B_i(x)$ are normalized B-splines of degree 3 defined on the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$. When the knots and coefficients are defined in this way, the function $y = f(x)$ can be used as a model-free best fit curve to obtain point estimates for the derivatives y', y'', y''', as well as the area A, arc length L, or total absolute curvature K over a range $\alpha \le x \le \beta$, defined as

$$\begin{aligned}
A &= \int_\alpha^\beta y\,dx \\
L &= \int_\alpha^\beta \sqrt{1 + y'^2}\,dx \\
K &= \int_0^L \frac{|y''|}{(1 + y'^2)^{\frac{3}{2}}}\,dl \\
&= \int_\alpha^\beta \frac{|y''|}{1 + y'^2}\,dx
\end{aligned}$$

which are valuable parameters to use when comparing data sets. For instance, the arc length s provides a valuable measure of the length of the fitted curve, while the total absolute curvature indicates the total angle turned by the tangent to the curve and indicates the amount of oscillatory behavior.

17.7 Advice on which type of spline to use

The following recommendations indicate which SIMFIT programs to use for spline smoothing. However, as splines can easily lead to over-smoothing or over-fitting, best-fit curves should always be inspected before being accepted.

❏ User-defined fixed knots.
calcurve provides a comprehensive interface for calibration as users can alter knot density and position in addition to data transformation to generate a satisfactory standard curve.

❏ Automatically calculated knots.
compare uses this type of spline fitting to create smoothed representations of two data sets in order to compare the two samples for similarities and differences.

❏ In between knots: smoothing factor input.
spline provides this option which is probably the easiest way to create an acceptable smoothed approximation to a given noisy data set.

❏ In between knots: smoothing factor by generalized cross validation.
spline can use this technique for calibration, data smoothing, and estimation of derivatives, areas, and curvatures. It has the advantage of being fully automatic, as the knots are fixed and the smoothing factor is estimated and not under user control, but can be prone to over-smoothing or over-fitting.

Part 18

Statistical calculations

18.1 Introduction

In data analysis it is frequently necessary to compare theoretical with observed distributions, or perform calculations rather than tests, as discussed in this section.

18.2 Statistical power and sample size

Experiments often generate random samples from a population so that parameters estimated from the samples can be use to test hypotheses about the population parameters. So it is natural to investigate the relationship between sample size and the absolute precision of the estimates, given the expectation $E(X)$ and variance $\sigma^2(X)$ of the random variable. For a single observation, i.e., $n = 1$, the Chebyshev inequality

$$P(|X - E(X)| < \epsilon) \geq 1 - \frac{\sigma^2(X)}{\epsilon^2}$$

with $\epsilon > 0$, indicates that, for an unspecified distribution,

$$\begin{aligned} P(|X - E(X)| < 4.5\sigma(X)) &\geq 0.95, \\ \text{and } P(|X - E(X)| < 10\sigma(X)) &\geq 0.99, \end{aligned}$$

but, for an assumed normal distribution,

$$\begin{aligned} P(|X - E(X)| < 1.96\sigma(X)) &\geq 0.95, \\ \text{and } P(|X - E(X)| < 2.58\sigma(X)) &\geq 0.99. \end{aligned}$$

However, provided that $E(X) \neq 0$, it is more useful to formulate the Chebyshev inequality in terms of the relative precision, that is, for $\delta > 0$

$$P\left(\left|\frac{X - E(X)}{E(X)}\right| < \delta\right) \geq 1 - \frac{1}{\delta^2}\frac{\sigma^2(X)}{E^2(X)}.$$

Now, for an unspecified distribution,

$$\begin{aligned} P\left(\left|\frac{X - E(X)}{E(X)}\right| < 4.5\frac{\sigma(X)}{|E(X)|}\right) &\geq 0.95, \\ \text{and } P\left(\left|\frac{X - E(X)}{E(X)}\right| < 10\frac{\sigma(X)}{|E(X)|}\right) &\geq 0.99, \end{aligned}$$

but, for an assumed normal distribution,

$$\begin{aligned} P\left(\left|\frac{X - E(X)}{E(X)}\right| < 1.96\frac{\sigma(X)}{|E(X)|}\right) &\geq 0.95, \\ \text{and } P\left(\left|\frac{X - E(X)}{E(X)}\right| < 2.58\frac{\sigma(X)}{|E(X)|}\right) &\geq 0.99. \end{aligned}$$

So, for high precision, the coefficient of variation $cv\%$

$$cv\% = 100\frac{\sigma(X)}{|E(X)|}$$

must be as small as possible, while the signal-to-noise ratio $SN(X)$

$$SN(X) = \frac{|E(X)|}{\sigma(X)}$$

must be as large as possible. For instance, for the single measurement to be within 10% of the mean 95% of the time requires $SN \geq 45$ for an arbitrary distribution, or $SN \geq 20$ for a normal distribution. A particularly valuable application of these results concerns the way that the signal-to-noise ratio of sample means depends on the sample size n. From

$$\begin{aligned} \bar{X} &= \frac{1}{n}\sum_{i=1}^{n} x_i, \\ Var(\bar{X}) &= \frac{1}{n^2}\sum_{i=1}^{n} Var(X) \\ &= \frac{1}{n}\sigma^2(X), \end{aligned}$$

it follows that, for arbitrary distributions, the signal-to-noise ratio of the sample mean $SN(\bar{X})$ is given by $SN(\bar{X}) = \sqrt{n}SN(X)$, that is

$$SN(\bar{X}) = \sqrt{n}\frac{E(X)}{\sigma(X)}.$$

This result, known as the law of $\sqrt{n}$, implies that the signal-to-noise ratio of the sample mean as an estimate of the population mean increases as $\sqrt{n}$, so that the the relative error in estimating the mean decreases like $1/\sqrt{n}$.

If $f(x)$ is the density function for a random variable X, then the null and alternative hypotheses can sometimes be expressed as

$$\begin{aligned} H_0 &: f(x) = f_0(x) \\ H_1 &: f(x) = f_1(x) \end{aligned}$$

while the error sizes, given a critical region C, are

$$\begin{aligned} \alpha &= P_{H_0}(\text{reject } H_0) \text{ (i.e., the Type I error)} \\ &= \int_C f_0(x)\,dx \\ \beta &= P_{H_1}(\text{accept } H_0) \text{ (i.e., the Type II error)} \\ &= 1 - \int_C f_1(x)\,dx. \end{aligned}$$

Usually α is referred to as the significance level, β is the operating characteristic, while $1 - \beta$ is the power, frequently expressed as a percentage, i.e., $100(1 - \beta)\%$, and these will both alter as the critical region is

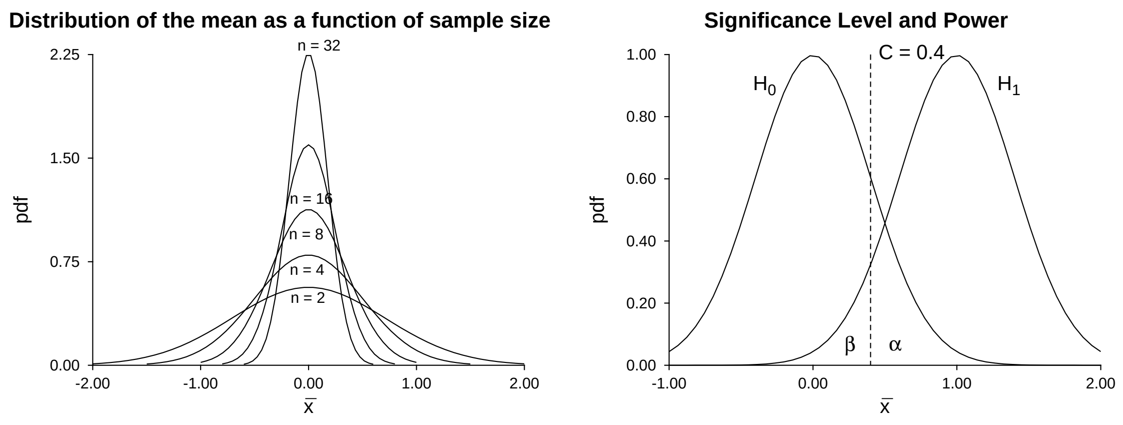


Figure 18.1: Significance level and power

changed. Figure 18.1 illustrates the concepts of signal-to-noise ratio, significance level, and power. The family of curves on the left are the probability density functions for the distribution of the sample mean $\bar{x}$ from a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. The curves on the right illustrate the significance level α, and operating characteristic β for the null and alternative hypotheses

$$H_0 : \mu = 0, \sigma^2 = 4$$
$$H_1 : \mu = 1, \sigma^2 = 4$$

for a test using the sample mean from a sample of size $n = 25$ from a normal distribution, with a critical point $C = 0.4$. The significance level is the area under the curve for H_0 to the right of the critical point, while the operating characteristic is the area under the curve for H_1 to the left of the critical point. Clearly, increasing the critical value C will decrease α and increase β, while increasing the sample size n will decrease both α and β.

Often it is wished to predict power as a function of sample size, which can sometimes be done if distributions $f_0(x)$ and $f_1(x)$ are assumed, necessary parameters are provided, the critical level is specified, and the test procedure is defined. Essentially, given an implicit expression in k unknowns, this option solves for one given the other $k - 1$, using iterative techniques. For instance, you might set α and β, then calculate the sample size n required, or you could input α and n and estimate the power. Note that 1-tail tests can sometimes be selected instead of 2-tail tests (e.g., by replacing $Z_{\alpha/2}$ by Z_α in the appropriate formula) and also be very careful to make the correct choice for supplying proportions, half-widths, absolute differences, theoretical parameters or sample estimates, etc. A word of warning is required on the subject of calculating n required for a given power. The values of n will usually prove to be very large, probably much larger than can be used. So, for pilot studies and typical probing investigations, the sample sizes should be chosen according to cost, time, availability of materials, past experience, and so on. Sample size calculations are only called for when Type II errors may have serious consequences, as in clinical trials, so that large samples are justified. Of course, the temptation to choose 1-tail instead of 2-tail tests, or use variance estimates that are too small, in order to decrease the n values should be avoided.

18.2.1 Power calculations for 1 binomial sample

The calculations are based on the binomial test (page 152), the binomial distribution (page 417), and the normal approximation to it for large samples and p not close to 0 or 1, using the normal distribution (page 420). If the theoretical binomial parameters p_0 and $q_0 = 1 - p_0$ are not too close to 0 or 1 and it is wished to estimate this with an error of at most δ, then the sample size required is

$$n = \frac{Z^2_{\alpha/2} p_0 q_0}{\delta^2},$$
$$\text{where } P(Z > Z_{\alpha/2}) = \alpha/2,$$
$$\text{or } \Phi(Z_{\alpha/2}) = 1 - \alpha/2,$$

which, for many purposes, can be approximated by $n \approx 1/\delta^2$. The power in a binomial or sign test can be approximated, again if the sample estimates p_1 and $q_1 = 1 - p_1$ are not too close to 0 or 1, by

$$1 - \beta = P\left(Z < \frac{p_1 - p_0}{\sqrt{p_0 q_0 / n}} - Z_{\alpha/2}\sqrt{\frac{p_1 q_1}{p_0 q_0}}\right) + P\left(Z > \frac{p_1 - p_0}{\sqrt{p_0 q_0 / n}} + Z_{\alpha/2}\sqrt{\frac{p_1 q_1}{p_0 q_0}}\right).$$

18.2.2 Power calculations for 2 binomial samples

For two sample proportions p_1 and p_2 that are similar and not too close to 0 or 1, the sample size n and power $1 - \beta$ associated with a binomial test for $H_0 : p_{01} = p_{02}$ can be estimated using one of numerous methods based upon normal approximations. For example

$$n = \frac{(p_1 q_1 + p_2 q_2)(Z_{\alpha/2} + Z_\beta)^2}{(p_1 - p_2)^2},$$
$$Z_\beta = \sqrt{\frac{n(p_1 - p_2)^2}{p_1 q_1 + p_2 q_2}} - Z_{\alpha/2},$$
$$\beta = P(Z \geq Z_\beta),$$
$$1 - \beta = \Phi(Z_\beta).$$

Power for the Fisher exact test (page 143) with sample size n used to estimate both p_1 and p_2, as for the binomial test, can be calculated using

$$1 - \beta = 1 - \sum_{r=0}^{2n} \sum_{C_r} \binom{n}{x}\binom{n}{r-x},$$
$$\text{where } r = \text{ total successes,}$$
$$x = \text{ number of successes in the group,}$$
$$\text{and } C_r = \text{ the critical region.}$$

This can be inverted by SIMFIT to estimate n, but unfortunately the sample sizes required may be too large to implement by the normal procedure of enumerating probabilities for all 2 by 2 contingency tables with consistent marginals.

18.2.3 Power calculations for 1 normal sample

The calculations are based upon the confidence limit formula for the population mean μ from a sample of size n, using the sample mean $\bar{x}$, sample variance s^2 and the t distribution (page 422), as follows

$$P\left(\bar{x} - t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}\right) = 1 - \alpha,$$
$$\text{where } \bar{x} = \sum_{i=1}^{n} x_i / n,$$
$$s^2 = \sum_{i=1}^{n} (x_i - \bar{x})^2 / (n - 1),$$
$$P(t \leq t_{\alpha/2,\nu}) = 1 - \alpha/2,$$
$$\text{and } \nu = n - 1.$$

You input the sample variance, which should be calculated using a sample size comparable to those predicted above. Power calculations can be done using the half width $h = t_{\alpha/2,n-1}s/\sqrt{n}$, or using the absolute difference δ between the population mean and the null hypothesis mean as argument. The following options are available:

- To calculate the sample size necessary to estimate the true mean within a half width h

$$n = \frac{s^2 t^2_{\alpha/2,n-1}}{h^2};$$

- To calculate the sample size necessary for an absolute difference δ

$$n = \frac{s^2}{\delta^2}(t_{\alpha/2,n-1} + t_{\beta,n-1})^2; \text{ or}$$

- To estimate the power

$$t_{\beta,n-1} = \frac{\delta}{\sqrt{s^2/n}} - t_{\alpha/2,n-1}.$$

It should be noted that the sample size occurs in the degrees of freedom for the t distribution, necessitating an iterative solution to estimate n.

18.2.4 Power calculations for 2 normal samples

These calculations are based upon the same type of t test approach (page 136) as just described for 1 normal sample, except that the pooled variance s_p^2 should be input as the estimate for the common variance σ^2, i.e.,

$$s_p^2 = \frac{\sum_{i=1}^{n_x}(x_i - \bar{x})^2 + \sum_{j=1}^{n_y}(y_j - \bar{y})^2}{n_x + n_y - 2}$$

where X has sample size n_x and Y has sample size n_y. The following options are available:

- To calculate the sample size necessary to estimate the difference between the two population means within a half width h

$$n = \frac{2s_p^2 t^2_{\alpha/2,2n-2}}{h^2};$$

- To calculate the sample size necessary to detect an absolute difference δ between population means

$$n = \frac{2s_p^2}{\delta^2}(t_{\alpha/2,2n-2} + t_{\beta,2n-2})^2; \text{ or}$$

- To estimate the power

$$t_{\beta,2n-2} = \frac{\delta}{\sqrt{2s_p^2/n}} - t_{\alpha/2,2n-2}.$$

The t test has maximum power when $n_x = n_y$ but, if the two sample sizes are unequal, calculations based on the the harmonic mean n_h should be used, i.e.,

$$n_h = \frac{2n_x n_y}{n_x + n_y},$$

$$\text{so that } n_y = \frac{n_h n_x}{2n_x - n_h}.$$

18.2.5 Power calculations for k normal samples

The calculations are based on the 1-way analysis of variance technique (page 161). Note that the SIMFIT power as a function of sample size procedure also allows you to plot power as a function of sample size (page 282), which is particularly useful with ANOVA designs where the number of columns k can be of interest, in addition to the number per sample n. The power calculation involves the F and non-central F distributions (page 423) and you calculate the required n values by using graphical estimation to obtain starting estimates for the iteration. If you choose a n value that is sufficient to make the power as a function on n plot cross the critical power, the program then calculates the power for sample sizes adjacent to the intersection, which is of use when studying k and n for ANOVA.

18.2.5.1 Plotting power as a function of sample size

It is important in the design of experiments to be able to estimate the sample size needed to detect a significant effect. For such calculations you must specify all the parameters of interest except one, then calculate the unknown parameter using numerical techniques. For example, the problem of deciding whether one or more samples differ significantly is a problem in the Analysis of Variance, as long as the samples are all normally distributed and with the same variance. You specify the known variance, σ^2, the minimum detectable difference between means, Δ, the number of groups, k, the significance level, α, and the sample size per group, n. Then, using nonlinear equations involving the F and noncentral F distributions, the power, $100(1-\beta)$ can be calculated. It can be very confusing trying to understand the relationship between all of these parameters so, in order to obtain an impression of how these factors alter the power, a graphical technique is very useful, as in figure 18.2.

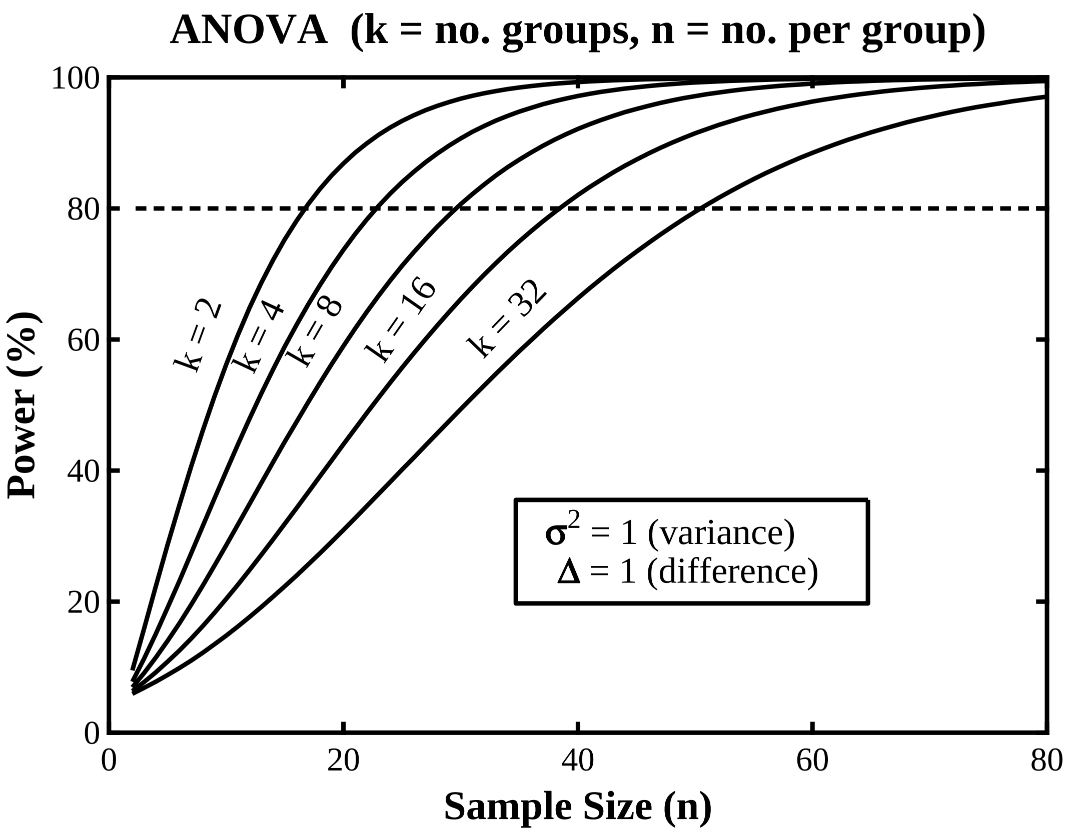


Figure 18.2: Power as a function of sample size

simstat was used to create this graph. The variance, significance level, minimum detectable difference and number of groups were fixed, then power was plotted as a function of sample size. The ASCII text coordinate files from several such plots were collected together into a library file to compose the joint plot using **simplot**. Note that, if a power plot reaches the current power level of interest, the critical power level is plotted (80% in the above plot) and the n values either side of the intersection point are displayed.

18.2.6 Power calculations for 1 and 2 variances

The calculations depend on the fact that, for a sample of size n from a normal distribution with true variance σ_0^2, the function χ^2 defined as

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

is distributed as a chi-square variable (page 423) with $n-1$ degrees of freedom. Also, given variance estimates s_x^2 and s_y^2 obtained with sample sizes n_x and n_y from the same normal distribution, the variance ratio F (page 136) defined as

$$F = \max\left(\frac{s_x^2}{s_y^2}, \frac{s_y^2}{s_x^2}\right)$$

is distributed as an F variable (page 423) with either n_x, n_y or n_y, n_x degrees of freedom. If possible n_x should equal n_y, of course. The 1-tailed options available are:

- ❑ $H_0 : \sigma^2 \le \sigma_0^2$ against $H_1 : \sigma^2 > \sigma_0^2$

$$1 - \beta = P(\chi^2 \ge \chi^2_{\alpha,n-1}\sigma_0^2/s^2);$$

- ❑ $H_0 : \sigma^2 \ge \sigma_0^2$ against $H_1 : \sigma^2 < \sigma_0^2$

$$1 - \beta = P(\chi^2 \le \chi^2_{1-\alpha,n-1}\sigma_0^2/s^2);\ \text{or}$$

- ❑ Rearranging the samples, if necessary, so that $s_x^2 > s_y^2$ then
  $H_0 : \sigma_x^2 = \sigma_y^2$ against $H_1 : \sigma_x^2 \ne \sigma_y^2$

$$Z_\beta = \sqrt{\frac{2m(n_y-2)}{m+1}}\log\left(\frac{s_x^2}{s_y^2}\right) - Z_\alpha$$
$$\text{where } m = \frac{n_x-1}{n_y-1}.$$

18.2.7 Power calculations for 1 and 2 correlations

The correlation coefficient r (page 190) calculated from a normally distributed sample of size n has a standard error

$$s_r = \sqrt{\frac{1-r^2}{n-2}}$$

and is an estimator of the population correlation ρ. A test for zero correlation, i.e., $H_0 : \rho = 0$, can be based on the statistics

$$t = \frac{r}{s_r},$$
$$\text{or } F = \frac{1+|r|}{1-|r|},$$

where t has a t distribution with $n-2$ degrees of freedom, and F has an F distribution with $n-2$ and $n-2$ degrees of freedom. The Fisher z transform and standard error s_z, defined as

$$z = \tanh^{-1} r,$$
$$= \frac{1}{2}\log\left(\frac{1+r}{1-r}\right),$$
$$s_z = \sqrt{\frac{1}{n-3}},$$

are also used to test $H_0 : \rho = \rho_0$, by calculating the unit normal deviate

$$Z = \frac{z - \zeta_0}{s_z}$$

where $\zeta_0 = \tanh^{-1} \rho_0$. The power is calculated using the critical value

$$r_c = \sqrt{\frac{t^2_{\alpha/2,n-2}}{t^2_{\alpha/2,n-2} + n - 2}}$$

which leads to the transform $z_c = \tanh^{-1} r_c$ and

$$Z_\beta = (z - z_c)\sqrt{n-3}$$

then the sample size required to reject $H_0 : \rho = 0$, when actually ρ is nonzero, can be calculated using

$$n = \left(\frac{Z_\beta + Z_{\alpha/2}}{\zeta_0}\right)^2 + 3.$$

For two samples, X of size n_x and Y of size n_y, where it is desired to test $H_0 : \rho_x = \rho_y$, the appropriate Z statistic is

$$Z = \frac{z_x - z_y}{s_{xy}}$$

$$\text{where } s_{xy} = \sqrt{\frac{1}{n_x - 3} + \frac{1}{n_y - 3}}$$

and the power and sample size are calculated from

$$Z_\beta = \frac{|z_x - z_y|}{s_{xy}} - Z_{\alpha/2},$$

$$\text{and } n = 2\left(\frac{Z_{\alpha/2} + Z_\beta}{z_x - z_y}\right)^2 + 3.$$

18.2.8 Power calculations for a chi-square test

The calculations are based on the chi-square test (page 143) for either a contingency table, or sets of observed and expected frequencies. However, irrespective of whether the test is to be performed on a contingency table or on samples of observed and expected frequencies, the null hypotheses can be stated in terms of k probabilities as
H_0 : the probabilities are $p_0(i)$, for $i = 1, 2, \ldots, k$,
H_1 : the probabilities are $p_1(i)$, for $i = 1, 2, \ldots, k$.
The power can then be estimated using the non-central chi-square distribution with non-centrality parameter λ and ν degrees of freedom given by

$$\begin{aligned}\lambda &= nQ,\\ \text{where } Q &= \sum_{i=1}^{k} \frac{(p_0(i) - p_1(i))^2}{p_0(i)},\\ n &= \text{ total sample size,}\\ \text{and } \nu &= k - 1 - \text{ no. of parameters estimated.}\end{aligned}$$

You can either input the Q values directly, or read in vectors of observed and expected frequencies. If you do input frequencies $f_i \geq 0$ they will be transformed internally into probabilities, i.e., the frequencies only have to be positive integers as they are normalized to sum unity using

$$p_i = f_i / \sum_{i=1}^{k} f_i.$$

In the case of contingency table data with r rows and c columns, the probabilities are calculated from the marginals $p_{ij} = p(i)p(j)$ in the usual way, so you must input $k = rc$, and the number of parameters estimated as $r + c - 2$, so that $\nu = (r-1)(c-1)$.

18.3 Parameter confidence limits

You choose the distribution required and the significance level of interest, then input the estimates and sample sizes required. Note that the confidence intervals may be asymmetric for those distributions (Poisson, binomial) where exact methods are used, not calculations based on the normal approximation.

18.3.1 Confidence limits for a Poisson parameter

Given a sample $x_1, x_2, \dots, x_n$ of n non-negative integers from a Poisson distribution with parameter λ (page 419), the parameter estimate $\hat{\lambda}$, i.e., the sample mean, and confidence limits λ_1, λ_2 are calculated as follows

$$\begin{aligned} K &= \sum_{i=1}^{n} x_i, \\ \hat{\lambda} &= K/n, \\ \lambda_1 &= \frac{1}{2n}\chi^2_{2K,\alpha/2}, \\ \lambda_2 &= \frac{1}{2n}\chi^2_{2K+2,1-\alpha/2}, \\ \text{so that } \exp(-n\lambda_1)\sum_{x=K}^{\infty}\frac{(n\lambda_1)^x}{x!} &= \frac{\alpha}{2}, \\ \exp(-n\lambda_2)\sum_{x=0}^{K}\frac{(n\lambda_2)^x}{x!} &= \frac{\alpha}{2}, \\ \text{and } P(\lambda_1 \le \lambda \le \lambda_2) &= 1-\alpha, \end{aligned}$$

using the lower tail critical points of the chi-square distribution (page 423). The following very approximate rule-of-thumb can be used to get a quick idea of the range of a Poisson mean λ given a single count x and exploiting the fact that the Poisson variance equals the mean

$$P(x - 2\sqrt{x} \le \lambda \le x + 2\sqrt{x}) \approx 0.95.$$

18.3.2 Confidence limits for a binomial parameter

For k successes in n trials, the binomial parameter estimate (page 417) $\hat{p}$ is k/n and three methods are used to calculate confidence limits p_1 and p_2 so that

$$\begin{aligned} \sum_{x=k}^{n}\binom{n}{x}p_1^x(1-p_1)^{n-x} &= \alpha/2, \\ \text{and } \sum_{x=0}^{k}\binom{n}{x}p_2^x(1-p_2)^{n-x} &= \alpha/2. \end{aligned}$$

- If $\max(k, n-k) < 10^6$, the lower tail probabilities of the beta distribution are used (page 424) as follows

$$\begin{aligned} p_1 &= \beta_{k,n-k+1,\alpha/2}, \\ \text{and } p_2 &= \beta_{k+1,n-k,1-\alpha/2}. \end{aligned}$$

- If $\max(k, n-k) \geq 10^6$ and $\min(k, n-k) \leq 1000$, the Poisson approximation (page 419) with $\lambda = np$ and the chi-square distribution (page 423) are used, leading to

$$\begin{aligned} p_1 &= \frac{1}{2n}\chi^2_{2k,\alpha/2}, \\ \text{and } p_2 &= \frac{1}{2n}\chi^2_{2k+2,1-\alpha/2}. \end{aligned}$$

- If $\max(k, n-k) > 10^6$ and $\min(k, n-k) > 1000$, the normal approximation (page 420) with mean np and variance $np(1-p)$ is used, along with the lower tail normal deviates $Z_{1-\alpha/2}$ and $Z_{\alpha/2}$, to obtain approximate confidence limits by solving

$$\begin{aligned} \frac{k - np_1}{\sqrt{np_1(1-p_1)}} &= Z_{1-\alpha/2}, \\ \text{and } \frac{k - np_2}{\sqrt{np_2(1-p_2)}} &= Z_{\alpha/2}. \end{aligned}$$

The following very approximate rule-of-thumb can be used to get a quick idea of the range of a binomial mean np given x and exploiting the fact that the binomial variance variance equals $np(1-p)$

$$P(x - 2\sqrt{x} \leq np \leq x + 2\sqrt{x}) \approx 0.95.$$

18.3.3 Confidence limits for a normal mean and variance

If the sample mean is $\bar{x}$, and the sample variance is s^2, with a sample of size n from a normal distribution (page 420) having mean μ and variance σ^2, the confidence limits are defined by

$$\begin{aligned} P(\bar{x} - t_{\alpha/2,n-1}s/\sqrt{n} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}s/\sqrt{n}) &= 1 - \alpha, \\ \text{and } P((n-1)s^2/\chi^2_{\alpha/2,n-1} \leq \sigma^2 \leq (n-1)s^2/\chi_{1-\alpha/2,n-1}) &= 1 - \alpha \end{aligned}$$

where the upper tail probabilities of the t (page 422) and chi-square (page 423) distribution are used.

18.3.4 Confidence limits for a correlation coefficient

If a Pearson product-moment correlation coefficient r (page 190) is calculated from two samples of size n that are jointly distributed as a bivariate normal distribution (page 421), the confidence limits for the population parameter ρ are given by

$$\begin{aligned} P\left(\frac{r - r_c}{1 - rr_c} \leq \rho \leq \frac{r + r_c}{1 + rr_c}\right) &= 1 - \alpha, \\ \text{where } r_c &= \sqrt{\frac{t^2_{\alpha/2,n-2}}{t^2_{\alpha/2,n-2} + n - 2}}. \end{aligned}$$

18.3.5 Confidence limits for trinomial parameters

If, in a trinomial distribution (page 418), the probability of category i is p_i for $i = 1, 2, 3$, then the probability P of observing n_i in category i in a sample of size $N = n_1 + n_2 + n_3$ from a homogeneous population is given by

$$P = \frac{N!}{n_1!n_2!n_3!}p_1^{n_1}p_2^{n_2}p_3^{n_3}$$

and the maximum likelihood estimates, of which only two are independent, are

$$\begin{aligned}\hat{p}_1 &= n_1/N,\\ \hat{p}_2 &= n_2/N,\\ \text{and } \hat{p}_3 &= 1 - \hat{p}_1 - \hat{p}_2.\end{aligned}$$

The bivariate estimator is approximately normally distributed, when N is large, so that

$$\begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{bmatrix} \sim MN_2\left(\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}, \begin{bmatrix} p_1(1-p_1)/N & -p_1p_2/N \\ -p_1p_2/N & p_2(1-p_2)/N \end{bmatrix}\right)$$

where MN_2 signifies the bivariate normal distribution (page 421). Consequently

$$((\hat{p}_1 - p_1), (\hat{p}_2 - p_2)) \begin{bmatrix} p_1(1-p_1)/N & -p_1p_2/N \\ -p_1p_2/N & p_2(1-p_2)/N \end{bmatrix}^{-1} \begin{pmatrix} \hat{p}_1 - p_1 \\ \hat{p}_2 - p_2 \end{pmatrix} \sim \chi^2_2$$

and hence, with probability 95%,

$$\frac{(\hat{p}_1 - p_1)^2}{p_1(1-p_1)} + \frac{(\hat{p}_2 - p_2)^2}{p_2(1-p_2)} + \frac{2(\hat{p}_1 - p_1)(\hat{p}_2 - p_2)}{(1-p_1)(1-p_2)} \le \frac{(1 - p_1 - p_2)}{N(1-p_1)(1-p_2)}\chi^2_{2;0.05}.$$

Such inequalities define regions in the (p_1, p_2) parameter space which can be examined for statistically significant differences between $p_{i(j)}$ in samples from populations subjected to treatment j. Where regions are clearly disjoint, parameters have been significantly affected by the treatments, as illustrated next.

18.3.5.1 Plotting trinomial parameter joint confidence regions

A useful rule of thumb to see if parameter estimates differ significantly is to check their approximate central 95% confidence regions. If the regions are disjoint it indicates that the parameters differ significantly and, in fact, parameters can differ significantly even with limited overlap. If two or more parameters are estimated, it is valuable to inspect the joint confidence regions defined by the estimated covariance matrix and appropriate chi-square critical value. Consider, for example, figure 18.3 generated by the contour plotting function of **binomial**. Data triples x, y, z can be any partitions, such as number of male, female or dead hatchlings from a batch of eggs where it is hoped to determine a shift from equi-probable sexes. The contours are defined by

$$((\hat{p}_x - p_x), (\hat{p}_y - p_y)) \begin{bmatrix} p_x(1-p_x)/N & -p_xp_y/N \\ -p_xp_y/N & p_y(1-p_y)/N \end{bmatrix}^{-1} \begin{pmatrix} \hat{p}_x - p_x \\ \hat{p}_y - p_y \end{pmatrix} = \chi^2_{2:0.05}$$

where $N = x + y + z$, $\hat{p}_x = x/N$ and $\hat{p}_y = y/N$ as discussed on page 286. When $N = 20$ the triples 9,9,2 and 7,11,2 cannot be distinguished, but when $N = 200$ the orbits are becoming elliptical and converging to asymptotic values. By the time $N = 600$ the triples 210,330,60 and 270,270,60 can be seen to differ significantly.

18.4 Robust analysis of one sample

Robust techniques are required when samples are contaminated by the presence of outliers, that is, observations that are not typical of the underlying distribution. Such observations can be caused by experimental accidents, such as pipetting enzyme aliquots twice into an assay instead of once, or by data recording mistakes, such as entering a value with a misplaced decimal point into a data table, but they can also occur because of additional stochastic components such as contaminated petri dishes or sample tubes. Proponents of robust techniques argue that extreme observations should always be down-weighted, as observations in the tails of distributions can seriously bias parameter estimates; detractors argue that it is scientifically dishonest to discard experimental observations, unless the experimentalists have independent grounds for suspecting particular observations. Table 18.1 illustrates the analysis of `robust.tf1`. These data are for `normal.tf1` but with five outliers, analyzed first by the exhaustive analysis of a vector procedure (page 122), then by the robust parameter estimates procedure. It should be noted that the Shapiro-Wilks test rejects normality and the robust estimators give much better parameter estimates in this case. If the sample vector is $x_1, x_2, \ldots, x_n$ the following calculations are done.

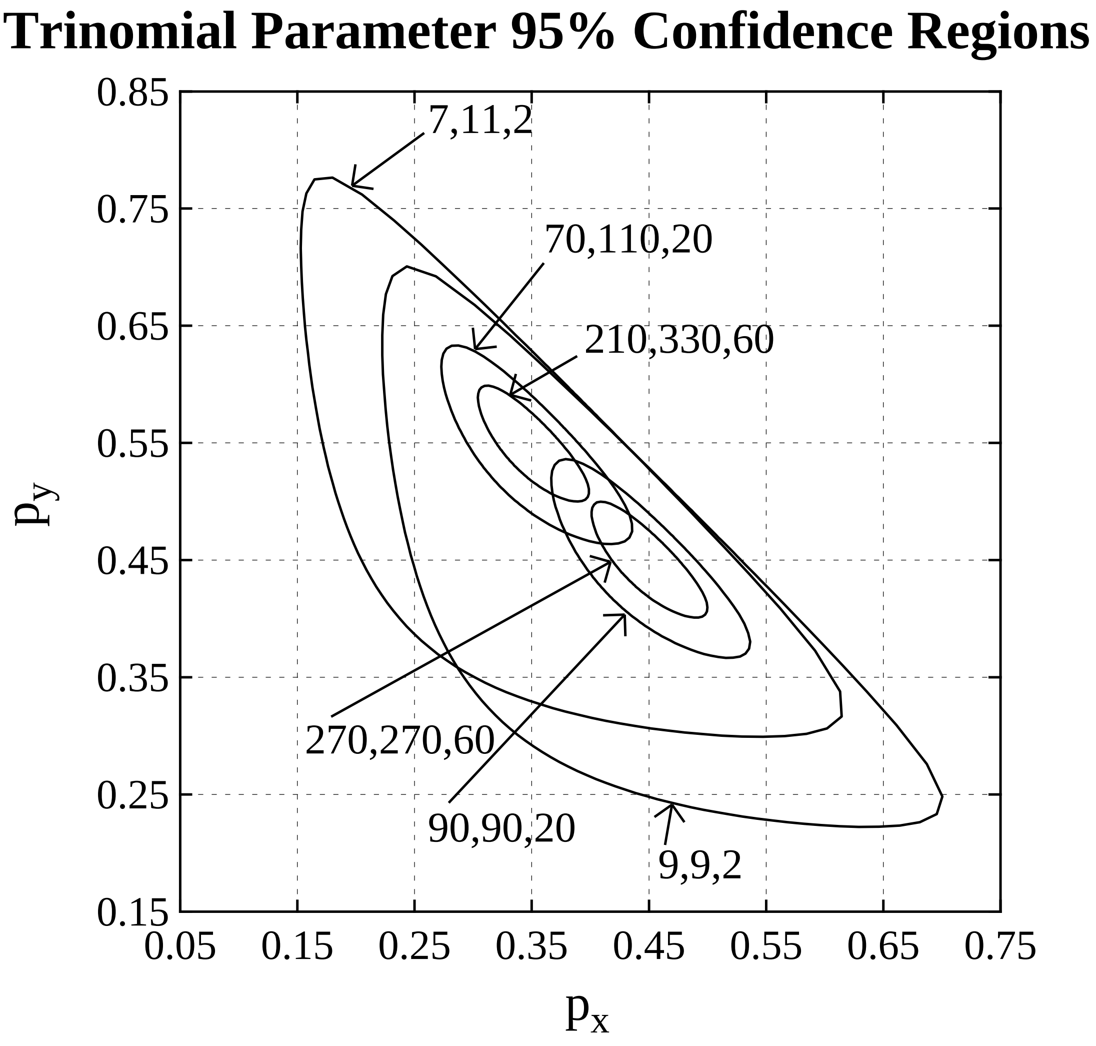


Figure 18.3: Trinomial parameter joint confidence contours

- Using the whole sample and the inverse normal function $\Phi^{-1}(.)$, the median M, median absolute deviation D and a robust estimate of the standard deviation S are calculated as

$$\begin{aligned} M &= \text{median}(x_i) \\ D &= \text{median}(|x_i - M|) \\ S &= D/\Phi^{-1}(0.75). \end{aligned}$$

- The percentage of the sample chosen by users to be eliminated from each of the tails is $100\alpha\%$, then the trimmed mean TM, and Winsorized mean WM, together with variance estimates VT and VW, are

```
Procedure 1: Exhaustive analysis of vector
Data: 50 N(0,1) random numbers with 5 outliers
Sample mean                = 0.5124
Sample standard deviation = 1.853: CV% = 361.736%
Shapiro-Wilks W statistic = 0.85
Significance level for W  = 0.0000     Reject normality at 1% sig.level

Procedure 2: Robust 1-sample analysis
Total sample size            = 50
Median value                 = 0.20189
Median absolute deviation    = 1.0311
Robust standard deviation    = 1.5288
Trimmed mean (TM)            = 0.22267
Variance estimate for TM     = 0.019178
Winsorized mean (WM)         = 0.2326
Variance estimate for WM     = 0.019176
Number of discarded values   = 10
Number of included values    = 40
Percentage of sample used    = 80.00% (for TM and WM)
Hodges-Lehmann estimate (HL) = 0.25856
```

Table 18.1: Robust analysis of one sample

calculated as follows, using $k = [\alpha n]$ as the integer part of αn.

$$TM = \frac{1}{n-2k}\sum_{i=k+1}^{n-k} x_i$$

$$WM = \frac{1}{n}\left\{\sum_{i=k+1}^{n-k} x_i + kx_{k+1} + kx_{n-k}\right\}$$

$$VT = \frac{1}{n^2}\left\{\sum_{i=k+1}^{n-k}(x_i - TM)^2 + k(x_{k+1} - TM)^2 + k(x_{n-k} - TM)^2\right\}$$

$$VW = \frac{1}{n^2}\left\{\sum_{i=k+1}^{n-k}(x_i - WM)^2 + k(x_{k+1} - WM)^2 + k(x_{n-k} - WM)^2\right\}.$$

❑ If the assumed sample density is symmetrical, the Hodges-Lehmann location estimator HL can be used to estimate the center of symmetry. This is

$$HL = \text{ median}\left\{\frac{x_i + x_j}{2}, 1 \le i \le j \le n\right\},$$

and it is calculated along with 95% confidence limit. This would be useful if the sample was a vector of differences between two samples X and Y for a Wilcoxon signed rank test (page 142) that X is distributed $F(x)$ and Y is distributed $F(x - \theta)$.

18.5 Robust analysis of two samples

Table 18.2 illustrates the analysis of `ttest.tf4` and `ttest.tf5` used earlier for a Mann-Whitney U test (page 140). The procedure is based on the assumption that X of size n_x is distributed as $F(x)$ and Y of size n_y as $F(x - \theta)$, so an estimate $\hat{\theta}$ for the difference in location is calculated as

$$\hat{\theta} = \text{ median}(y_j - x_i, i = 1, 2, \ldots, n_x, j = 1, 2, \ldots, n_y).$$

```
X-sample size                  =  12
Y-sample size                  =  7
Difference in location         = -18.501
Lower confidence limit         = -40.0
Upper confidence limit         =  2.997
Percentage confidence limit    =  95.30%
Lower Mann-whitney U-value     =  19.0
Upper Mann-Whitney U-value     =  66.0
```

Table 18.2: Robust analysis of two samples

$100\alpha\%$ confidence limits U_L and U_H are then estimated by inverting the Mann-Whitney U statistic so that

$$\begin{aligned} P(U \le U_L) &\le \alpha/2 \\ P(U \le U_L + 1) &> \alpha/2 \\ P(U \ge U_H) &\le \alpha/2 \\ P(U \ge U_H - 1) &> \alpha/2. \end{aligned}$$

18.6 Indices of diversity

It is often required to estimate the entropy or degree of randomness in the distribution of observations into categories. For instance, in ecology several indices of diversity are used, as illustrated in table 18.3 for two

```
Data: 5,5,5,5
Number of groups        = 4
Total sample size       = 20
Pielou J-prime evenness = 1.0    [complement = 0.0]
Brillouin J evenness    = 1.0    [complement = 0.0]
Shannon H-prime         = 0.6021(log10)  1.386(ln)    2.0(log2)
Brillouin H             = 0.5035(log10)  1.159(ln)  1.672(log2)
Simpson lambda          = 0.25   [complement = 0.75  ]
Simpson lambda-prime    = 0.2105 [complement = 0.7895]

Data: 1,1,1,17
Number of groups        = 4
Total sample size       = 20
Pielou J-prime evenness = 0.4238 [complement = 0.5762]
Brillouin J evenness    = 0.3809 [complement = 0.6191]
Shannon H-prime         = 0.2551(log10)  0.5875(ln)  0.8476(log2)
Brillouin H             = 0.1918(log10)  0.4415(ln)   0.637(log2)
Simpson lambda          = 0.3    [complement = 0.27  ]
Simpson lambda-prime    = 0.7158 [complement = 0.2842]
```

Table 18.3: Indices of diversity

extreme cases. Given positive integer frequencies $f_i > 0$ in $k > 1$ groups with n observations in total, then proportions $p_i = f_i/n$ can be defined, leading to the Shannon H', Brillouin H, and Simpson λ and λ' indices, and the evennness parameters J and J' defined as follows.

$$\begin{aligned}
\text{Shannon diversity } H' &= -\sum_{i=1}^{k} p_i \log p_i \\
&= [n \log n - \sum_{i=1}^{k} f_i \log f_i]/n \\
\text{Pielou evenness } J' &= H'/\log k \\
\text{Brilloin diversity } H &= [\log n! - \log \prod_{i=1}^{k} f_i!]/n \\
\text{Brilloin evenness } J &= nH/[\log n! - (k-d)\log c! - d\log(c+1)!] \\
\text{Simpson lambda } \lambda &= \sum_{i=1}^{k} p_i^2 \\
\text{Simpson lambda prime } \lambda' &= \sum_{i=1}^{k} f_i(f_i - 1)/[n(n-1)]
\end{aligned}$$

where $c = [n/k]$ and $d = n - ck$. Note that H and H' are given using logarithms to bases ten, e, and two, while the forms J and J' have been normalized by dividing by the corresponding maximum diversity and so are independent of the base. The complements $1 - J$, $1 - J'$, $1 - \lambda$, and $1 - \lambda'$ are also tabulated within the square brackets. In table 18.3 we see that evenness is maximized when all categories are equally occupied, so that $f_i = 1/k$ and $H' = \log k$, and is minimized when one category dominates.

18.7 Standard and non-central distributions

SimFIT uses discrete (page 417) and continuous (page 419) distributions for modelling and hypothesis tests, and the idea behind this procedure is to provide the option to plot and obtain percentage points for the standard statistical distributions to replace table look up. However, you can also obtain values for the distribution functions, given the arguments, for the non-central t, beta, chi-square or F distributions (page 425), or you can plot graphs, which are very useful for advanced studies in the calculation of power as a function of sample size. Figure 18.4 illustrates the chi-square distribution with 10 degrees of freedom for noncentrality parameter λ at values of 0, 5, 10, 15, and 20.

18.8 Generating random numbers, permutations and Latin squares

In the design of experiments it is frequently necessary to generate sequences of pseudo-random numbers, or random permutations. For instance, assigning patients randomly to groups requires that a consecutive list of integers, names or letters be scrambled, while any ANOVA based on Latin squares should employ randomly generated Latin squares. SimFIT will generate sequences of random numbers and permutations for these purposes. For example, all possible 4×4 Latin squares can be generated by random permutation of the rows and columns of the four basic designs shown in Table 18.4. Higher order designs that are sufficiently random for most purposes can be generated by random permutations of the rows and columns of a default $n \times n$ matrix

A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
B	A	D	C	B	C	D	A	B	D	A	C	B	A	D	C
C	D	B	A	C	D	A	B	C	A	D	B	C	D	A	B
D	C	A	B	D	A	B	C	D	C	B	A	D	C	B	A

Table 18.4: Latin squares: 4 by 4 random designs

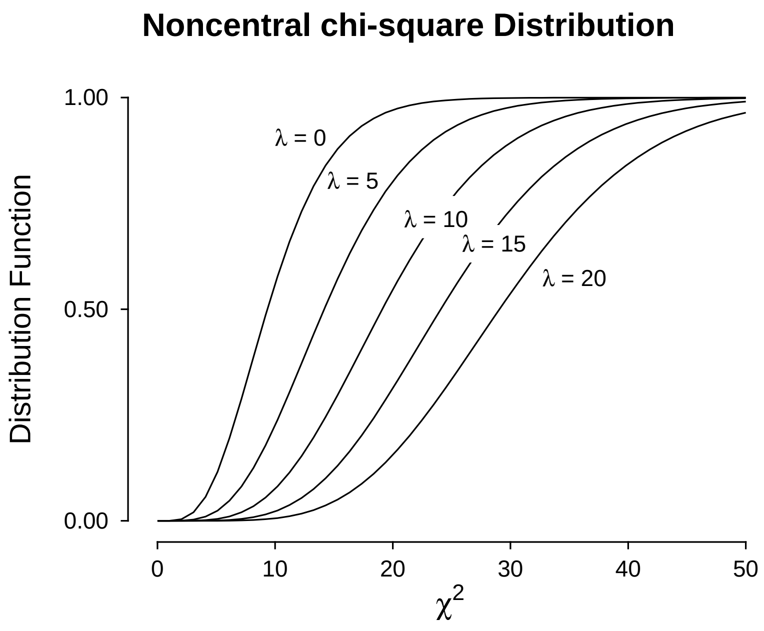


Figure 18.4: Noncentral chi-square distribution

A	B	C	D	E	F	G
B	C	D	E	F	G	A
C	D	E	F	G	A	B
D	E	F	G	A	B	C
E	F	G	A	B	C	D
F	G	A	B	C	D	E
G	A	B	C	D	E	F

Table 18.5: Latin squares: higher order random designs

with sequentially shifted entries of the type shown in Table 18.5, for a possible 7×7 starting matrix, although this will not generate all possible examples for $n > 4$. Note that program **rannum** provides many more options for generating random numbers.

18.8.1 Plotting random walks

Many experimentalists record movements of bacteria or individual cells in an attempt to quantify the effects of attractants, etc. so it is useful to compare experimental data with simulated data before conclusions about persistence or altered motility are reached. Program **rannum** can generate such random walks starting from arbitrary initial coordinates and using specified distributions. The probability density functions for the axes can be chosen independently and different techniques can be used to visualize the walk depending on the number of dimensions. Figure 18.5 shows a classical unrestricted walk on an integer grid, that is, the steps can be $+1$ with probability p and -1 with probability $q = 1 - p$. It also shows 2- and 3-dimensional walks for standard normal distributions.

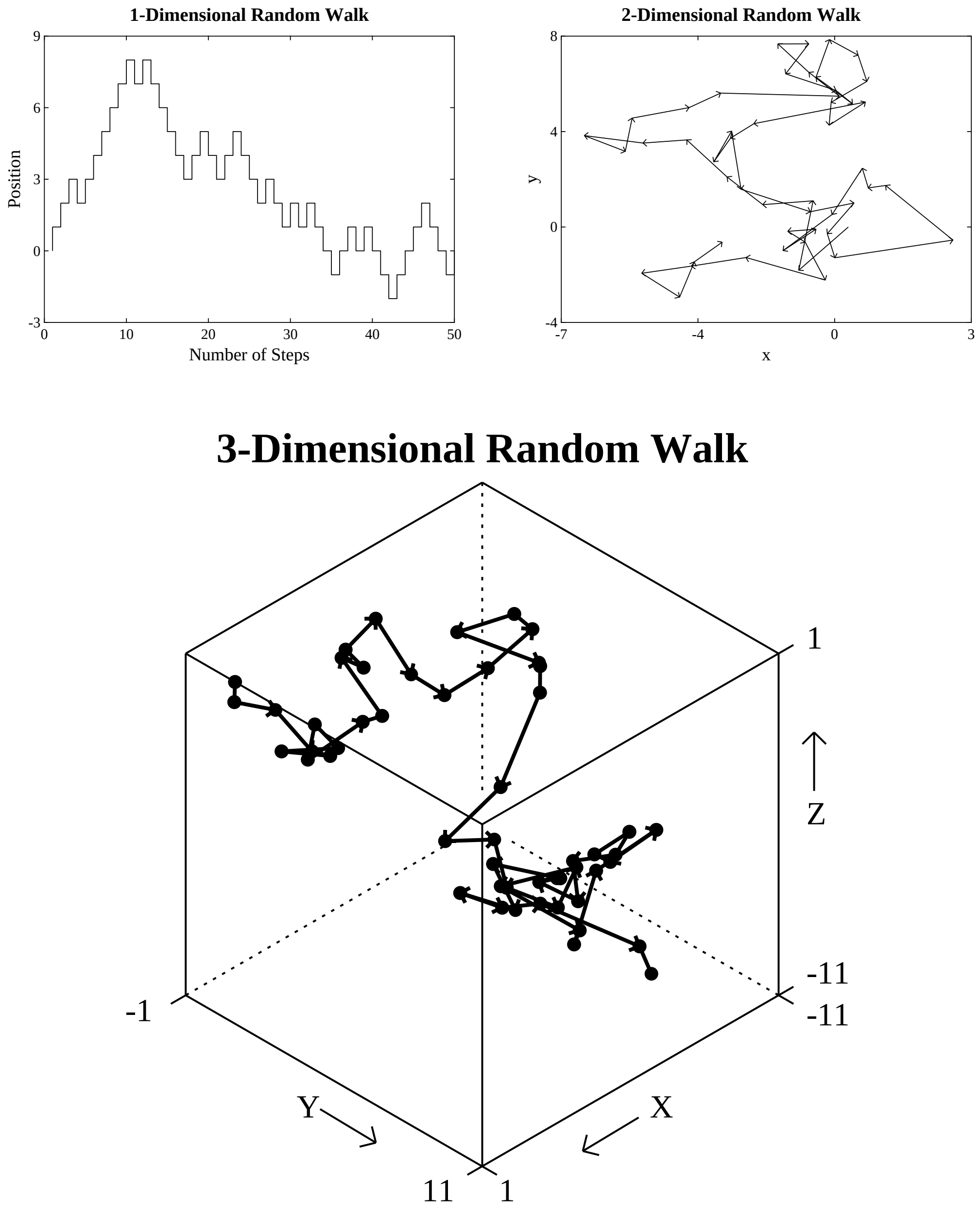


Figure 18.5: Random walks

18.9 Kernel density estimation

This technique is used to create a numerical approximation to the density function given a random sample of observations for which there is no known density. Figure 18.6 illustrates the results when this was done with data simulated from a normal distribution with $\mu = 0$ and $\sigma^2 = 1$, using 5 bins for the histogram in the top row of figures, but using 10 bins for the histogram in the bottom row. In this example changing the number of bins k alters the density estimate since, given a sample of n observations $x_1, x_2, \ldots, x_n$ with $A \le x_i \le B$,

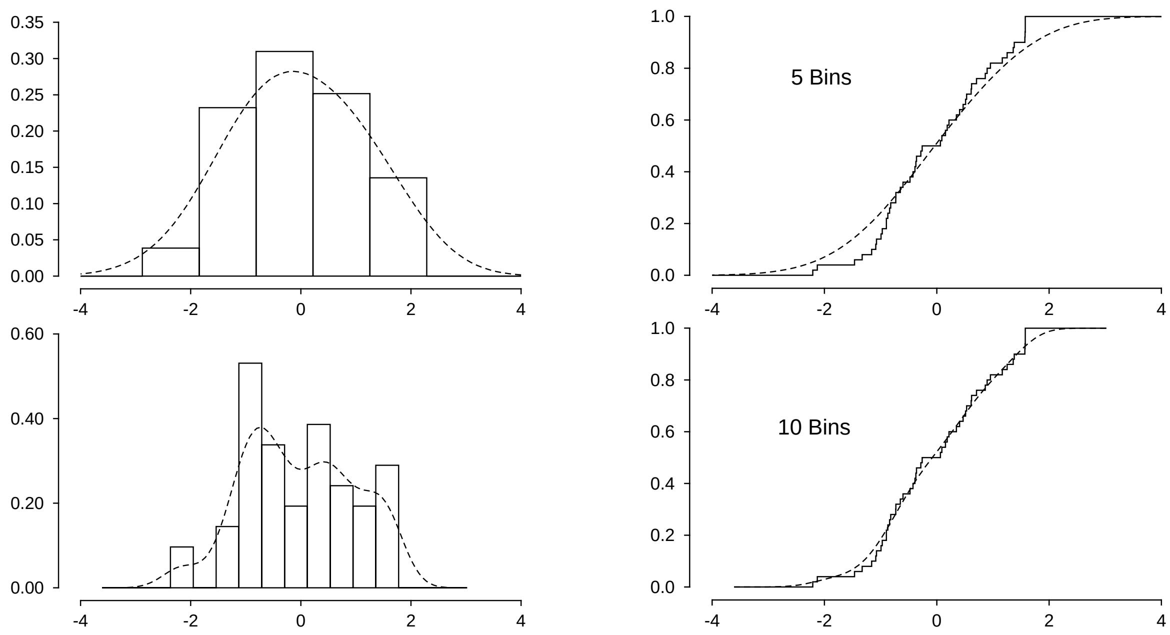


Figure 18.6: Kernel density estimation

the Gaussian kernel density estimate $\hat{f}(x)$ is defined as

$$\begin{aligned}\hat{f}(x) &= \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)\\ \text{where } K(t) &= \frac{1}{\sqrt{2\pi}}\exp(-t^2/2)\\ \text{and in this case } h &= (B-A)/(k-2).\end{aligned}$$

Clearly, a window width h similar to the bin width, as in the top row, can generate an unrealistic over-smoothed density estimate, while using narrower many bins, as in the second row, can lead to over-fitting.

Details are as follows.

❏ The calculation involves four steps.

1. From the n data points x_i choose a lower limit a, an upper limit b, and m equally spaced points t_i where
$$a = A - 3h \le t_i \le B + 3h = b,$$
and m is power of 2. The value of m can be altered interactively from the default value of 128 if necessary for better representation of multi-modal profiles. Data are discretized by binning the x_i at points t_i to generate weights ξ_l.
2. Compute FFT of the weights, ξ_l to give Y_l.
3. Compute $\xi_l = Y_l \exp{(h^2 s_l^2/2)}$ where $s_l = 2\pi l/(b-a)$
4. Find the inverse FFT of ξ_l to give $\hat{f}(x)$.

❏ The histograms shown on the left use k bins to contain the sample, and the height of each bin is the fraction of sample values in the bin. The value of k can be changed interactively, and the dotted curves are the density estimates for the m values of t. The program generates additional empty bins for the FFT outside the range set by the data to allow for wrap round. However, the total area under the histogram is one, and the density estimate integrates to one between $-\infty$ and ∞.

❏ In addition to the definition of the smoothing parameter h depending on the number of bins chosen for display in figure 18.6 the default setting, which is

$$h = 1.06\hat{\sigma}n^{-1/5},$$

uses the sample standard deviation and sample size, as recommended for a normal distribution. Users can also set arbitrary smoothing parameters and, with these two options, the histograms plotted simply illustrate the fit of the kernel density estimate to the data and do not alter the smoothing parameter h.

❏ The sample cumulative distributions shown on the right have a vertical step of $1/n$ at each sample value, and so they increase stepwise from zero to one. The density estimates are integrated numerically to generate the theoretical cdf functions, which are shown as dashed curves. They will attain an asymptote of one if the number of points m is sufficiently large to allow accurate integration, say ≥ 100.

❏ The density estimates are unique given the data, h and m, but they will only be meaningful if the sample size is fairly large, say ≥ 50 and preferably much more. Further, the histogram bins will only be representative of the data if they have a reasonable content, say $n/k \geq 10$.

❏ The histogram, sample distribution, pdf estimate and cdf estimate can be saved to file by selecting the [Advanced] option then creating ASCII text coordinate files.

18.10 Fitting probability distributions

Often a sample x of n observations x_i is available and it is wished to display a graph just to summarize the sample, or with a best-fit distribution overlayed. Suppose the sample is in nondecreasing order, as in

$$x_1 \leq x_2 \leq x_3 \leq \cdots \leq x_n,$$

then there are two ways to do this.

❍ **A histogram**
Limits $A \leq x_1$ and $B \geq x_n$ are chosen and with the number of bins $k < n$, then the sample is used to generate frequencies for each of the bins. That is, the distance between A and B is divided into k equal intervals, and the number of observations in each interval gives k frequencies f_j, where

$$\sum_{j=1}^{k} f_j = n, \text{ and}$$
$$\text{Area} = n(B - A)/k.$$

Clearly the histogram shape will depend upon k, A, and B. As an extremely approximate idea of the reliability of the bins as a representation of a *pdf*, error bars equal to twice the square root of the frequencies can be added. To fit a known *pdf* to the frequencies, the limits A and B must include the whole sample, and sometimes the frequencies are divided by the area to normalize the area to 1 and avoid having to fit a *pdf* pre-multiplied by a scaling factor.

❍ **A cumulative distribution**
The sample is used to create a curve increasing by steps of $1/n$ at each observation. Such a curve is unique, which is better than a histogram as it does not require the selection of end points, and the function must starts at zero and rise to 1. However *cdf*s are usually simple sigmoid curves which do not display the properties of the underlying distribution as well as histograms.

For instance, Figure 18.7 shows a discrete probability distributions histogram plotted using vertical lines.
Also figure 18.8 illustrates that a good compromise to demonstrate goodness of fit is to plot the scaled *pdf* along with the best fit *cdf*, as with the beta distribution in this case.

18.11 Fitting a mixture of two normal distributions

Fitting the standard distributions to samples is very easy and is done automatically by SIMFIT when performing the Kolmogorov-Smirnov (page 132), Normal distribution (page 133), Poisson (page 134), and similar tests. However it is sometimes suspected that a sample of observations comes from a mixture of distributions. For instance a sample of observations of height from a population would be a mixture of two distributions: females and males. Now the problem of resolving a distribution into constituent distributions with no a priori knowledge of the separate distributions is extremely difficult and controversial, so the SIMFIT technique of fitting sample histograms and cumulative distributions is only one of the various methods that can be used. It does have the advantage of being applicable to any mixture of distributions provided the model equation can be defined. To illustrate this SIMFIT technique, the analysis of a mixture of two Gaussian distributions will be considered.

In the case of two Normal distributions the model for the *pdf* is

$$\frac{\alpha}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left\{\frac{x-\mu_1}{\sigma_1}\right\}^2\right)+\frac{1-\alpha}{\sigma_2\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left\{\frac{x-\mu_2}{\sigma_2}\right\}^2\right),$$

while that for the *cdf* is

$$\frac{\alpha}{\sigma_1\sqrt{2\pi}}\int_{-\infty}^{x}\exp\left(-\frac{1}{2}\left\{\frac{t-\mu_1}{\sigma_1}t\right\}^2\right)dt+\frac{1-\alpha}{\sigma_2\sqrt{2\pi}}\int_{-\infty}^{x}\exp\left(-\frac{1}{2}\left\{\frac{t-\mu_2}{\sigma_2}\right\}^2\right)dt.$$

These models, which are both in the SIMFIT library of statistical models, require the estimation of five parameters: the two means μ_1 and μ_2, the two standard deviations σ_1 and σ_2, and an extra parameter α. Obviously the model must be constrained so that σ_1 and σ_2 are positive, and the partitioning constant limits must be chosen so that $0 \le \alpha \le 1$.

Figure 18.9 shows the fit of these two models to the test files `normalpdf.tf3` and `normalcdf.tf3` which were derived by adding `normal.tf1` to `normal.tf2` using **editmt** to create `normal.tf3` with a mixture of 50 random numbers from $N(0,1)$ with 50 random numbers from $N(5,1)$. The file `normalpdf.tf3` was created

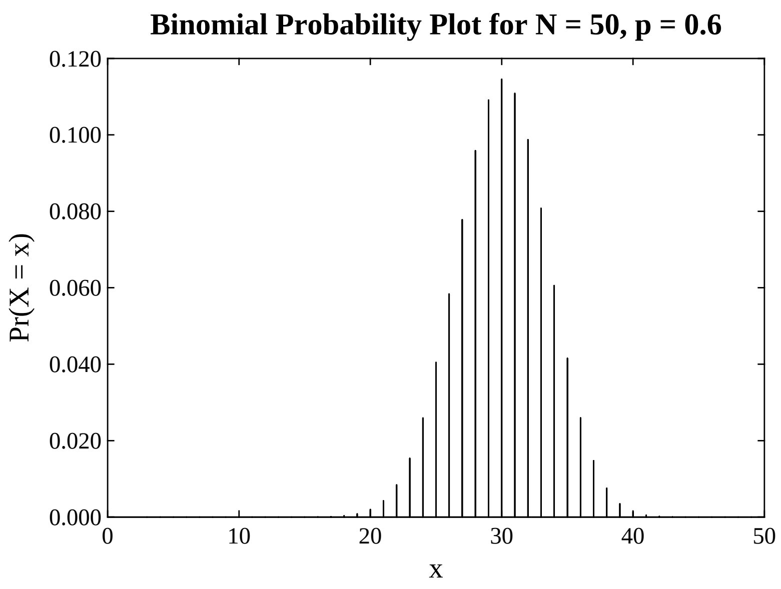


Figure 18.7: Binomial probability distributions

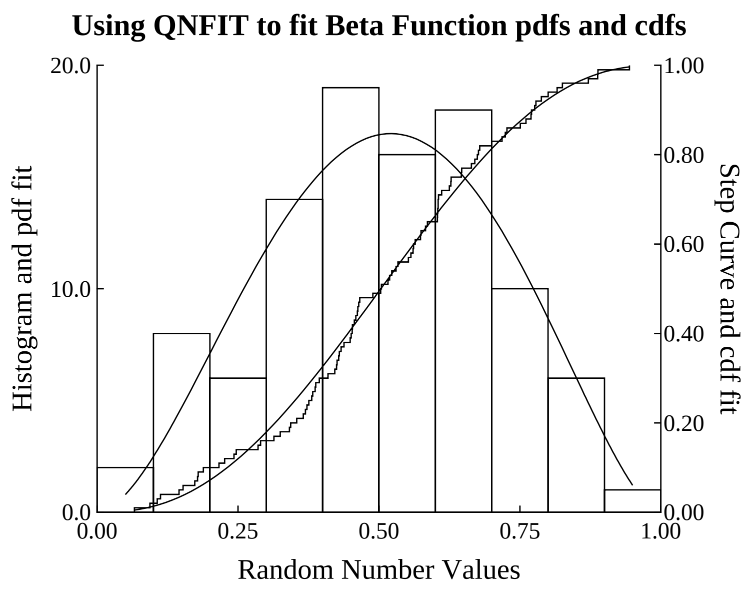


Figure 18.8: Beta probability distributions

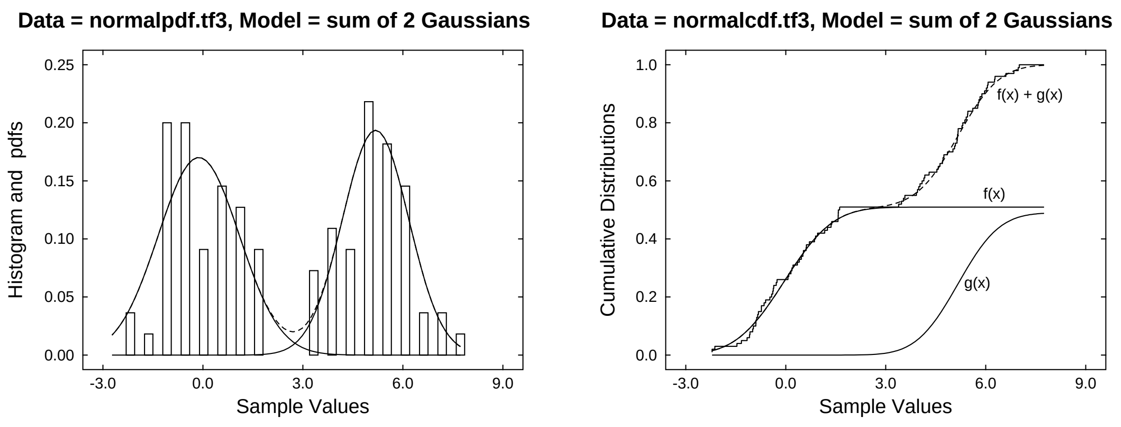
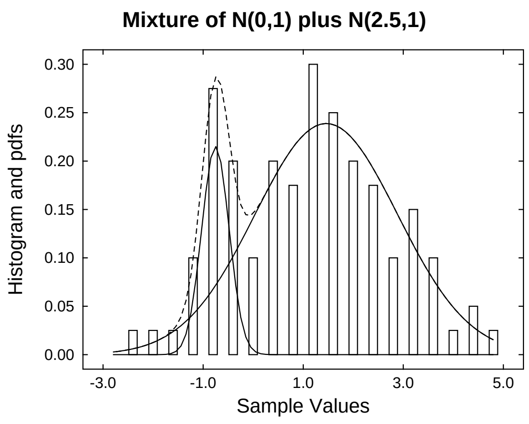


Figure 18.9: Fitting a mixture of two disjoint normal distributions

from `normal.tf3` using the exhaustive analysis of a vector technique (page 122) followed by requesting a *pdf* file normalized to area 1, while requesting a *cdf* file generated `normalcdf.tf3`. After curve fitting using **qnfit** (page 79) the option for graphical deconvolution (page 38) was selected to show the data, best-fit model and contributing sub-models. Outline bars were chosen for the *pdf* data, and a *cdf* type stair step curve was selected for the sample cumulative. This example was chosen to illustrate an extreme case where the two distributions are disjoint and, not surprisingly, the parameters were determined accurately. In fact, users would have probably split such a sample into two distinct data sets for univariate analysis. However, it is important to realize that such fitting is likely to be very disappointing if the distributions have an appreciable overlap.

Figure 18.10 shows the fit in a less favorable situation.

The data set consisted of a mixture of 50 random numbers from a normal distribution with $\mu = 0, \sigma = 1$ with 50 random numbers from a normal distribution with $\mu = 2.5, \sigma = 1$, and the analysis proceeded as with the



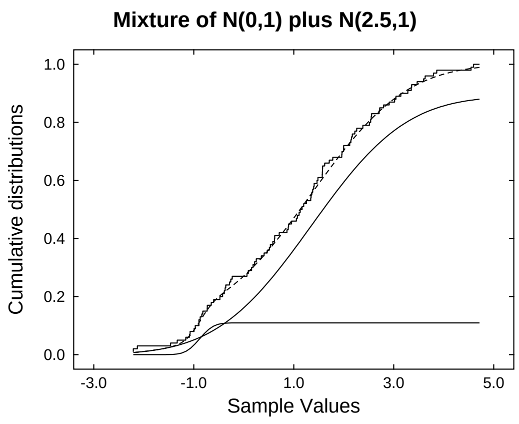


Figure 18.10: Fitting a mixture of two overlapping normal distributions

previous data. It is perfectly clear that the best-fit *pdf* and *cdf* shown as dotted curves in figure 18.10 give an excellent fit to the data, but the component distributions shown as solid curves bear no resemblance to the actual distributions generating the data set.

The parameters estimated by the fitting procedure can be identified by this scheme

Parameter	Symbol	Actual Value	*pdf* estimate	*cdf* estimate
p_1	μ_1	0.0	-0.75	-0.88
p_2	σ_1	1.0	0.27	0.22
p_3	α	0.5	0.15	0.11
p_4	μ_2	2.5	1.46	1.36
p_5	σ_2	1.0	1.42	1.49

while table 18.6 shows the SIMFIT summary of results table.

```
Best-fit pdf
Number Lower-Limit Upper-Limit    Value      Std. Err.   ..95% Conf. Lim..    p
   1          -2.0         2.0   -0.7.5315    0.05757    -0.8758 -0.6304  0.0000
   2          0.05         3.0     0.2732     0.05379     0.1585  0.3879  0.0001
   3           0.0         1.0     0.14731    0.3744      0.0675  0.2271  0.0013
   4           0.5         4.5     1.4612     0.135       1.173   1.749   0.0000
   5          0.05         3.0     1.4243     0.1232      1.162   1.687   0.0000

Best-fit cdf
Number Lower-Limit Upper-Limit    Value      Std. Err.   ..95% Conf. Lim..    p
   1          -2.0         2.0 -0.88103    0.02025      -0.9212 -0.8408  0.0000
   2          0.05         3.0  0.21874    0.04357       0.1322  0.3052  0.0000
   3           0.0         1.0  0.10937    0.01173      0.08608  0.1327  0.0000
   4           0.5         4.5   1.3566    0.02935        1.298   1.415  0.0000
   5         0.05          3.0   1.4883    0.02569        1.437   1.539  0.0000
```

Table 18.6: Fitting a mixture of two overlapping normal distributions

The conclusion is just commonsense and inescapable: it is only possible to obtain meaningful estimates for the parameters of mixed distributions when the sample sizes are very large and the distributions are well separated.

18.12 Fitting flow cytometry histograms

Sometimes there is a large sample of observations that are only available after being sorted into bins and, despite the fact that no standard distribution fits the data, it is nevertheless important to perform some sort of analysis. For instance, in flow cytometry it may be required to compare the expression of cell surface antigens in the same cell line before and after treatment, or to compare two related cell lines. This can be done using the SIMFIT program **csafit** as described next.

Suppose that there are n bins between limits A and B for both profiles, and the frequencies for cell line X are $\Phi_X(i)$ (normalized to area 1) while those of cell line Y are $\Phi_Y(i)$ (normalized to area 1) so that from the n bins we would have estimates for the distribution of X and Y. The density functions for X and Y could be unrelated of course, but a common situation would be where the gene expression Y would be that of X subjected to a possible shift α due to a relative increase in the amount of gene expression, plus a possible absolute increase β, say due to an extra copy of the gene as in trisomy. Then we would have the relationship

$$Y = \alpha X + \beta \text{ in distribution,}$$

for random variables X and Y leading to the following identity between density functions f_X and f_Y

$$f_Y(y) = \left(\frac{\gamma}{\alpha}\right) f_X\left(\frac{y-\beta}{\alpha}\right)$$

where the normalizing factor is

$$\frac{1}{\gamma} = \left(\frac{1}{\alpha}\right)\int_a^b f_X\left(\frac{u-\beta}{\alpha}\right) du.$$

Rearranging if necessary to assume $\alpha > 1$ then, to account for truncation, we would have the limits

$$\begin{aligned} a &= \alpha A + \beta \text{ if } \alpha A + \beta > A \\ &= A \text{ otherwise} \\ b &= \alpha B + \beta \text{ if } \alpha B + \beta < B \\ &= B \text{ otherwise.} \end{aligned}$$

Figure 18.11 illustrates the analysis of data in test file `csafit.tf3` while table 18.7 shows the best fit parameters estimated by **csafit**.

```
Best-fit parameters from CSAFIT

      alpha =  1.15   (Dimensionless)
       beta =  0.0518 (Internal coordinates)
    stretch = 14.95 %
translation =  5.18 %
```

Table 18.7: Fitting a flow cytometry histogram

The way that **csafit** works is that the two histograms are first read in and normalized to area 1, and a least squares smoothing cubic splines constrained to have area 1 is fitted to the Φ_i normalized X frequencies, and goodness of fit is performed. Then parameters α and β are estimated by nonlinear optimization to the Ψ_i normalized Y frequencies obtained by translation and stretching of the spline fitted to the X data, and goodness of fit is performed. The program provides options for selecting the spline nodes interactively or from the data file, for choosing to vary either α alone, β alone, or α and β together, and for choosing starting estimates

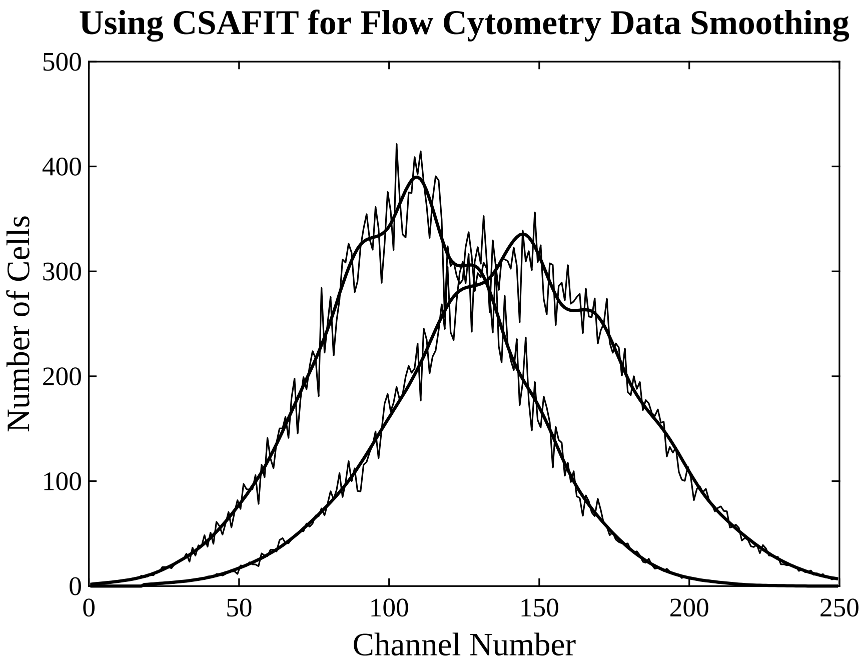


Figure 18.11: Flow cytometry

interactively should that be required. As **csafit** is an advanced program, there is a program called **makcsa** that should be used to simulate data with random error and fit using **csafit**.

This approach of attempting to explain the shift of one distribution from another by a combination of relative stretching together with an absolute translation can be used in many other situations other than flow cytometry but it remains to mention a complication. Often data are collected over a very large range of values so that bins are in a geometric rather than an arithmetic progression and this necessitates further consideration. If the transformed variables are now $\log X$ and $\log Y$, where the logarithms in flow cytometers would usually be to base 10, then the densities in transformed space would then be

$$\begin{aligned} f_{\log Y}(\log y) &= \left(\frac{\gamma 10^{\log y}}{\psi(y)}\right) f_{\log X}(\log \psi(y)) \\ \text{where } \psi(y) &= \frac{10^{\log y} - \beta}{\alpha}, \end{aligned}$$

and the normalizing integral will now be

$$\frac{1}{\gamma} = \int_{a^*}^{b^*} \left(\frac{10^{\log u}}{\psi(u)}\right) f_{\log X}(\log \psi(u))\, d(\log u)$$

with limits

$$\begin{aligned} a^* &= \log(\alpha A + \beta) \text{ if } \log(\alpha A + \beta > \log A \\ &= \log A \text{ otherwise} \\ b^* &= \log(\alpha B + \beta) \text{ if } \log(\alpha B + \beta) < \log B \\ &= \log B \text{ otherwise} \end{aligned}$$

where it is assumed that the bin limits are now $\log A$ and $\log B$.

Program **csafit** can deal with this situation but it has been found better to just vary α and fix $\beta = 0$ in the linear model when using logarithmically spaced bins, as this is equivalent is equivalent to fitting $\alpha = 1$ and β varied in the transformed space and changing the interpretation of parameter estimates since

$$\begin{aligned} \text{if } Y &= \alpha X \\ \log Y &= \log X + \delta \\ \text{where } \delta &= \log \alpha. \end{aligned}$$

That is, the model $\log Y = \log X + \delta$ would be fitted to the transformed histograms, but the estimate for the additive parameter δ would be interpreted in terms of α as a stretch in the natural coordinates from the point of view of mechanistic interpretation.

18.13 Optimal design for model discrimination

The SIMFIT program **eoqsol** provides algorithms designed to answer a frequently asked question.

> Given two rival models, then what is the best separation of data points to employ in order to maximize the power for the process of deciding on statistical grounds which model to accept ?

Consider fitting a deficient model $g_1(x, \Phi)$ to n data points generated by a correct model $g_2(x, \Theta)$, using weights $w(x) > 0$ and density function $f(x)$ to define the spacing of support points between limits $a \le x \le b$. Then this question can be explored by considering several procedures which require techniques for optimization, quadrature, and extracting the zeros of nonlinear equations.

❍ The average weighted model error squared $S_n(\Theta)$ can be defined as

$$\begin{aligned} S_n(\Theta) &= \min_{\Phi} \frac{1}{n} \sum_{i=1}^{n} w(x_i)[g_2(x_i, \Theta) - g_1(x_i, \Phi)]^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} w(x_i)[g_2(x_i, \Theta) - g_1(x_i, \hat{\Phi})]^2. \end{aligned}$$

This can be regarded as an idealized analogue of the weighted sum of squares from curve fitting divided by the degrees of freedom $WSSQ/NDOF$, so that a large value would indicate a greater possibility of eliminating the deficient model, say by a F test.

❍ $Q(\Theta)$, the limit of $S_n(\Theta)$ as $n \to \infty$ is

$$\begin{aligned} Q(\Theta) &= \lim_{n \to \infty} S_n(\Theta) \\ &= \min_{\Phi} \frac{\int_a^b w(x)[g_2(x, \Theta) - g_1(x, \Phi)]^2 f(x)\,dx}{\int_a^b f(x)\,dx}. \end{aligned}$$

This represents the final asymptotic value of $WSSQ/NDOF$ as the number of support points increases without limit.

❍ The relative error $R(n)$ defined as

$$R(n) = \left| \frac{S_n(\Theta) - Q(\Theta)}{Q(\Theta)} \right|, Q(\Theta) > 0.$$

is a measure of the number of experimental points that would be needed for $S_n(\Theta)$ to converge to $Q(\theta)$, and which would therefore be sufficient for optimum model discrimination.

❍ The spacing of support points needed for such investigations must satisfy

$$\begin{aligned} x_1 &= a \\ \frac{1}{n-1} &= \frac{\int_{x_i}^{x_{i+1}} f(t)\,dt}{\int_a^b f(t)\,dt},\ i = 1, 2, \ldots, n-1 \\ x_n &= b, \end{aligned}$$

which indicates how the spacing is dictated by the density function $f(x)$.

❍ For each choice of Θ, two numbers μ and ν have to be selected so that

$$\begin{aligned} g_2(a, \Theta) &= \mu \\ g_2(b, \Theta) &= \nu. \end{aligned}$$

For instance, if $0 \le g_2(x, \Theta) \le 1$, then $\mu = 0.1$ and $\nu = 0.9$ might be a reasonable possibility.

To illustrate the use of **eoqsol** we shall consider the case with normalized models

$$\begin{aligned} g_1(x) &= \frac{\phi x}{1 + \phi x} \\ g_2(x) &= \frac{\theta_1 x}{1 + x} + \frac{(1 - \theta_1)\theta_2 x}{1 + \theta_2 x}, \end{aligned}$$

that is, fitting a single binding site to a mixture of two sites.

Figure 18.12 shows the best possible fit of a single binding site by varying ϕ to such a two site model with $\theta_1 = 0.5$ and $\theta_2 = 10$ under two circumstances:

- Uniform: data spacing in an arithmetic progression using

$$f(x) = 1, \text{ and}$$

- Geometric: data spacing in a geometric progression using

$$f(x) = \frac{1}{x}.$$

In both cases the weighting function was taken as $w(x) = 1$, i.e. constant variance, while the range of independent variable $(a = 0.022, b = 4.6)$ was calculated for each choice of θ_1 and θ_2 to cover 80% of the maximum possible variation in g_2 for $x \ge 0$ by solving

$$\begin{aligned} g_2(a) &= 0.1, \text{ and} \\ g_2(b) &= 0.9. \end{aligned}$$

The solid curve in figure 18.12 is the curve for the true model $g2(x, (0.5, 10))$, the dashed curve is for the best-fit deficient model $g_1(x, 2.45), Q = 0.0008$ with a uniform distribution of points, while the dotted curve is for the best-fit deficient model $g_1(x, 3.16), Q = 0.002$ for a geometric distribution of points, which indicates that a geometric distribution of points is better for model discrimination than a uniform distribution in this particular example.

Figure 18.13 shows another of the **eoqsol** procedures: studying the change in Q as a function of just one θ varied. The value of all the θ parameters are fixed and a so-called reference Q is calculated. Then one of the parameters is varied systematically and the way that Q changes as this parameter is varied is then plotted.

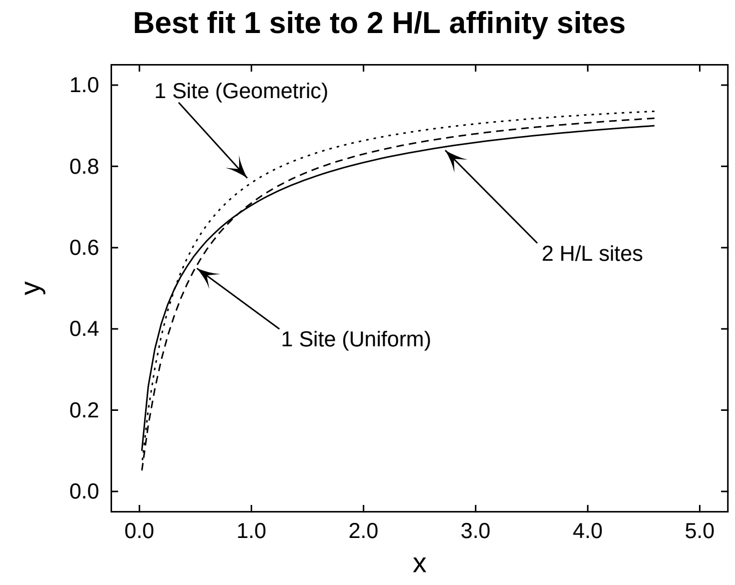


Figure 18.12: Fitting 1 site to 2 H/L sites: best fit model

From this data it is clear that, with θ_1 fixed at $\theta_1 = 0.5$ and θ_2 varied, again the geometric distribution is to be preferred for model discrimination, and that a value of say $\theta_2 \geq 10$ is required to provide sufficient power for model discrimination.

Figure 18.14 shows another of the **eoqsol** functions: studying the change in $R(n)$ as a function of just one θ varied. The value of all the θ parameters are fixed and a so-called reference Q is calculated. Then one of the parameters is varied and the way that $R(n)$ changes as this parameter is varied is then plotted. From this data

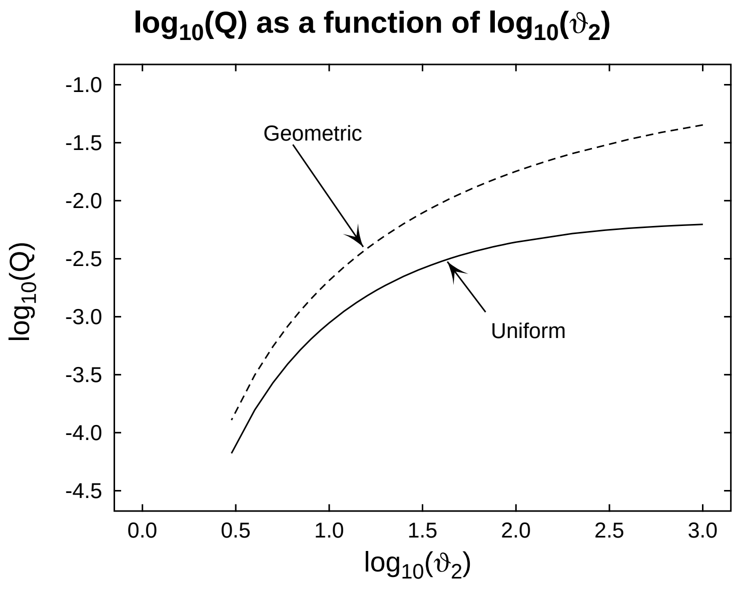


Figure 18.13: Fitting 1 site to 2 H/L sites: behaviour of Q

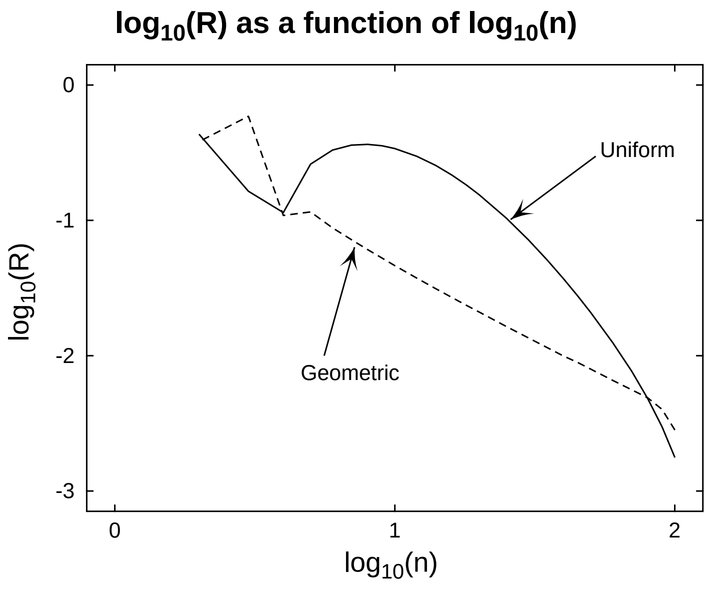


Figure 18.14: Fitting 1 site to 2 H/L sites: behaviour of R

it is clear that once more the geometric distribution is to be preferred for model discrimination, and that at value of say $n \geq 10$ is required to provide sufficient power for model discrimination.

18.14 False discovery rates FDR(BH)

Multiple testing is when several statistical tests are performed on the same data so it is necessary to control false results. One procedure is the Bonferroni correction where, for m tests and p values, results are considered significant at level α if $p \le \alpha/m$, rather than $p \le \alpha$ for single tests.

If at least one of the p values satisfies the Bonferroni restriction, the FDR(BH) false discovery rate technique (Benjamini and Hochberg J.R.satist.Soc. B (1995) 57,1, 289–300, and Benjamini et al Behavioural Brain Research 125 (2001) 279–284) is available to see if there other p values, not necessarily satisfying the Bonferroni restriction, that could also be regarded as possibly significant.

18.14.1 Example 1: FDR(BH) for a vector of p values

From the main SimFIT menu choose [Statistics] then [Statistical calculations] and then [False discovery rates from a vector p(i)] and scrutinize the default test file `fdr_bh.tf1` provided. After selecting to calculate the false discovery rates, view the table of results for all the data arranged into order of rank which is displayed next. Here m is the number of tests and i is the rank of the sample in terms of the ordered p values.

False discovery rates for a vector of p(i) values: 1
Title: Data for BH False Discovery rate calculation
Sample size = 17
Number rejected = 0
Number analysed = 17
Significance level, α = 0.05

		p	$m * p/i$	$\alpha * i/m$	
Rank	Sample	p-value	p-adjusted	BH-level	Result
1	12	0.000001	0.000017	0.002941	1
2	1	0.000013	0.000110	0.005882	1
3	3	0.000065	0.000368	0.008824	1
4	6	0.000630	0.002678	0.011765	1
5	5	0.000800	0.002720	0.014706	1
6	16	0.001700	0.004817	0.017647	1
7	2	0.003200	0.007771	0.020588	1
8	7	0.006500	0.013813	0.023529	1
9	11	0.014800	0.027956	0.026471	1
10	13	0.049000	0.083300	0.029412	0
11	14	0.094000	0.145273	0.032353	0
12	17	0.110000	0.155833	0.035294	0
13	9	0.150000	0.196154	0.038235	0
14	8	0.240000	0.291429	0.041176	0
15	15	0.450000	0.510000	0.044118	0
16	10	0.560000	0.595000	0.047059	0
17	4	0.870000	0.870000	0.050000	0

There are other options to view the results in sample order or to just show significant results, but the above table is the easiest to understand and follows the example given by Benjamini et al on this same data set.

In order to understand the FDR(BH) technique we shall explain the meanings of the above columns and, in particular, the interpretation of the colors and meaning of the 1's and 0's in the last column.

1. **Column 1**

   This is the rank i of the sample with respect to the p values. That is, the rows of the table are arranged so that the samples in row i are arranged in order of increasing p values.

2. **Column 2**

   This registers the actual number of the sample in the original order.

3. **Column 3**

   Here are the p values corresponding to the rank recorded in column 1 for the sample identified in column 2.

4. **Column 4**

   If this is table line for rank i then this is the p value adjusted by the rank and the sample size m. In other words, the adjusted p value is mp/i. Note that this column only depends on p, i and m, and the last adjusted p value is always the same as the uncorrected p value since $i = m$.

5. **Column 5**

   Here are listed the BH-levels, i.e., the BH threshold values $\alpha i/m$. Note that these only depend on α, i and m, and they have the following sequence. The value at row 1 is the Bonferroni corrected level for significance testing, and the value at line m is the significance level α, while between these extremes the values slowly increase as a function of the rank.

6. **Column 6**

   This column has a 1 if the sample is in the FDR(BH) set and a 0 otherwise

18.14.2 The systematic FDR(BH) procedure

The technique starts at row m and advances up the table until the first rank is encountered, say k, where the p value is less than or equal to the BH threshold. We then conclude that all samples from line 1 up to line k must be considered as possibly significant. So the set of possibly significant samples contains those where

$$p \le \alpha i/m,$$
$$\text{or equivalently } mp/i \le \alpha.$$

So now the importance of the color change will be clear and the interpretation of the table is obvious.

All samples numbered in column 2 up to level k with a 1 in column 6 are colored blue, which makes identification of the set of possibly significant samples easy to recognize.

The table can also be rearranged into sample order and can be displayed in such a way as to only identify the set of possibly significant samples. Also, for very large samples it is possible to scroll through the table to select sections or even to write the whole table to file.

18.14.3 Example 2: FDR(BH) for a matrix of p values

Some procedures result in matrices of p values, and this requires a more complicated approach because we have to keep track of the row and column indices. As a typical example, select the option for false discovery rate for a matrix and read in the default test file `matrix_p.tf1` which is as follows

```
0.00023   0.00060   0.40906   0.41318
0.00050   0.00005   0.32055   0.23282
0.00560   0.01362   0.43751   0.06327
```

This results from the directed correlation procedure in the multivariate statistics options using the default test files `matrix_a.tf1` for the A matrix which has dimensions 30 by 3, and `matrix_b.tf1` for the B matrix which has dimensions 30 by 4.

Proceeding with the false discovery option we obtain the following table in rank order.

```
 False discovery rates for a matrix of p(i,j) values: 1
Title: Data from directed correlation
Number of columns = 4
Number of rows = 3
Number out of range = 0
Significance level, α = 0.05
```

$A(i)$	$B(j)$	p-value	p-adjusted	BH-level	Result
2	2	0.000053	0.000632	0.004167	1
1	1	0.000231	0.001387	0.008333	1
2	1	0.000500	0.002000	0.012500	1
1	2	0.000598	0.001793	0.016667	1
3	1	0.005602	0.013446	0.020833	1
3	2	0.013624	0.027247	0.025000	1
3	4	0.063269	0.108461	0.029167	0
2	4	0.232822	0.349234	0.033333	0
2	3	0.320548	0.427398	0.037500	0
1	3	0.409063	0.490875	0.041667	0
1	4	0.413176	0.450738	0.045833	0
3	3	0.437508	0.437508	0.050000	0

As before the set of possibly significant samples is easy to identify by the 1 in the last column or blue color, but columns 1 and 2 need some explanation.

In this example column 1 indicates what the row indices of p values are, because the matrix of p values had 3 rows which originated from the 3 columns of the A matrix in the directed correlation. The second column identifies column indices for the 4 columns corresponding to the 4 columns of the B matrix. This situation is valid only for this particular matrix, and results from the convention dictating the way that the matrix of p values was constructed.

Part 19

Numerical analysis

19.1 Introduction

In data analysis it is frequently necessary to perform calculations rather than tests, e.g., calculating the determinant, eigenvalues, or singular values of a matrix to check for singularity.

19.2 Zeros of a polynomial of degree n - 1

Every real polynomial $f(x)$ of degree $n-1$ with n coefficients A_i can be represented in either coefficient or factored form, that is

$$\begin{aligned} f(x) &= A_1x^{n-1} + A_2x^{n-2} + \cdots + A_{n-1}x + A_n, \\ &= A_1(x-\alpha_1)(x-\alpha_2)\ldots(x-\alpha_{n-1}), \end{aligned}$$

where α_i for $i = 1, 2, \ldots, n-1$ are the $n-1$ zeros, i.e., roots of the equation $f(x) = 0$, and these may be non-real and repeated. In data analysis it is frequently useful to calculate the $n-1$ zeros α_i of a polynomial given the n coefficients A_i, and this SIMFIT polynomial solving procedure performs the necessary iterative calculations. However, note that such calculations are notoriously difficult for repeated zeros and high degree polynomials. Table 19.1 illustrates a calculation for the roots of the fourth degree polynomial

```
Zeros of f(x) = A(1)x^(n-1) + A(2)x^(n-2) + ... + A(n)
                Real Part  Imaginary Part
A( 1) =  1.0         0.0           -1.0i
A( 2) =  0.0         0.0            1.0i
A( 3) =  0.0        -1.0
A( 4) =  0.0         1.0
A( 5) = -1.0 (constant term)
```

Table 19.1: Zeros of a polynomial

$$\begin{aligned} f(x) &= x^4 - 1 \\ &= (x-i)(x+i)(x-1)(x+1) \end{aligned}$$

which are i, $-i$, 1, and -1. Be careful to note when using this procedure that the sequential elements of the input vector must be the polynomial coefficients in order of decreasing degree. Zeros of nonlinear functions of one or several variables are sometimes required, and these can be estimated using **usermod**.

19.3 Determinants, inverses, eigenvalues, and eigenvectors

Table 19.2 illustrates an analysis of data in the test file `matrix.tf1` to calculate parameters such as the de-

```
Value of the determinant = 4.4834E+04
Values for the current square matrix are as follows:
   1.2    4.5    6.1    7.2    8.0
   3.0    5.6    3.7    9.1   12.5
  17.1   23.4    5.5    9.2    3.3
  7.15   5.87   9.94   8.82   10.8
  12.4   4.30    7.7   8.95    1.6
Values for the current inverse are as follows:
  -0.24111  0.062912  0.00044392   0.10123  0.029774
  0.085853 -0.044069    0.052548 -0.019963   -0.0586
   0.11818  -0.17354   -0.005537   0.11957  -0.03076
   0.22291  0.067828   -0.019731  -0.25804   0.13802
  -0.17786  0.086634  -0.0076447   0.13711 -0.072265
Eigenvalues:   Real Part  Imaginary Part
                  38.861             0.0
                 -8.3436             0.0
                 -2.7508          7.2564
                 -2.7508         -7.2564
                 -2.2960             0.0
Eigenvector columns (real parts only)
 0.31942   -0.34409  -0.13613  -0.13613  -0.35398
 0.37703  -0.071958 -0.050496 -0.050496  0.062282
 0.60200    0.78212   0.80288   0.80288  -0.13074
 0.48976   -0.44619  -0.26270  -0.26270   0.78507
 0.39185    0.25617  -0.21156  -0.21156  -0.48722
Eigenvector columns (imaginary parts only)
 0.0  0.0 -0.075605  0.075605 0.0
 0.0  0.0  0.39888  -0.39888  0.0
 0.0  0.0       0.0       0.0  0.0
 0.0  0.0 -0.19106   0.19106  0.0
 0.0  0.0 -0.13856   0.13856  0.0
```

Table 19.2: Matrix example 1: Determinant, inverse, eigenvalues, eigenvectors

terminant, inverse, eigenvalues, and eigenvectors, that are frequently needed when studying design matrices. Note that the columns of eigenvectors correspond in sequence to the eigenvalues, but with non-real eigenvalues the corresponding adjacent columns correspond to real and imaginary parts of the complex conjugate eigenvectors. Thus, in the case of eigenvalue 1, i.e. 38.861, column 1 is the eigenvector, while for eigenvalue 2, i.e. $-2.7508 + 7.2564i$, eigenvector 2 has column 2 as real part and column 3 as imaginary part. Similarly, for eigenvalue 3, i.e. $-2.7508 - 7.2564i$, eigenvector 3 has column 2 as real part and minus column 3 as imaginary part. Note that with SIMFIT matrix calculations the matrices will usually be written just once to the results file for relatively small matrices if the option to display is selected, but options are also provided to save matrices to file.

19.4 Singular value decomposition

Table 19.3 shows results from a singular value decomposition of data in `f08kff.tf1`. Analysis of your own design matrix should be carried out in this way if there are singularity problems due to badly designed experiments, e.g., with independent variables equal to 0 or 1 for binary variables such as female and male. If

```
Current matrix:
 -0.57  -1.28  -0.39   0.25
 -1.93   1.08  -0.31  -2.14
   2.3   0.24    0.4  -0.35
 -1.93   0.64  -0.66   0.08
  0.15    0.3   0.15  -2.13
 -0.02   1.03  -1.43    0.5
Index  Sigma Fraction Cumulative Sigma^2 Fraction Cumulative rank = 4
 1   3.99872      0.4        0.4 15.9898   0.5334     0.5334
 2   3.00052   0.3002     0.7002  9.0031   0.3003     0.8337
 3   1.99671   0.1998        0.9 3.98686    0.133     0.9666
 4  0.999941      0.1        1.0 .999882   0.0334        1.0
Right singular vectors by row (V-transpose)
  0.825146 -0.279359  0.204799  0.446263
 -0.453045 -0.212129 -0.262209  0.825226
 -0.282853 -0.796096  0.495159 -0.202593
  0.184064 -0.493145 -0.802572 -0.280726
Left singular vectors by column (U)
 -0.0202714  0.279395   0.469005   0.769176
  -0.728415 -0.346414 -0.0169416 -0.0382903
   0.439270 -0.495457  -0.286798  0.0822225
  -0.467847  0.325841  -0.153556  -0.163626
  -0.220035 -0.642775   0.112455   0.357248
 -0.0935234  0.192680  -0.813184   0.495724
```

Table 19.3: Matrix example 2: Singular value decomposition

your design matrix has m rows and n columns, $m > n$, there should be n nonzero singular values. Otherwise only linear combinations of parameters will be estimable. Actually, many statistical techniques, such as multilinear regression, or principal components analysis, are applications of singular value decompositions. Now, given any m by n matrix A, the SVD procedure calculates a singular value decomposition in the form

$$A = U\Sigma V^T,$$

where U is an m by m orthonormal matrix of left singular vectors, V is an n by n orthonormal matrix of right singular vectors, and Σ is an m by n diagonal matrix, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ with $\sigma_i \geq 0$. However, note that SIMFIT can display the matrices U, Σ, or V^T, or write them to file, but superfluous rows and columns of zeros are suppressed in the output. Another point to note is that, whereas the singular values are uniquely determined, the left and right singular vectors are only pairwise determined up to sign, i.e., corresponding pairs can be multiplied by -1.

19.5 Pseudo inverse and rank of a matrix

Table 19.4 displays the results from calculating the pseudo inverse and rank of the matrix contained in test file `g02blf.tf1`.

Note that, for any m by n matrix A with $m \geq n$, the pseudo inverse A^+ can be obtained from the singular value decomposition as

$$A^+ = V\Sigma^{-1}U^T,$$

where, if r is the rank of A, then Σ^{-1} is the diagonal matrix where the first r diagonal elements are the inverse of the non-zero singular values, and the remaining diagonal elements are zero. The rank is estimated as the number of singular values that are greater than or equal to a tolerance factor TOL multiplied by the largest singular value.

```
Pseudo inverse with TOL = 0.0000001, rank = 4
  0.01780713  -0.01182626    0.0471568  -0.05663634  -0.003674122    0.03840807
 -0.02156477   0.04341726   0.02944571   0.02913215   -0.01378104    0.03425613
  0.05202857  -0.08126532   0.01392615   0.04744183    0.01664658    0.05759353
  0.02368605   0.03571685  -0.01380834   0.03047762    0.03566550   -0.05713431
 0.007195698 -0.001395747  0.007672032  0.005041525   0.003485692   0.007312341
```

Table 19.4: Matrix example 3: Pseudo inverse and rank

19.6 LU factorization of a matrix, norms and condition numbers

Table 19.5 illustrates the LU factorization of the matrix A in `matrix.tf1`, displayed previously in table 19.2,

```
Matrix 1-norm = 43.67 Condition number = 36.94
Matrix I-norm = 58.5  Condition number = 26.184
Lower triangular/trapezoidal L where A = PLU
       1.0
   0.72515      1.0
  0.070175 -0.22559      1.0
   0.17544 -0.11799  0.48433      1.0
   0.41813  0.30897  0.99116 -0.63186    1.0
Upper triangular/trapezoidal U where A = PLU
  17.11    23.4      5.5      9.2      3.3
        -1.2668  3.7117   2.2787 -0.79298
                 6.5514   7.0684   7.5895
                          4.3314   8.1516
                                   7.2934
Row pivot indices equivalent to P where A = PLU
     3
     5
     3
     5
     5
```

Table 19.5: Matrix example 4: LU factorization and condition number

along with the vector of row pivot indices corresponding to the pivot matrix P in the factorization $A = PLU$. As the LU representation is of interest in the solution of linear equations, this procedure also calculates the matrix norms and condition numbers needed to assess the sensitivity of the solutions to perturbations when the matrix is square. Given a vector norm $\|.\|$, a matrix A, and the set of vectors x where $\|x\| = 1$, the matrix norm subordinate to the vector norm is

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

For a m by n matrix A, the three most important norms are

$$\|A\|_1 = \max_{1\le j\le n} (\sum_{i=1}^{m} |a_{ij}|)$$

$$\|A\|_2 = (\lambda_{\max}|A^T A|)^{\frac{1}{2}}$$

$$\|A\|_\infty = \max_{1\le i\le m} (\sum_{j=1}^{n} |a_{ij}|),$$

so that the 1-norm is the maximum absolute column sum, the 2-norm is the square root of the largest eigenvalue of $A^T A$, and the infinity norm is the maximum absolute row sum. The condition numbers estimated are

$$\begin{aligned}\kappa_1(A) &= \|A\|_1\|A^{-1}\|_1\\ \kappa_\infty(A) &= \|A\|_\infty\|A^{-1}\|_\infty\\ &= \kappa_1(A^T)\end{aligned}$$

which satisfy $\kappa_1 \ge 1$, and $\kappa_\infty \ge 1$ and they are included in the tabulated output unless A is in singular, when they are infinite. For a perturbation δb to the right hand side of a linear system with $m = n$ we have

$$\begin{aligned}Ax &= b\\ A(x+\delta x) &= b+\delta b\\ \frac{\|\delta x\|}{\|x\|} &\le \kappa(A)\frac{\|\delta b\|}{\|b\|},\end{aligned}$$

while a perturbation δA to the matrix A leads to

$$\begin{aligned}(A+\delta A)(x+\delta x) &= b\\ \frac{\|\delta x\|}{\|x+\delta x\|} &\le \kappa(A)\frac{\|\delta A\|}{\|A\|},\end{aligned}$$

and, for complete generality,

$$\begin{aligned}(A+\delta A)(x+\delta x) &= b+\delta b\\ \frac{\|\delta x\|}{\|x\|} &\le \frac{\kappa(A)}{1-\kappa(A)\|\delta A\|/\|A\|}\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right)\end{aligned}$$

provided $\kappa(A)\|\delta A\|/\|A\| < 1$. These inequalities estimate bounds for the relative error in computed solutions of linear equations, so that a small condition number indicates a well-conditioned problem, a large condition number indicates an ill-conditioned problem, while an infinite condition number indicates a singular matrix and no solution. To a rough approximation; if the condition number is 10^k and computation involves n-digit precision, then the computed solution will have about $(n-k)$-digit precision.

19.7 QR factorization of a matrix

Table 19.6 illustrates the QR factorization of data in `matrix.tf2`. This involves factorizing a n by m matrix as in

$$\begin{aligned}A &= QR \text{ when } n = m\\ &= Q_1Q_2\begin{pmatrix}R\\0\end{pmatrix} \text{ when } n > m\\ &= Q(R_1R_2) \text{ when } n < m\end{aligned}$$

where Q is a n by n orthogonal matrix and R is either upper triangular or upper trapezoidal. You can display or write to file the matrices Q, Q_1, R, or R_1.

19.8 Cholesky factorization of a positive-definite symmetric matrix

Table 19.7 shows how factorization of data in `matrix.tf3` into a lower and upper triangular matrix can be achieved. Note that factorization as in

$$A = R^T R$$

will only succeed when the matrix A supplied is symmetric and positive-definite, as when A is a covariance matrix. In all other cases, error messages will be issued.

```
The orthogonal matrix Q1
 -0.10195  -0.25041   0.07498   0.73028   0.55734
 -0.39929  -0.21649   0.72954  -0.29596   0.27394
 -0.53861   0.26380  -0.32945  -0.14892   0.18269
 -0.31008  -0.30018  -0.15220  -0.42044   0.65038
 -0.28205  -0.52829  0.075783   0.27828  -0.69913
 -0.30669  -0.31512  -0.55196  0.032818   0.14906
 -0.51992   0.59358   0.14157   0.31879  -0.25637
The upper triangular/trapezoidal matrix R
 -11.771  -18.44  -13.989 -10.803  -15.319
         -6.8692 -0.12917  -4.551  -4.2543
                           5.8895 -0.34487 -0.0057542
                                   8.6062    -1.1373
                                              2.5191
```

Table 19.6: Matrix example 5: QR factorization

```
Current positive-definite symmetric matrix
  4.16 -3.12  0.56  -0.1
 -3.12  5.03 -0.83  1.09
  0.56 -0.83  0.76  0.34
  -0.1  1.09  0.34  1.18
 Lower triangular R where A = (R^T)R
    2.0396
   -1.5297   1.6401
   0.27456 -0.24998  0.78875
 -0.049029  0.61886  0.64427  0.61606
 Upper triangular R^T where A = (R^T)R
  2.0396 -1.5297  0.27456 -0.049029
          1.6401 -0.24998   0.61886
                  0.78875   0.64427
                            0.61606
```

Table 19.7: Matrix example 6: Cholesky factorization

19.9 Matrix multiplication

Given two matrices A and B, it is frequently necessary to form the product, or the product of the transposes, as an m by n matrix C, where $m \geq 1$ and $n \geq 1$. The options are

$$\begin{aligned}
C &= AB, \text{ where } A \text{ is } m \times k, \text{ and } B \text{ is } k \times n,\\
C &= A^T B, \text{ where } A \text{ is } k \times m, \text{ and } B \text{ is } k \times n,\\
C &= AB^T, \text{ where } A \text{ is } m \times k, \text{ and } B \text{ is } n \times k,\\
C &= A^T B^T, \text{ where } A \text{ is } k \times m, \text{ and } B \text{ is } n \times k,
\end{aligned}$$

as long as $k \geq 1$ and the dimensions of A and B are appropriate to form the product, as indicated. For instance, using the singular value decomposition routine just described, followed by multiplying the U, Σ, and

V^T matrices for the simple 4 by 3 matrix indicated shows that

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1/\sqrt{6} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \\ -1/\sqrt{6} & 0 & -1/\sqrt{2} \\ -2/\sqrt{6} & 0 & 0 \end{pmatrix} \begin{pmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 0 & 1 & 0 \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \end{pmatrix}.$$

19.10 Evaluation of quadratic forms

Table 19.8 illustrates a special type of matrix multiplication that is frequently required, namely

```
Title of matrix A:
4 by 4 positive-definite symmetric matrix
Title of vector x:
Vector with 4 components 1, 2, 3, 4
(x^T)*A*x = 55.72

Title of matrix A:
4 by 4 positive-definite symmetric matrix
Title of vector x:
Vector with 4 components 1, 2, 3, 4
(x^T)*(A^{-1})*x = 20.635
```

Table 19.8: Matrix example 7: Evaluation of quadratic forms

$$Q_1 = x^T A x$$
$$Q_2 = x^T A^{-1} x$$

for a square n by n matrix A and a vector x of length n. In this case the data analyzed are in the test files `matrix.tf3` and `vector.tf3`. The form Q_1 can always be calculated but the form Q_2 requires that the matrix A is positive-definite and symmetric, which is the case when A is a covariance matrix and a Mahalanobis distance is required.

19.11 Solving $Ax = b$ (full rank)

Table 19.9 shows the computed solution for

$$Ax = b$$
$$x = A^{-1}b$$

where A is the square matrix of table 19.2 and b is the vector $1, 2, 3, 4$. When the n by m matrix A is not square or is singular, a m by n pseudo inverse A^+ can be defined in terms of the QR factorization or singular value decomposition as

$$\begin{aligned} A^+ &= R^{-1}Q_1^T \text{ if } A \text{ has full column rank} \\ &= V\Omega U^T \text{ if } A \text{ is rank deficient,} \end{aligned}$$

where diagonal elements of Ω are reciprocals of the singular values.

19.12 Solving $Ax = b$ $(L_1, L_2, L_\infty$norms$)$

Table 19.10 illustrates the solutions of the overdetermined linear system $Ax = b$ where A is the 7 by 5 matrix of table 19.3 b is the vector $(1, 2, 3, 4, 5, 6, 7,)$, i.e. the test files `matrix.tf2` and `vector.tf2`.

```
Solution to Ax = b where the square matrix A is:
Matrix of dimension 5 by 5 (i.e. matrix.tf1}
and the vector b is:
Vector with 5 components 1, 2, 3, 4, 5 (i.e. vector.tf1}
rhs vector (b)   Solution (x)
       1.0        0.43985
       2.0        -0.2175
       3.0        0.07896
       4.0      -0.042704
       5.0        0.15959
```

Table 19.9: Solving $Ax = b$: square where A^{-1} exists

```
L1-norm solution to Ax = b
      1.9514
     0.42111
    -0.56336
    0.043038
    -0.67286
L1-norm objective function = 4.9252

L2-norm solution to Ax = b
      1.2955
     0.77603
    -0.33657
    0.082384
    -0.98542
The rank of A (from SVD) = 5
L2-norm objective function = 10.962

L_infinity norm solution to Ax = b
      1.053
    0.74896
   -0.27683
    0.26139
   -0.97905
L_infinity norm objective function = 1.5227
```

Table 19.10: Solving $Ax = b$: overdetermined in 1, 2 and ∞ norms

The solutions illustrated list the parameters that minimize the residual vector $r = Ax - b$ corresponding to the three usual vector norms as follows.

- The 1-norm $\|r\|_1$
  This finds a possible solution such that the sum of the absolute values of the residuals is minimized. The solution is achieved by iteration from starting estimates provided, which can be all -1, all 0 (the usual first choice), all 1, all user-supplied, or chosen randomly from a uniform distribution on [-100,100]. It may be necessary to scale the input data and experiment with starting estimates to locate a global best-fit minimum with difficult cases.

- The 2-norm $\|r\|_2$
  This finds the unique least squares solution that minimizes the Euclidean distance, i.e. the sum of

squares of residuals.

- The ∞-norm $||r||_\infty$
  This finds the solution that minimizes the largest absolute residual.

19.13 The symmetric eigenvalue problem

Table 19.11 illustrates the solution for a symmetric eigenvalue problem, that is, finding the eigenvectors and eigenvalues for the system

$$Ax = \lambda Bx,$$

where A and B are symmetric matrices of the same dimensions and, in addition, B is positive definite.

```
Matrix A:
  0.24  03.9  0.42 -0.16
  0.39 -01.1  0.79  0.63
  0.42  07.9 -0.25  0.48
 -0.16  06.3  0.48 -0.03

Matrix B:
  4.16 -3.12  0.56  -0.1
 -3.12  5.03 -0.83  1.09
  0.56 -0.83  0.76  0.34
  -0.1  1.09  0.34  1.18

Eigenvalues...Case: Ax = lambda*Bx
  -2.2254
 -0.45476
  0.10008
   1.1270

Eigenvectors by column...Case Ax = lambda*Bx
 -0.069006  0.30795  -0.44694 -0.55279
 -0.5.7401  0.53286 -0.037084 -0.67661
   -1.5428 -0.34964  0.050477 -0.92759
    1.4004 -0.62111   0.47425  0.25095
```

Table 19.11: The symmetric eigenvalue problem

In the case of table 19.11, the data for A were contained in test file `matrix.tf4`, while B is the matrix in `matrix.tf3`. It should be noted that the alternative problems $ABx = \lambda x$ and $BAx = \lambda x$ can also be solved and, in each case, the eigenvectors are available as the columns of a matrix X that is normalized so that

$$X^T BX = I, \text{ for } Ax = \lambda Bx, \text{ and } ABx = \lambda x,$$
$$X^T B^{-1} X = I, \text{ for } BAx = \lambda x.$$

19.14 User-defined models

Many numerical procedures require users to provide model equations, and this is done in SIMFIT by supplying model files. In order to assist users to develop their own models a special program, **usermod**, is distributed as part of the SIMFIT package. This provides the following procedures.

- ❏ After a model has been selected it is checked for consistency. If all is well the appropriate parts of the program will now be able to use that particular model. If the model has an error it will be specified and you can use your editor to attempt a repair. Note that, after any editing, the model file must be read in again for checking.
- ❏ After a model has been accepted you can check, that is, supply the arguments and observe the stack operations which are displayed in color-code as the model is evaluated.
- ❏ For single functions of one variable you can plot, find zeros or integrate.
- ❏ For single functions of two variables you can plot, or integrate.
- ❏ For several functions of one variable you can plot selected functions.
- ❏ For n functions of m variables you can find simultaneous zeros if $n = m$, integrate for any n and m, or optimize if $m = n - 1$.
- ❏ Default settings are provided for parameters, limits, tolerances and starting estimates, and these can be edited interactively as required. Note that parameters $p(i)$ used by the models will be those set from the main program, and the same is true for absolute error tolerance $epsabs$, relative error tolerance $epsrel$, and the integration limits $blim(i)$ and $tlim(i)$.
- ❏ A default template is provided which users can edit to create their own models.

19.15 Locating a zero of one function of one variable

Users must supply a relative error accuracy factor $epsrel$, two values A and B, and a constant C such that, for $g(x) = f(x) - C$, then $g(A)g(B) < 0$. If the values supplied are such that $g(A)g(B) > 0$, the program will attempt to enlarge the interval in order to bracket a zero, but it will not change the sign of A or B. Users must do this if necessary by editing the starting estimates A and B. The program returns the root as X if successful, where X and Y have been located such that

$$|X - Y| \le 2.0 \times epsrel \times |Z|$$

and $|g(Z)|$ is the smallest known function value, as described for NAG routine C05AZF.

As an example, input the special function model file `usermods.tf1` which defines one equation in one variable, namely the cumulative normal distribution function (page 420) for which the model is as follows.

```
%
Model: f(x) = phi(x) i.e. the normal cdf
%
1 equation
1 variable
0 parameters
%
x
phi(x)
f(1)
%
```

Input $f(x) = 0.975$ so the routine is required to estimate x such that

$$0.975 = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{t^2}{2}\right) dt$$

and, after setting some reasonable starting estimates, e.g., the defaults (-1,1), the following message will be printed

```
Success : Root =  1.96000E+00 (EPSREL =  1.00000E-03)
```

giving the root estimated by the SimFIT implementation of C05AZF.

19.16 Locating zeros of n functions of n variables

The model file must define a system of n equations in n variables and the program will attempt to locate $x_1, x_2, \ldots, x_n$ such that

$$f_i(x_1, x_2, \ldots, x_n) = 0, \text{ for } i = 1, 2, \ldots, n.$$

Users must supply good starting estimates by editing the default $y_1, y_2, \ldots, y_n$, or installing a new y vector from a file, and the accuracy can be controlled by varying $epsrel$, since the program attempts to ensure that

$$|x - \hat{x}| \leq epsrel \times |\hat{x}|,$$

where $\hat{x}$ is the true solution, as described for NAG routine C05NBF. Failure to converge will lead to nonzero IFAIL values, requiring new starting estimates.

As an example, input the test file `c05nbf_e.mod` which defines 9 equations in 9 variables, and after setting $y(i) = 0$ for $i = 1, 2, \ldots, 9$ select to locate zeros of n equations in n variables. The following table will result

```
From C05NBF: IFAIL =  0, FNORM = 7.448E-10, XTOL = 0.001
x(1) = -0.570653 ... fvec(1) =  0.00000252679
x(2) = -0.681625 ... fvec(2) =  0.0000156881
x(3) = -0.701732 ... fvec(3) =  0.00000028357
x(4) = -0.704215 ... fvec(4) = -0.0000130839
x(5) = -0.701367 ... fvec(5) =  0.000987684
x(6) = -0.691865 ... fvec(6) =  0.00000655571
x(7) = -0.665794 ... fvec(7) = -0.0000130536
x(8) = -0.596034 ... fvec(8) =  0.00000117770
x(9) = -0.416411 ... fvec(9) =  0.00000295110
```

showing the solution vector and vector of function values for the tridiagonal system

$$\begin{aligned}(3 - 2x_1)x_1 - 2x_2 &= -1\\ -x_{i-1} + (3 - 2x_i)x_i - 2x_{i+1} &= -1, \ i = 2, 3, \ldots, 8\\ -x_8 + (3 - 2x_9)x_9 &= -1\end{aligned}$$

estimated by the SimFIT implementation of C05NBF. The model file is as follows.

```
%
Example: 9 functions of 9 variables as in NAG C05NBF
         set y(1) to y(9) = -1 for good starting estimates
.............
f(1)=(3-2x(1))x(1)-2x(2)+1, ..., f9=-x(8)+(3-2x(9))x(9)+1
The fortran loop is defined in the NAG documentation as:
do k = 1, n
   fvec(k) = (three - two*x(k))*x(k) + one
   if (k.gt.1) fvec(k) = fvec(k) - x(k-1)
   if (k.lt.n) fvec(k) = fvec(k) - two*x(k+1)
enddo
Usage as follows
Select simulation then open program usermod
Select n functions of m variables then read in this file
specifying 9 functions of 9 variables
Select to find a zero of n functions of n variables
Set all y(i) = -1 then solve f(x) = 0
NAG reports -.5707,-.6816,-.7017, -.7042,-.7014,
            -.6919,-.6658,-.5960,-.41640
.............
 %
 9 equations
 9 variables
 0 parameters
 %
begin{expression}
f(1) = (3 - 2y(1))y(1) + 1 - 2y(2)
f(2) = (3 - 2y(2))y(2) + 1 - y(1) - 2y(3)
f(3) = (3 - 2y(3))y(3) + 1 - y(2) - 2y(4)
f(4) = (3 - 2y(4))y(4) + 1 - y(3) - 2y(5)
f(5) = (3 - 2y(5))y(5) + 1 - y(4) - 2y(6)
f(6) = (3 - 2y(6))y(6) + 1 - y(5) - 2y(7)
f(7) = (3 - 2y(7))y(7) + 1 - y(6) - 2y(8)
f(8) = (3 - 2y(8))y(8) + 1 - y(7) - 2y(9)
f(9) = (3 - 2y(9))y(9) + 1 - y(8)
end{expression}
%
```

19.17 Integrating one function of one variable

The program accepts a user defined model for a single function of one variable and returns two estimates I_1 and I_2 for the integral

$$I = \int_A^B f(x)\,dx,$$

where A and B are supplied interactively. The value of I_1 is calculated by Simpson's rule and is rather approximate, while that of I_2 is calculated by adaptive quadrature. For smooth functions over a limited range these should agree fairly closely, but large differences suggest a difficult integral, e.g., with spikes, requiring more careful investigation. The values of $epsrel$ and $epsabs$ control the accuracy of adaptive quadrature such that, usually

$$\begin{aligned} |I - I_2| &\le tol \\ tol &= \max(|epsabs|, |epsrel| \times |I|) \\ |I - I_2| &\le ABSERR \\ ABSERR &\le tol, \end{aligned}$$

as described for NAG routine D01AJF.

As an example, input the file `usermod1.tf5` which defines the function

$$f(x) = p_1 \exp(p_2 x)$$

which is expressed as a model as follows.

```
 %
Model: f(x) = p(1)*exp[p(2)*x]
 %
 1 equation
 1 variable
 2 parameters
 %
 p(2)
 x
 multiply
 exponential
 p(1)
 multiply
 f(1)
 %
```

Setting $p_1 = p_2 = 1$ and requesting integration, gives

```
Numerical quadrature over the range: 0.0, 1.0

Number of Simpson divisions  = 200
Area by the Simpson rule     = 1.71828

IFAIL (from D01AJF)          = 0
EPSABS                       = 0.000001
EPSREL                       = 0.001
ABSERR                       = 3.81535E-15
Area by adaptive integration = 1.71828
```

for the areas by Simpson's rule and the SIMFIT implementation of D01AJF.

19.18 Integrating n functions of m variables

The program accepts a user defined model for n functions of m variables and estimates the n integrals

$$I_i = \int_{A_1}^{B_1} \int_{A_2}^{B_2} \dots \int_{A_m}^{B_m} f_i(x_1, x_2, \dots, x_m)\, dx_m \dots dx_2\, dx_1$$

for $i = 1, 2, \ldots, n$, where the limits are taken from the arrays $A_i = blim(i)$ and $B_i = tlim(i)$. The procedure only returns IFAIL = 0 when

$$\max_i (ABSEST(i)) \le \max(|epsabs|, |epsrel| \times \max_i |FINEST(i)|),$$

where $ABSEST(i)$ is the estimated absolute error in $FINEST(i)$, the final estimate for integral i, as described for NAG routine D01EAF.

As an example, input the test file `d01fcf_e.mod` which defines the the model used to evaluate the integral

$$I = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \frac{4u_1 u_3^2 \exp(2u_1 u_3)}{(1 + u_2 + u_4)^2} \, du_4 \, du_3 \, du_2 \, du_1$$

as follows.

```
%

NAG: D01FCF example of one function of four variables
Note: this model can also be used to test D01EAF
-------------------------------------------------------
f(y) = {4y(1)y(3)^2[exp(2y(1)y(3))]}/{1 + y(2) + y(4)}^2
-------------------------------------------------------
Usage as follows
Select simulation then open program usermod
Read in this model
Set tolerances and limits (suggest all limits 0,1)
Select integrate n functions of m variables
NAG reports 0.5754 for epsrel=.0001

%
1 equation
4 variables
0 parameters
%
begin{expression}
f(1) = 4y(1)y(3)^2[exp(2y(1)y(3))]/[1.0 + y(2) + y(4)]^2
end{expression}
%
```

On requesting integration of n functions of m variables over the range $(0, 1)$, the table

```
IFAIL (from D01EAF) =      0
EPSABS              =  1.00000E-06
EPSREL              =  1.00000E-03

Number  BLIM   TLIM
     1   0.0    1.0
     2   0.0    1.0
     3   0.0    1.0
     4   0.0    1.0
Number     INTEGRAL       ABSEST
     1     0.575333  0.000107821
```

will be printed, showing the results from the SɪᴍFɪT implementation of D01EAF.

19.19 Bound-constrained quasi-Newton optimization

The user supplied model must define $n + 1$ functions of n variables as follows

$$\begin{aligned} f(1) &= F(x_1, x_2, \ldots, x_n) \\ f(2) &= \partial F/\partial x_1 \\ f(3) &= \partial F/\partial x_2 \\ &\ldots \\ f(n+1) &= \partial F/\partial x_n \end{aligned}$$

as the partial derivatives are required in addition to the function value. The limited memory quasi-Newton optimization procedure also requires several other parameters, as now listed.

❏ $MHESS$ is the number of limited memory corrections to the Hessian that are stored. The value of 5 is recommended but, for difficult problems, this can be varied in the range 4 to 17.

❏ $FACTR$ should be about 1.0e+12 for low precision, 1.0e+07 for medium precision, and 1.0e+01 for high precision. Convergence is controlled by $FACTR$ and $PGTOL$ and will be accepted if

$$|F_k - F_{k+1}|/\max(|F_k|, |F_{k+1}|, 1) \le FACTR * EPSMCH$$

at iteration $k + 1$, where $EPSMCH$ is machine precision, or if

$$\max_i(\text{Projected Gradient}(i)) \le PGTOL.$$

❏ Starting estimates and bounds on the variables can be set by editing the defaults or by installing from a data file.

❏ The parameter $IPRINT$ allows intermediate output every $IPRINT$ iterations, and the final gradient vector can also be printed if required.

❏ The program opens two files at the start of each optimization session, `w_usermod.err` stores intermediate output every $IPRINT$ iterations plus any error messages, while `iterate.dat` stores all iteration details, as for **qnfit** and **deqsol**, when they use the LBFGSB suite for optimization. Note that, when $IPRINT > 100$ full output, including intermediate coordinates, is written to `w_usermod.err` at each iteration.

As an example, input the model file `optimum_e.mod`, defining Rosenbruck's two dimensional test function

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

which has a unique minimum at $x = 1, y = 1$ and is represented as follows.

```
%
 Rosenbrock's 2-dimensional function for optimisation

 f(1) = 100(y - x^2)^2 + (1 - x)^2
 f(2) = g(1)
      = d(f(1))/dx
      = -400x(y - x^2) - 2(1 - x)
 f(3) = g(2)
      = d(f(1))/dy
      = 200(y - x^2)
 %
 3 equations
 2 variables
 0 parameters
 %
 begin{expression}
 A = y - x^2
 B = 1 - x
 f(1) = 100*A^2 + B^2
 f(2) = -400A*x - 2B
 f(3) = 200A
 end{expression}
 %
```

The iteration, starting at $x = 0$, $y = 0$ with $IPRINT = 5$ proceeds as follows

```
Iterate     F(x)        |prj.grd.|    Task
      1      0.69219         5.0534    NEW_X
      6      0.21146         3.1782    NEW_X
     11     0.017938         0.3592    NEW_X
     16   0.00017768       0.044729    NEW_X
     20   5.5951E-13   0.0000072120    CONVERGENCE:
          NORM OF PROJECTED GRADIENT <= PGTOL
x(1) = 1.0, dF(x)/dx(1) = 0.00000721198
x(2) = 1.0, dF(x)/dx(2) = -0.00000287189
```

The parameter $TASK$ informs users of the action required after each intermediate iteration, then finally it records the reason for termination of the optimization.

19.20 Plotting contours for Rosenbrock optimization trajectory

The coordinates for the optimization trajectory, shown plotted as a thick segmented line in figure 19.1, were taken from the file `w_usermod.err`, which was constructed from a separate run with $IPRINT = 101$ then added as an extra file to overlay the contours.

Care is sometimes needed to create satisfactory contour diagrams, and it helps both to understand the mathematical properties of the function $f(x, y)$ being plotted, and also to appreciate how SIMFIT creates a default contour diagram from the function values supplied. The algorithm for creating contours first performs a scan of the function values for the minimum and maximum values, then it divides the interval into an arithmetic progression, plots the contours, breaks the contours randomly to add keys, and prints a table of contour values corresponding to the keys. As an example, consider figure 19.1 which plots contours for the previously discussed Rosenbrock's function in the vicinity of the unique minimum at (1,1). For smooth contours, 100 divisions were used on each axis, and user-defined proportionately increasing contour spacing was used to capture the shape of the function around the minimum. If the default arithmetic or geometric progressions

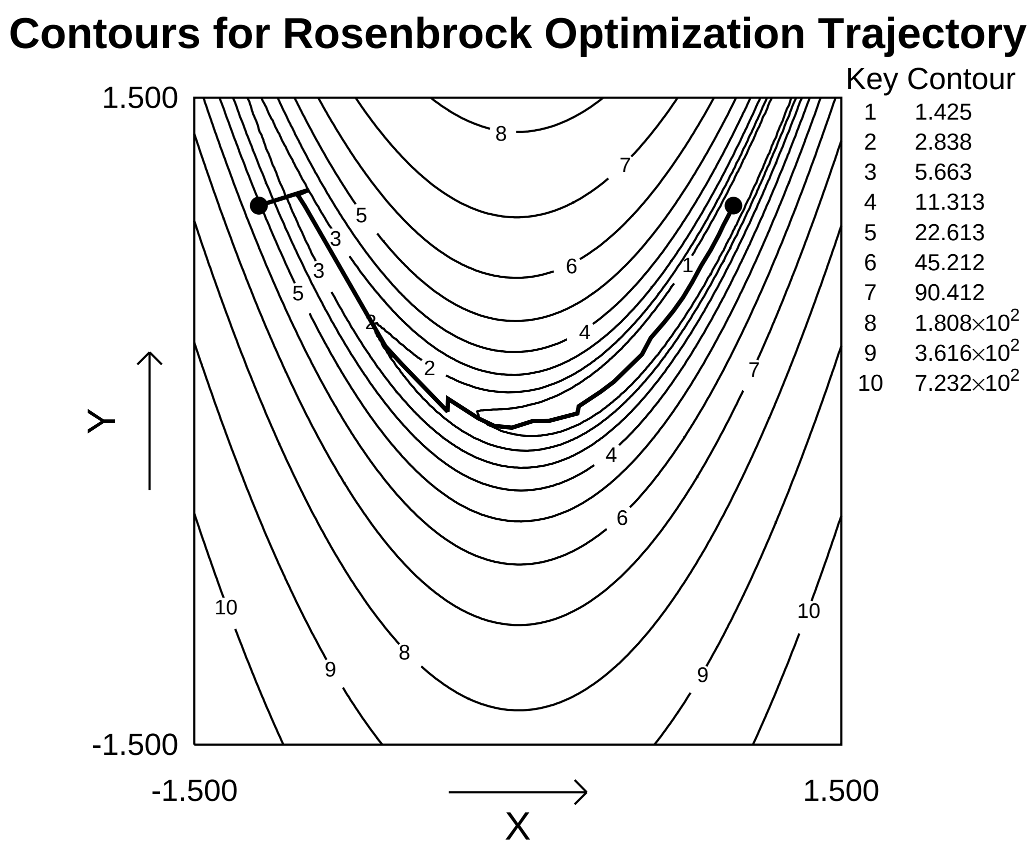


Figure 19.1: Contour diagram for Rosenbrock optimization trajectory

are inadequate, as in this case, users can select contour spacing by supplying a vector of proportions, which causes contours to be placed at those proportions of the interval between the minimum and maximum function values. The number of contours can be varied, and the keys and table of contour values can be suppressed.

Part 20

Graph plotting techniques

20.1 Graphical objects and plotting styles

20.1.1 Symbols

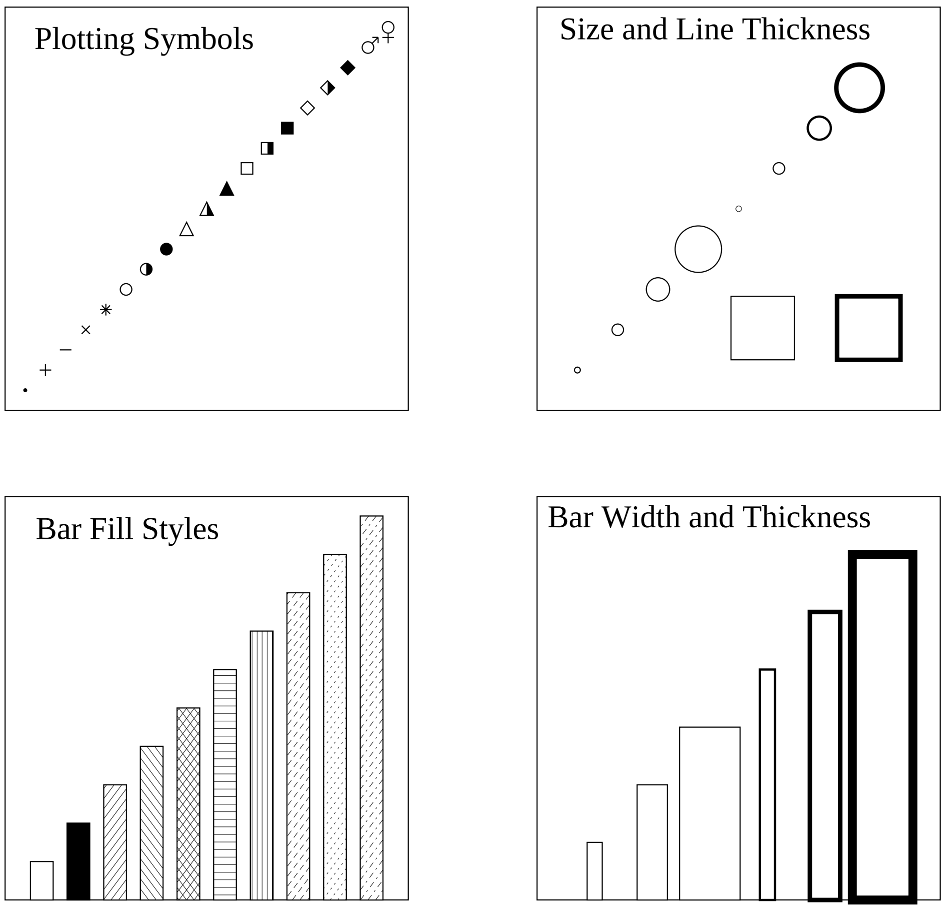


Figure 20.1: Symbols, fill styles, sizes and widths.

Figure 20.1 shows how individual sizes and line thicknesses of plotting symbols can be varied independently. Also, bars can be used as plotting symbols, with the convention that the bars extend from a baseline up to the x, y coordinate of the current point. Such bars can have widths, fill-styles and line thicknesses varied.

20.1.2 Lines: standard types

There are four standard SimFIT line types, normal, dashed, dotted and dot-dashed, and error bars can terminate with or without end caps if required, as shown in figure 20.2. Special effects can be created using stair step lines, which can be used to plot *cdfs* for statistical distributions, or survival curves from survivor functions, and vector type lines, which can be used to plot orbits of differential equations. Note that steps can be first y then x, or first x then y, while vector arrows can point in the direction of increasing or decreasing t, and lines can have variable thickness.

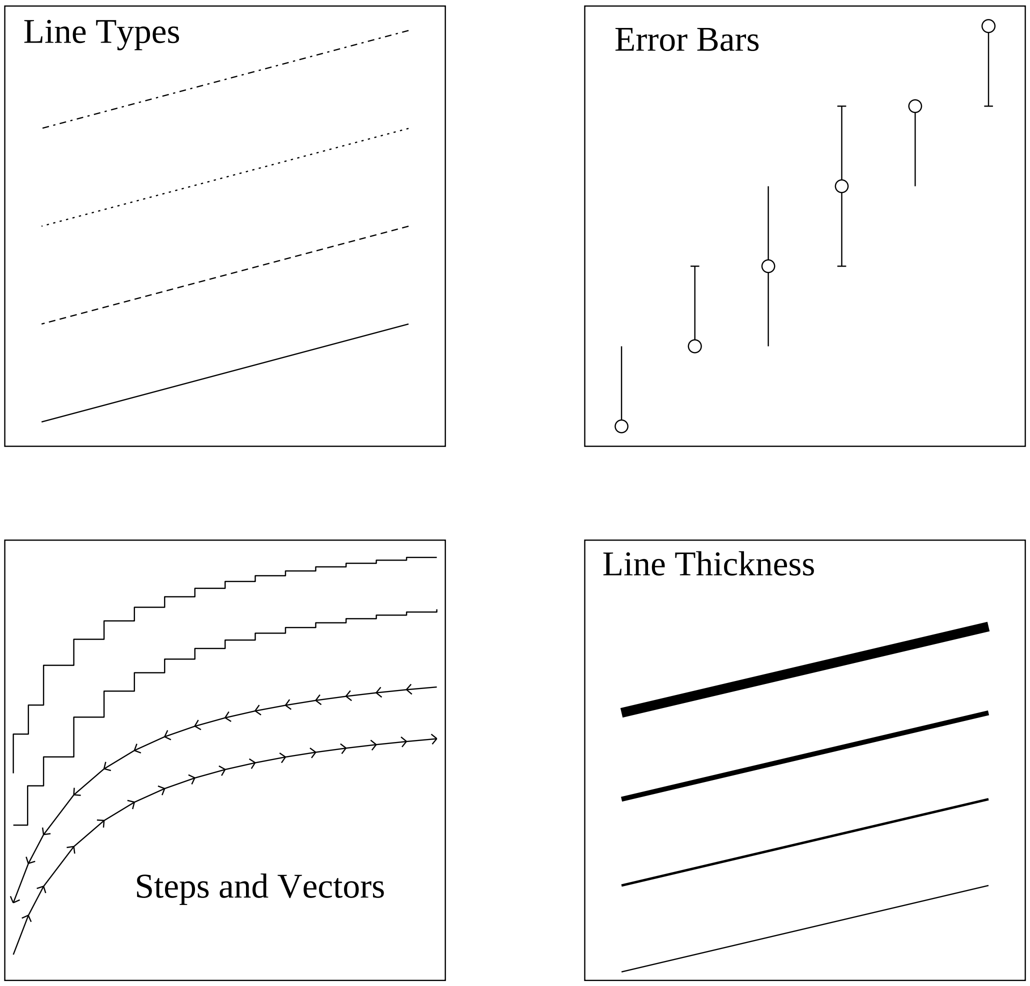


Figure 20.2: Lines: standard types

Program **simplot** reads in default options for the sequence of line types, symbol types, colors, barchart styles, piechart styles and labels which will then correspond to the sequence of data files. Changes can be made interactively and stored as graphics configuration templates if required. However, to make permanent changes to the defaults, you configure the defaults from the main SimFIT configuration option, or from program **simplot**.

20.1.3 Lines: extending to boundaries

Figure 20.3 illustrates the alternative techniques available in SIMFIT when the data to be plotted are clipped to boundaries so as to eliminate points that are identified by symbols and also joined by lines.

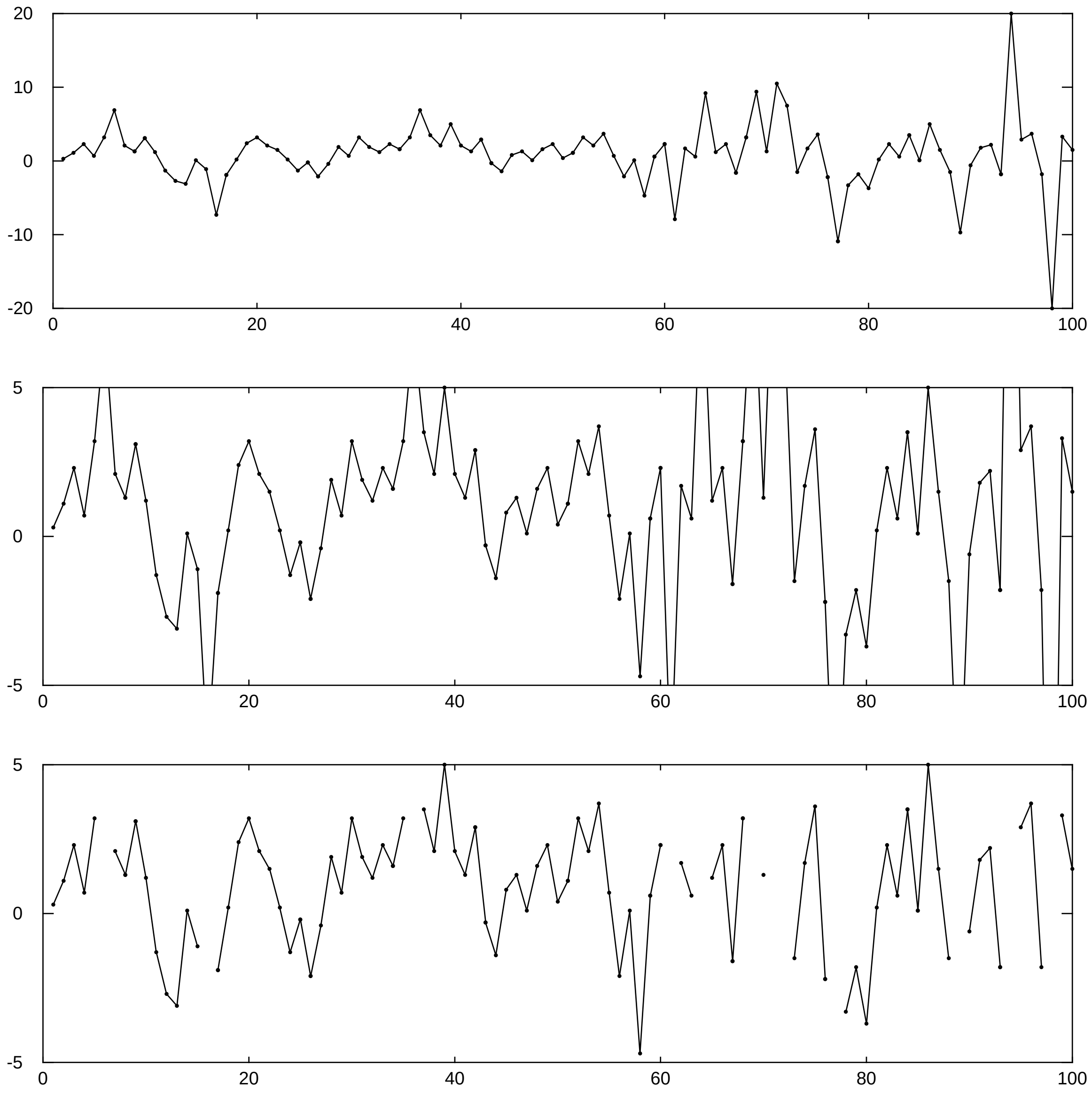


Figure 20.3: Lines: extending to boundaries

The first graph shows what happens when the test file `zigzag.tf1` was plotted with dots for symbols and lines connecting the points, but with all the data within the boundaries. The second graph illustrates how the lines can be extended to the clipping boundary to indicate the direction in which the next undisplayed symbol is located, while the third figure shows what happens when the facility to extend lines to boundaries is suppressed.

Note that these plots were first generated as .ps files using the flat-shape plotting option, then a PostScript x stretch factor of 2 (page 210) was selected, followed by the use of GSview to transform to .eps and so recalculate the BoundingBox parameters.

20.1.4 Text

Figure 20.4 shows how fonts can be used in any size or rotation and with many nonstandard accents, e.g., $\hat{\theta}$.

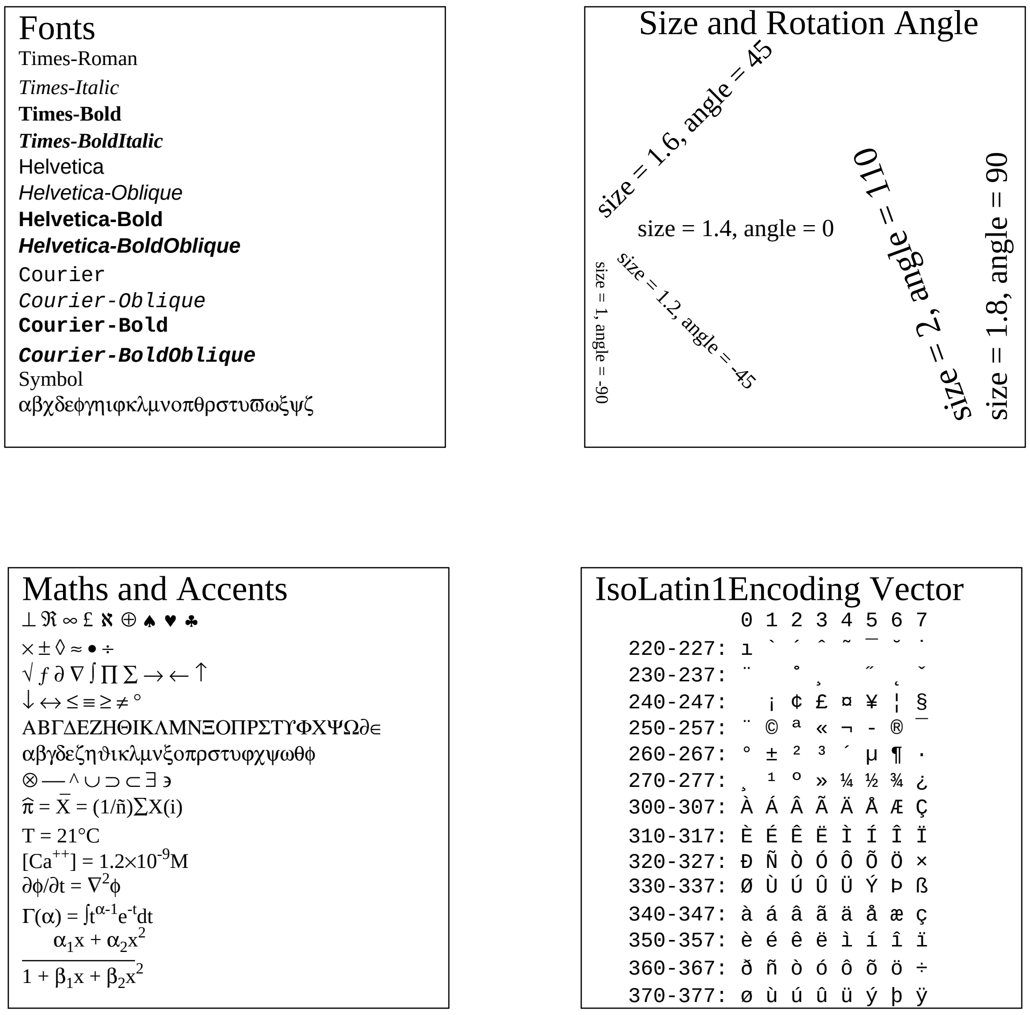


Figure 20.4: Text, maths and accents.

Special effects can be created using graphics fonts such as ZapfDingbats, or user-supplied dedicated special effect functions, as described elsewhere (page 366). Scientific symbols and simple mathematical equations can be generated, but the best way to get complicated equations, chemical formulas, photographs or other bitmaps into SimFIT graphs is to use PSfrag or **editps**.

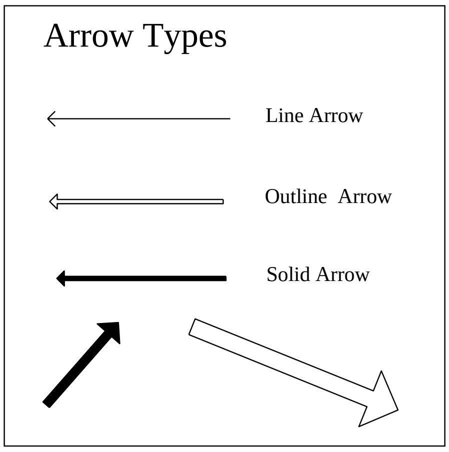
Figure 20.4 demonstrates several possibilities for displaying mathematical formulae directly in SimFIT graphs, and it also lists the octal codes for some commonly required characters from the IsoLatin1 encoding. Actually, octal codes can be typed in directly (e.g., `\361` instead of ñ), but note that text strings in SimFIT plots can be edited at two levels: at the simple level only standard characters can be typed in, but at the advanced level nonstandard symbols and maths characters can be selected from a font table. Note that, while accents can be added individually to any standard character, they will not be placed so accurately as when using the corresponding hard-wired characters e.g., from the IsoLatin1 encoding.

20.1.5 Fonts, character sizes and line thicknesses

The fonts, letter sizes, and line thicknesses used in SimFIT graphics are those chosen from the PostScript menu, so, whenever a font or line thickness is changed, the new details are written to the PostScript configuration file `w_ps.cfg`. If the size or thickness selected is not too extreme, it will then be stored as the default to be used next time. However, it should be noted that, when the default sizes are changed, the titles, legends, labels, etc. may not be positioned correctly. You can, of course, always make a title, legend, or label fit correctly by moving it about, but, if this is necessary, you may find that the defaults are restored next time you use SimFIT graphics. If you insist on using an extremely small or extremely large font size or line thickness and SimFIT keeps restoring the defaults, then you can overcome this by editing the PostScript configuration file `w_ps.cfg` and making it read-only. Users who know PostScript will prefer to use the advanced PostScript option, whereby the users own header file can be automatically added to the PostScript file after the SimFIT dictionary has been defined, in order to re-define the fonts, line thicknesses or introduce new definitions, logos plotting symbols, etc.

20.1.6 Arrows

Figure 20.5 shows that arrows can be of three types: line, hollow or solid and these can be of any size.



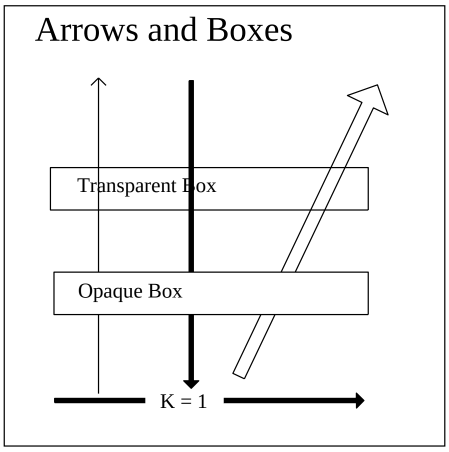


Figure 20.5: Arrows and boxes

However use can be made of headless arrows to create special effects. From this point of view a headless line arrow is simply a line which can be solid, dashed, dotted or dash-dotted. These are useful for adding arbitrary lines. A headless outline arrow is essentially a box which can be of two types: transparent or opaque. Note that the order of priority in plotting is

$$\boxed{\text{Extra Text} > \text{Graphical Objects} > \text{Data plotted, titles and legends}}$$

and this allows boxes to be used to simply obliterate plotted data or to surround extra text allowing the background to show through. Transparent boxes, are useful for surrounding information panels, opaque boxes are required for chemical formulae or mathematical equations, while background colored solid boxes can be used to blank out features as shown in figure 20.5. To surround a text string by a rectangular box for emphasis, position the string, generate a transparent rectangular box, then drag the opposing corners to the required coordinates.

20.1.7 Basic plotting styles

SimFIT offers a large choice of options to present data and results from model fitting and, to illustrate the basic plotting styles, an example from fitting the three epidemic differential equations using **deqsol** will be presented.

Perhaps the usual style for scientific graphics is to simply display the data and axes with tick marks pointing inwards and no additional features (see the Standard Axes plot of figure 20.6). However, the tick marks

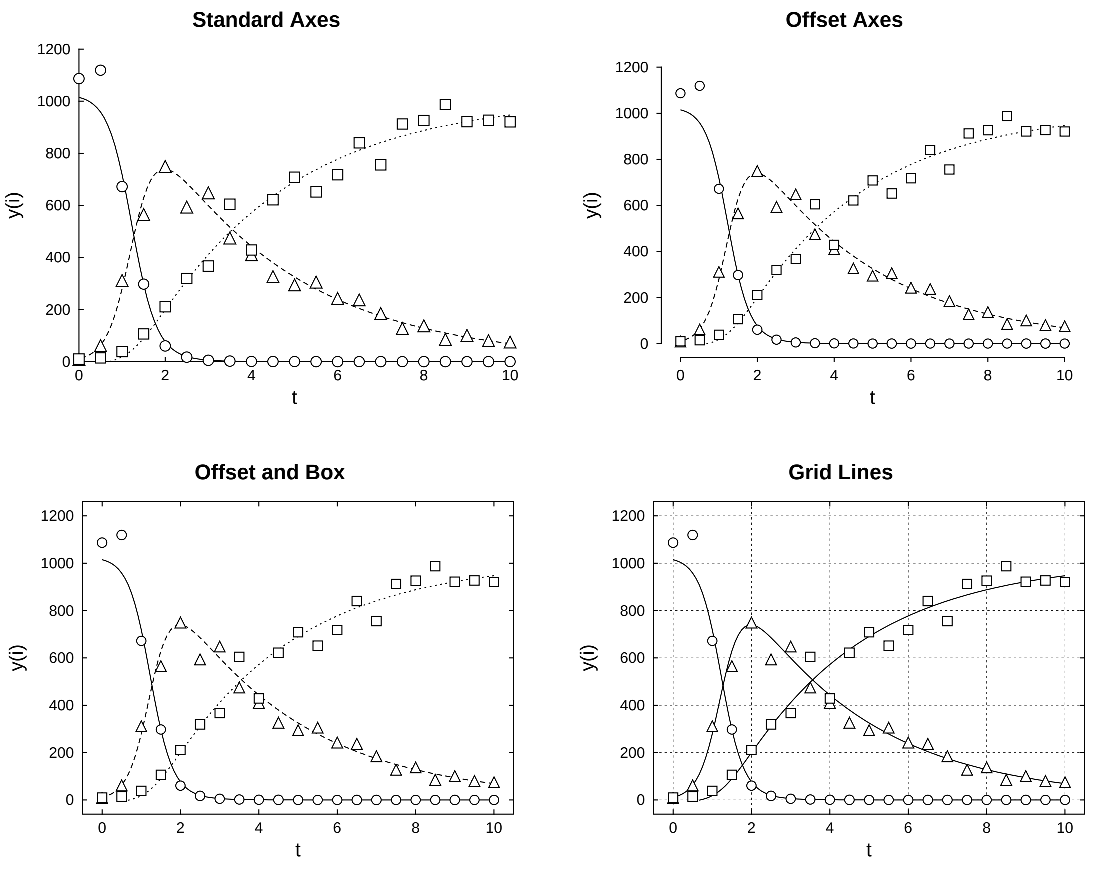


Figure 20.6: Basic plotting styles

pointing inwards coupled with overlaying the data and fitted curve on top of the X axis suggests that offset axes with tick marks pointing outwards (as in the Offset Axes plot of figure 20.6) could be an improvement. Again, some regard a box round the data plotted (as in the Offset and Box plot of figure 20.6) to be visually pleasing, while often grid lines (as in the Grid Lines example of figure 20.6) help to establish coordinates, especially in calibration curves.

SimFIT allows you to choose such default plotting styles from the configuration control which can then be edited at any subsequent stage for temporary changes. However, note that the default style set using the configuration control will always be re-instated each time a new plotting sequence is initiated.

Also, note that alternative notation is often recommended for labeling axes. It could be argued that, in a purely mathematical presentation, using t for time as in figure 20.6 is perfectly acceptable. However, if units need to be given, some would use labeling such as *Time (Days)* while purists could argue that this is ambiguous and that the numerical notation should be dimensionless so they might prefer a label such as *Time/Days*.

20.1.8 Example of plotting without data: Venn diagram

It is possible to use program **simplot** as a generalized diagram drawing program without any data points, as illustrated in figure 20.7.

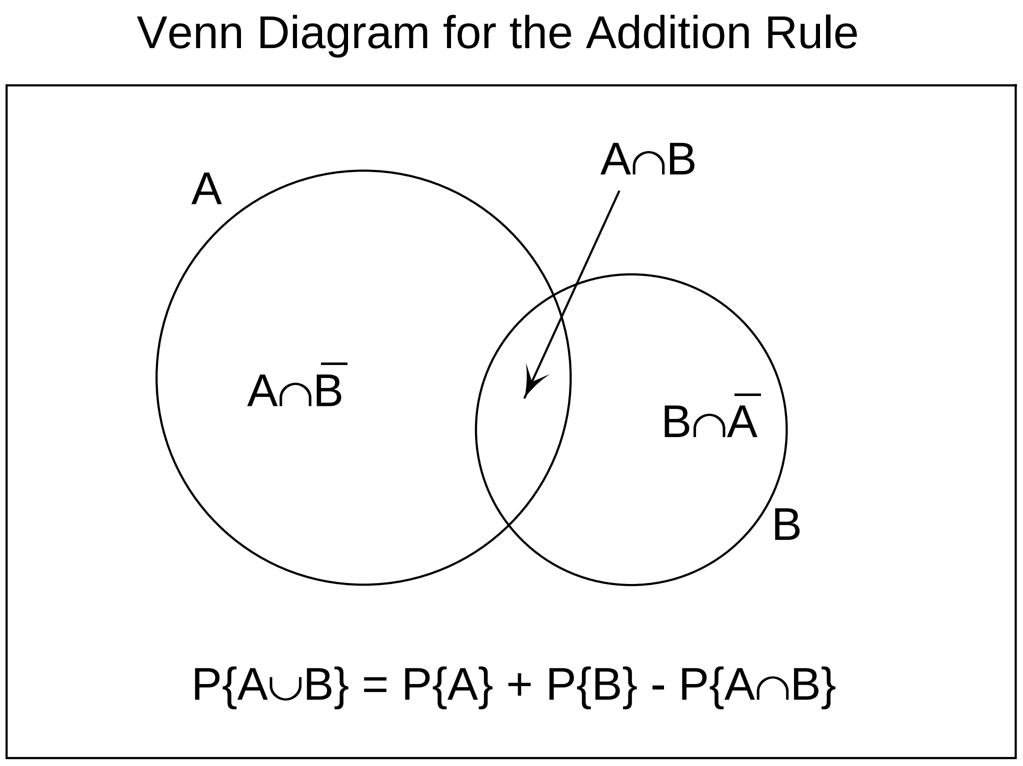

Venn Diagram for the Addition Rule
A
A∩B
A∩B̄
B∩Ā
B
P{A∪B} = P{A} + P{B} - P{A∩B}

Figure 20.7: Venn diagrams

The procedure used to create such graphs using any of the SIMFIT graphical objects will be clear from the details now given for this particular Venn diagram.

- ❍ Program **simplot** was opened using an arbitrary dummy graphics coordinate file.
- ❍ The [Data] option was selected and it was made sure that the dummy data would not be plotted as lines or symbols, by suppressing the lines and symbols for this dummy data set.
- ❍ This transformed program **simplot** into an arbitrary diagram creation mode and the display became completely blank, with no title, legends, or axes.
- ❍ The circles were chosen (as objects) to be outline circle symbols, the box was selected (as an arrow-line-box) to be a horizontal transparent box, the text strings were composed (as text objects), and finally the arrow was chosen (as an arrow-line-box) to be a solid script arrow.
- ❍ The diagram was then completed by editing the text strings (in the expert mode) to introduce the mathematical symbols.

20.1.9 Polygons

Program **simplot** allows filled polygons as an optional linetype. So this means that any set of n coordinates (x_i, y_i) can be joined up sequentially to form a polygon, which can be empty if a normal line is selected, or filled with a chosen color if the filled polygon option is selected. If the last (x_n, y_n) coordinate pair is not the same as the first (x_1, y_1), the polygon will be closed automatically. This technique allows the creation of arbitrary plotting objects of any shape, as will be evident from the sawtooth plot and stars in figure 20.8.

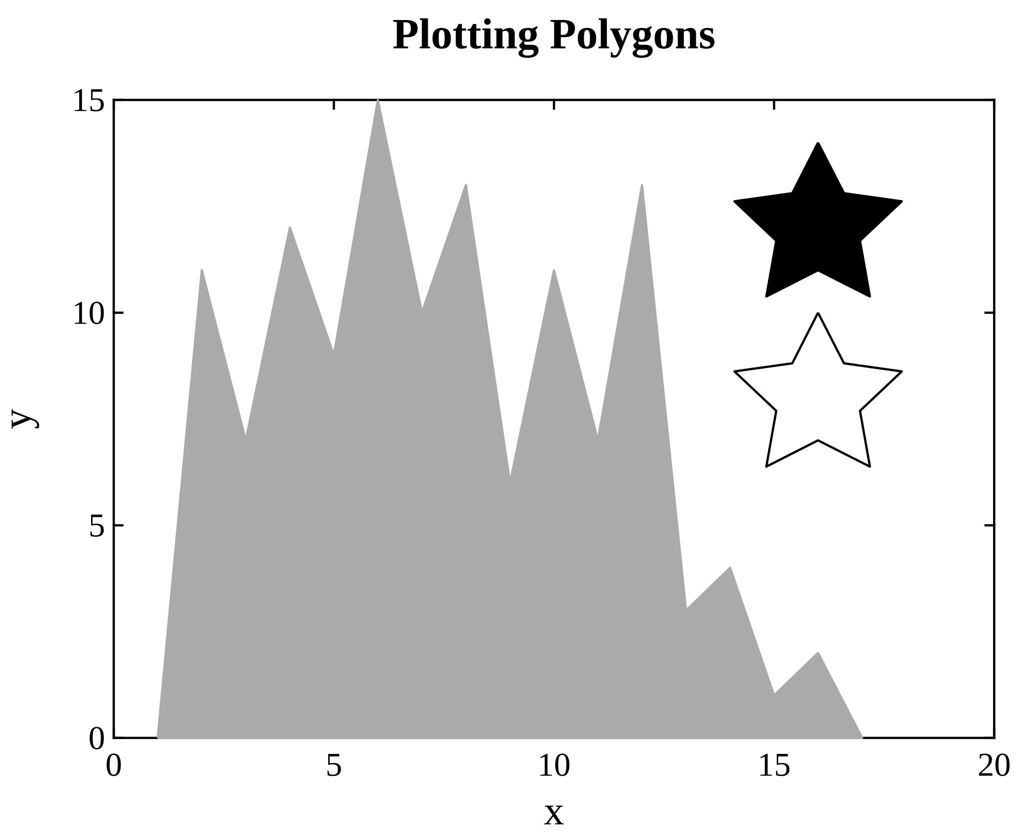


Figure 20.8: Polygons

The sawtooth graph above was generated from a set of (x, y) points in the usual way, by suppressing the plotting symbol but then requesting a filled polygon linetype, colored light gray. The open star was generated from coordinates that formed a closed set, but then suppressing the plotting symbol and requesting a normal, i.e. solid linetype. The filled star was created from a similar set, but selecting a filled polygon linetype, colored black.

If you create a set of ASCII text plotting coordinates files containing arbitrary polygons, such as logos or special plotting symbols, these can be added to any graph. However, since the files will simply be sets of coordinates, the position and aspect ratio of the resulting objects plotted on your graph will be determined by the ranges you have chosen for the x and y axes, and the aspect ratio chosen for the plot. Clearly, objects created in this way cannot be dragged and dropped or re-scaled interactively. The general rule is that the axes, title, plot legends, and displayed data exist in a space that is determined by the range of data selected for the coordinate axes. However, extra text, symbols, arrows, information panels, etc. occupy a fixed space that does not depend on the magnitude of data plotted. So, selecting an interactive data transformation will alter the position of data dependent structures, but will not move any extra text, lines, or symbols.

20.2 Sizes and shapes

20.2.1 Alternative axes and labels

It is useful to move axes to make plots more meaningful, and it is sometimes necessary to hide labels, as with the plot of $y = x^3$ in figure 20.9, where the second x and third y label are suppressed. The figure also illustrates moving an axis in barcharts with bars above and below a baseline.

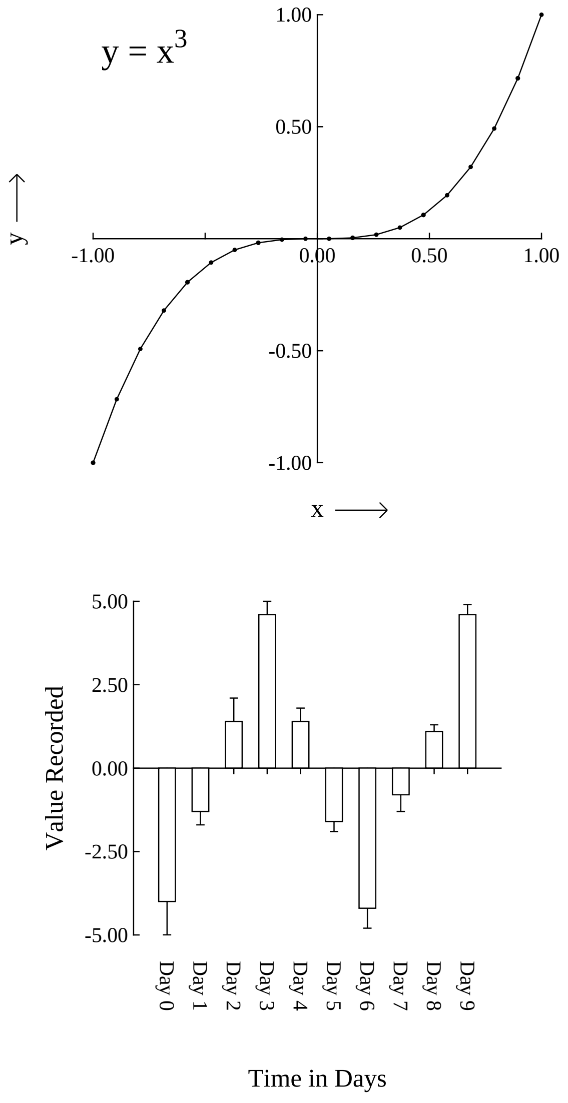


Figure 20.9: Axes and labels

20.2.2 Transformed data

Data should not be transformed before analysis as this distorts the error structure, but there is much to be said for viewing data with error bars and best fit curves in alternative spaces, and program **simplot** transforms automatically as in figure 20.10.

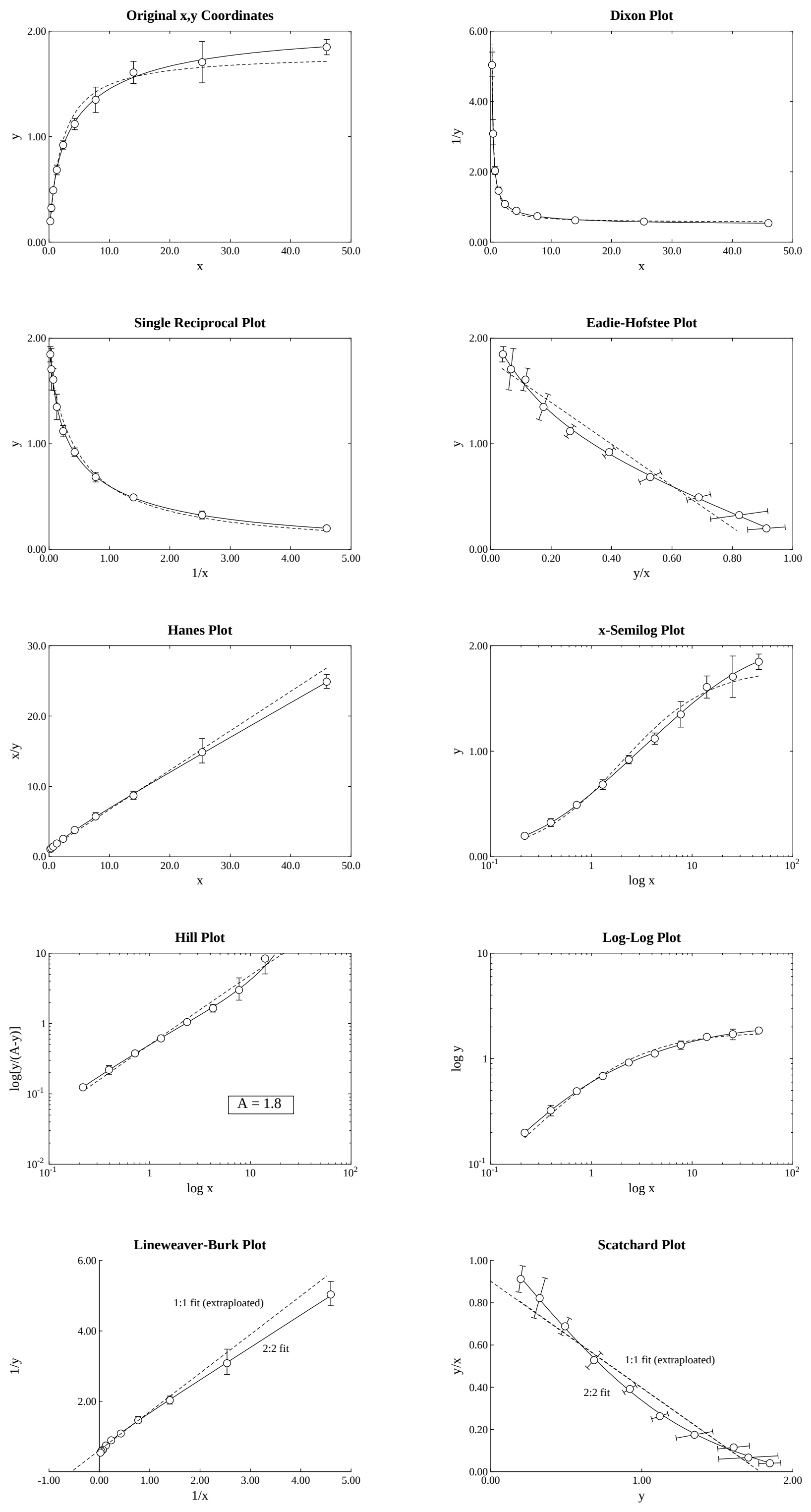


Figure 20.10: Plotting transformed data

20.2.3 Alternative sizes, shapes and clipping

Plots can have horizontal, square or vertical format as in figure 20.11, and user-defined clipping schemes can be used. After clipping, SIMFIT adds a standard BoundingBox so all plots with the same clipping scheme will have the same absolute size but, when GSview transforms ps into eps, it clips individual files to the boundary of white space and the desirable property of equal dimensions will be lost.

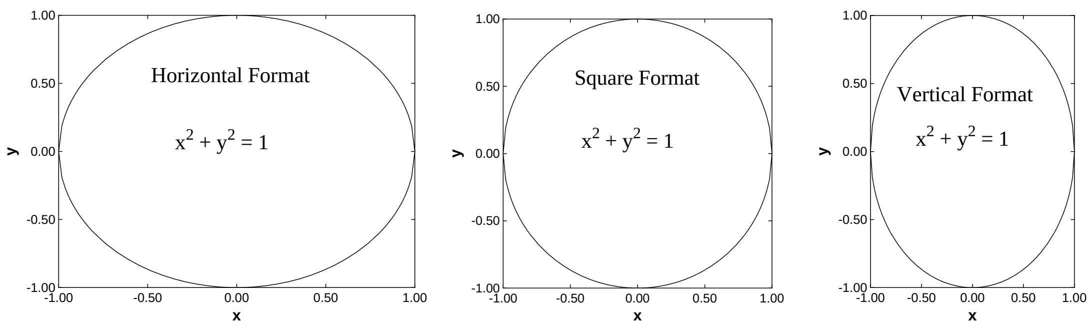


Figure 20.11: Sizes, shapes and clipping.

20.2.4 Rotated and re-scaled graphs

PostScript files can be read into **editps** which has options for re-sizing, re-scaling, editing, rotating, making collages, etc. (see page 364. In figure 20.12 the box and whisker plot was turned on its side to generate a side-on barchart. To do this sort of thing you should learn how to browse a SIMFIT PostScript file in the SIMFIT viewer to read BoundingBox coordinates, in PostScript units of 72 to one inch, and calculate how much to translate, scale, rotate, etc. PostScript users should be warned that the special structure of SIMFIT

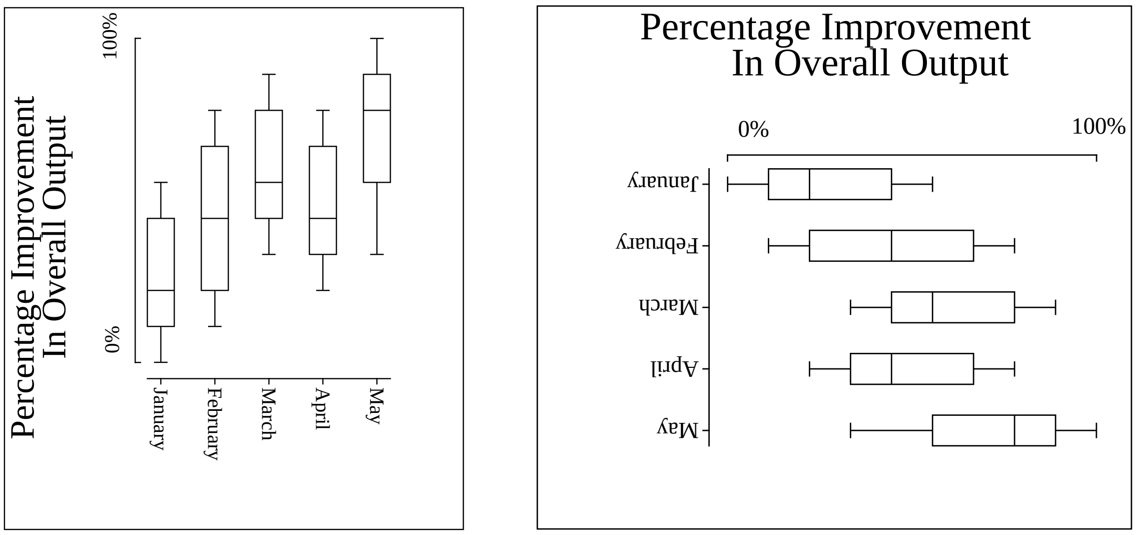


Figure 20.12: Rotating and re-scaling

PostScript files that allows extensive retrospective editing using **editps**, or more easily if you know how using a simple text editor like **notepad**, is lost if you read such graphs into a graphics editor program like Adobe Illustrator. Such programs start off by redrawing vector graphics files into their own conventions which are only machine readable.

20.2.5 Changed aspect ratios and shear transformations

The barchart in figure 20.13 below was scaled to make the X-axis longer than the Y-axis and vice-versa, but note how this type of differential scaling changes the aspect ratio as illustrated. Since rotation and scaling do not commute, the effect created depends on the order of concatenation of the transformation matrices. For instance, scaling then rotation cause shearing which can be used to generate 3-dimensional perspective effects as in the last sub-figure (see page 364).

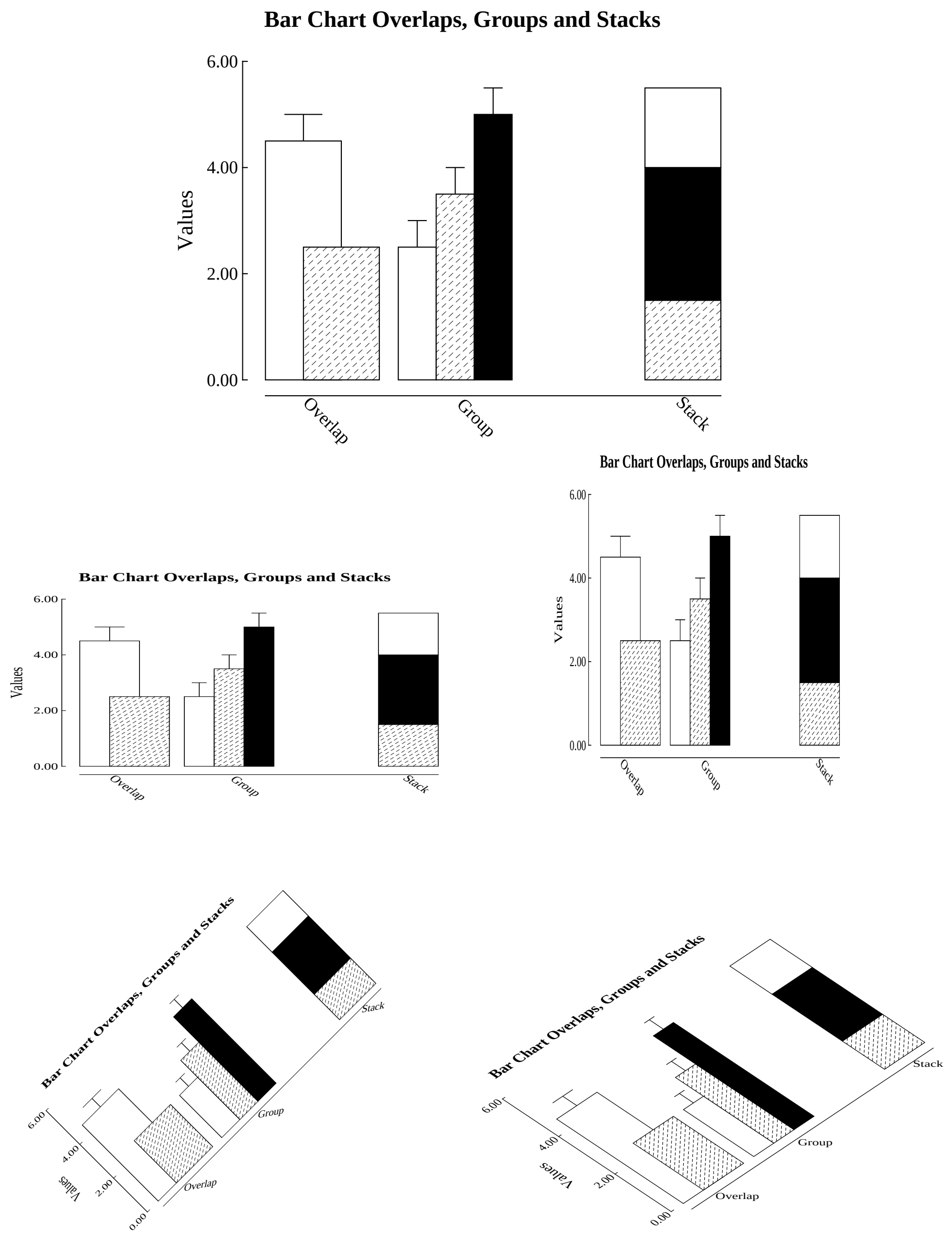


Figure 20.13: Aspect ratios and shearing effects

20.2.6 Reduced or enlarged graphs

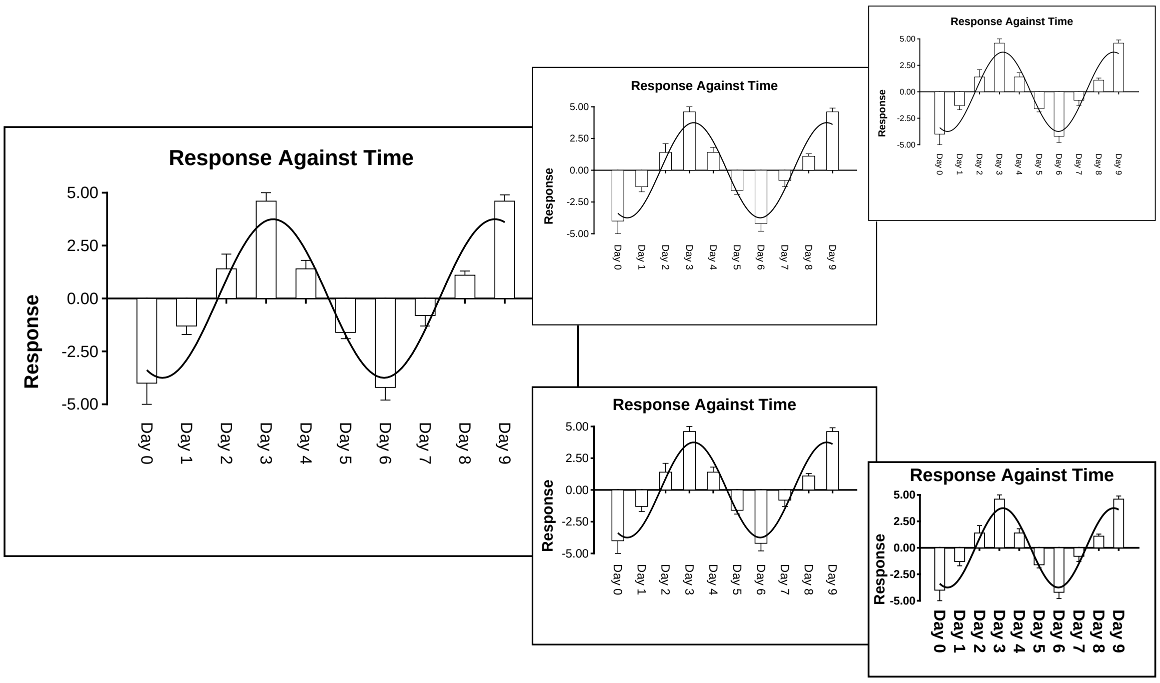


Figure 20.14: Resizing fonts

It is always valuable to be able to edit a graph retrospectively, to change line or symbol types, eliminate unwanted data, suppress error bars, change the title, and so on. SIMFIT PostScript files are designed for just this sort of thing, and a typical example would be altering line widths and font sizes as a figure is re-sized (see page 364). In figure 20.14 the upper sub-figures are derived from the large figure by reduction, so the text becomes progressively more difficult to read as the figures scale down. In the lower sub-figures, however, line thicknesses and font sizes have been increased as the figure is reduced, maintaining legibility. Such editing can be done interactively, but SIMFIT PostScript files are designed to make such retrospective editing easy as described in the `w_readme.*` files and now summarized.

- Line thickness: Changing 11.00 setlinewidth to 22 setlinewidth doubles, while, e.g. 5.5 setlinewidth halves all line thicknesses, etc. Relative thicknesses are set by **simplot**.
- Fonts: Times-Roman, Times-Bold, Helvetica, Helvetica-Bold (set by **simplot**), or, in fact, any of the fonts installed on your printer.
- Texts: ti(title), xl(x legend), yl(y legend), tc(centered for x axis numbers), tl(left to right), tr(right to left), td(rotated down), ty(centered for y axis numbers).
- Lines: pl(polyline), li(line), da(dashed line), do(dotted line), dd(dashed dotted).
- Symbols: ce(i.e. circle-empty), ch(circle-half- filled), cf(circle-filled), and similarly for triangles(te, th, tf), squares(se, sh, sf) and diamonds(de, dh, df). Coordinates and sizes are next to the abbreviations to move, enlarge, etc.

If files do not print after editing you have probably added text to a string without padding out the key. Find the fault using program **GSview** then try again.

20.2.7 Split axes

Sometimes split axes can show data in a more illuminating manner as in figure 20.15. The options are to delete the zero time point and use a log scale to compress the sparse asymptotic section, or to cut out the uninformative part of the best fit curve between 10 and 30 days.

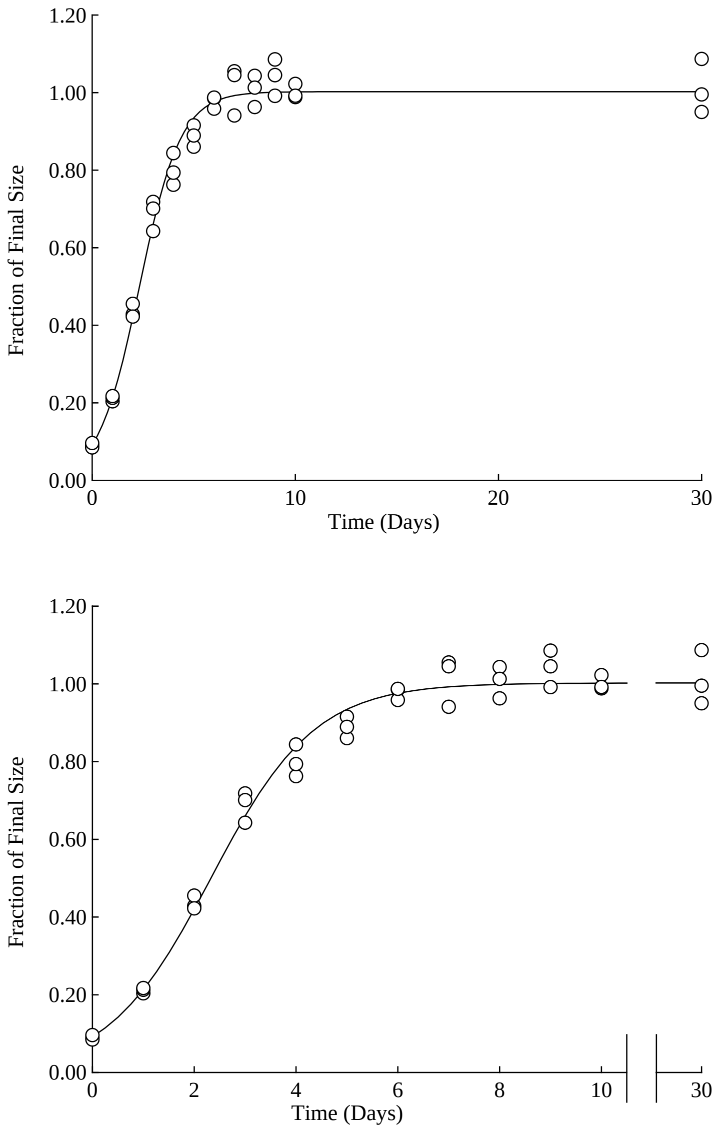


Figure 20.15: Split axes

Windows users can do such things with enhanced metafiles (`*.emf`), but there is a particularly powerful way for PostScript users to split SimFIT graphs in this way. When the SimFIT PostScript file is being created there is a menu selectable shape option that allows users to chop out, re-scale, and clip arbitrary pieces of graphs, but in such a way that the absolute position, aspect ratio, and size of text strings does not change. In this way a master graph can be decomposed into any number of appropriately scaled slave sub-graphs. Then **editps** can be used to compose a graph consisting of the sub-graphs rearranged, repositioned, and resized in any configuration (see page 364). Figure 20.15 was created in this way after first adding the extra lines shown at the splitting point.

20.2.8 Stepping over intermediate data points

Sometimes it is advantageous to step over intermediate data points and, for clarity, only plot points at intervals through the data set. For instance, there may be a large number of machine generated data points, far more than is required to define a curve by the usual technique of joining joining up successive points by straight lines. Plotting all the points in such cases would slow down the graphics display and, more importantly, could lead to very large PostScript output files. Again, there would be times when a continuous curve is required to illustrate a function defined by a reasonable number of closely spaced data points, but symbols at all points would obscure the curve, or might only be required for labeling purposes. Figure 20.16 illustrates a case in point.

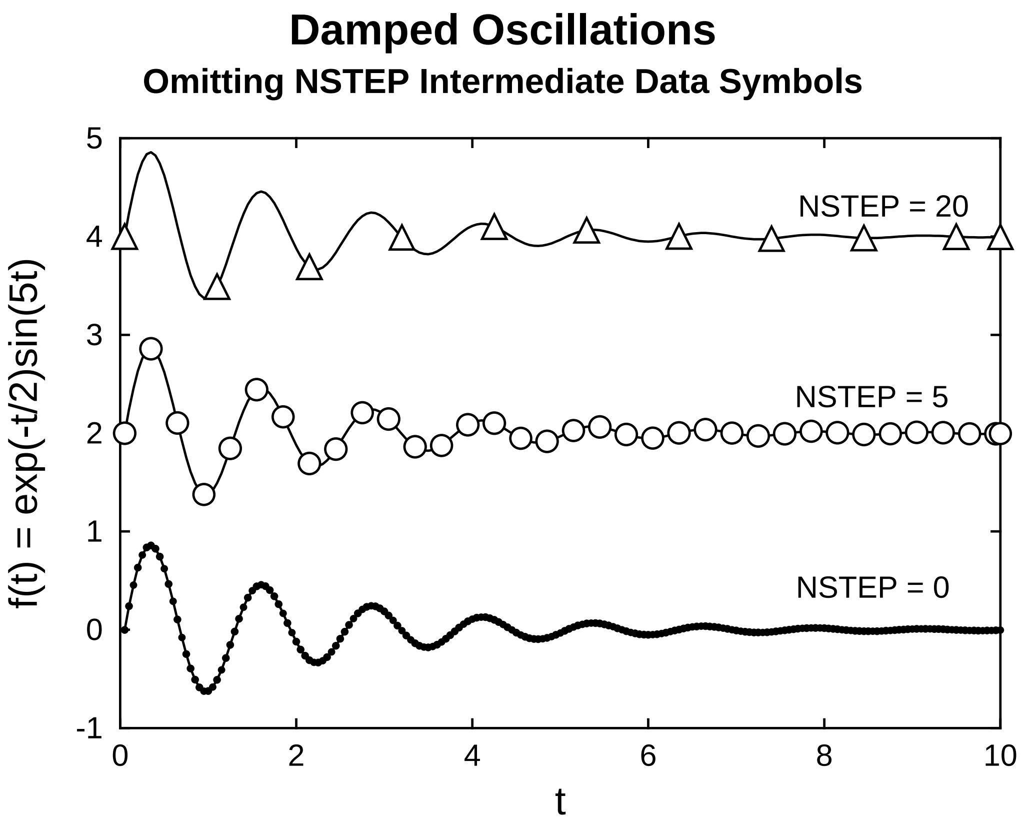


Figure 20.16: Stepping over intermediate points

Here even small symbols, like dots in the bottom figure where no points are omitted, would obscure the plot and the middle plot with intermediate groups of five suppressed, could also be regarded as too crowded, while the upper plot has sufficient symbols even with twenty points stepped over to identify the plot, say in an information panel.

Of course such graphs can easily be created by pre-processing the data files before submitting for plotting. However, SIMFIT has a special facility to do this interactively. The technique required to create plots like figure 20.16 is to submit two files with identical data, one to be plotted as a line joining up data to create the impression of a continuous curve, the other to be plotted as symbols. Then, from the [Data] menu when the plot is displayed, a parameter *NSTEP* can be defined, where *NSTEP* is the number of intermediate points in each group to be stepped over. Observe that, when using this technique, the first and last data points are always plotted to avoid misunderstanding.

20.3 Equations

20.3.1 Maths

You can add equations to graphs directly, but this will be a compromise, as specialized type setting techniques are required to display maths correctly. The LaTeX system is pre-eminent in the field of maths type-setting and the PSfrag system, as revised by David Carlisle and others, provides a simple way to add equations to SɪᴍFɪT graphs. For figure 20.17, **makdat** generated a Normal cdf with $\mu = 0$ and $\sigma = 1$, then **simplot** created `cdf.eps` with the key `phi(x)`, which was then used by this stand-alone code to generate the figure, where the equation substitutes for the key. LaTeX PostScript users should be aware that SɪᴍFɪT PostScript file format has been specially designed to be consistent with the PSfrag package but, if you want to then use **GhostScript** to create graphics file, say .png from .eps, the next section should be consulted.

```
\documentclass[dvips,12pt]{article}
   \usepackage{graphicx}
   \usepackage{psfrag}
   \pagestyle{empty}
\begin{document}
\large
\psfrag{phi(x)}{$\displaystyle
   \frac{1}{\sigma \sqrt{2\pi}}
   \int_{-\infty}^x \exp\left\{
  -\frac{1}{2} \left( \frac{t-\mu}{\sigma} \right)^2 \right\}\,dt$}
\mbox{\includegraphics[width=6.0in]{cdf.eps}}
\end{document}
```

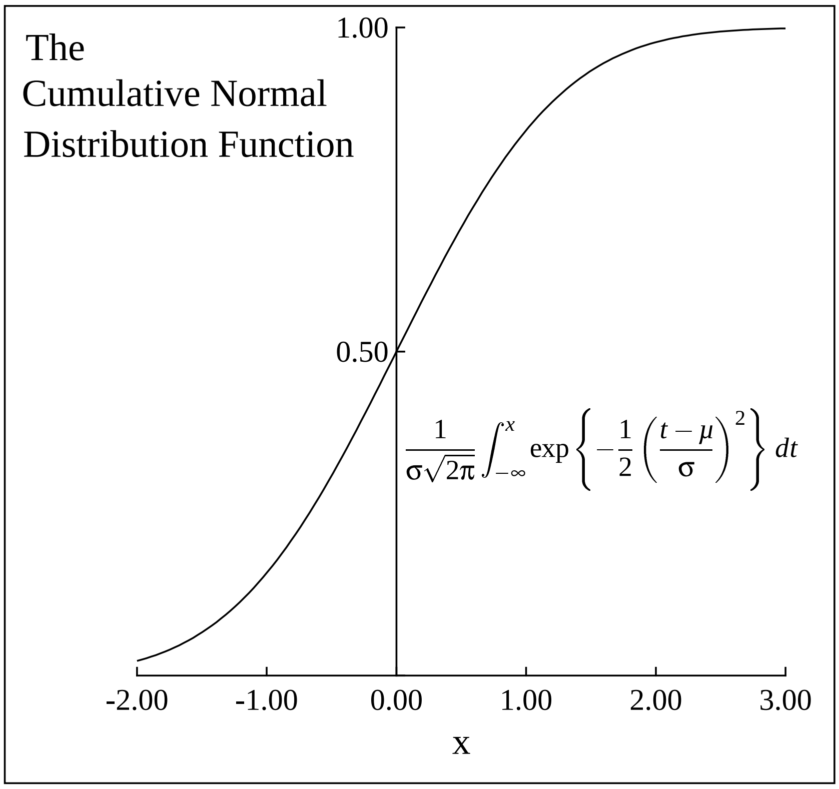


Figure 20.17: Plotting mathematical equations

20.3.2 Chemical Formulæ

LaTeX code, as below, is intended for document preparation and adds white space to the final .ps file. The easiest way round this complication is to add an outline box to the plot, as in figure **??**. Then, after the .png file has been created, it can be input into, e.g., GIMP, for auto clipping to remove extraneous white space, followed by deletion of the outline box if required.

```
\documentclass[dvips,12pt]{article}
   \usepackage{graphicx}
   \usepackage{psfrag}
   \usepackage{carom}
   \pagestyle{empty}
\begin{document}
\psfrag{formula}
{\begin{picture}(3000,600)(0,0)
\thicklines
\put(0,0){\bzdrv{1==CH$_{2}$NH$_{2}$;4==CH$_{2}$N(Me)$_{2}$}}
\put(700,450){\vector(1,0){400}}
\put(820,550){[O]}
\put(1000,0){\bzdrv{1==CHO;4==CH$_{2}$N(Me)$_{2}$}}
\put(1650,400){+}
\put(1750,400){NH$_{3}$}
\put(2000,450){\vector(1,0){400}}
\put(2120,550){[O]}
\put(2300,0){\bzdrv{1==CO$_{2}$H;4==CH$_{2}$N(Me)$_{2}$}}
\end{picture}}
\mbox{\includegraphics{chemistry.eps}}
\end{document}
```

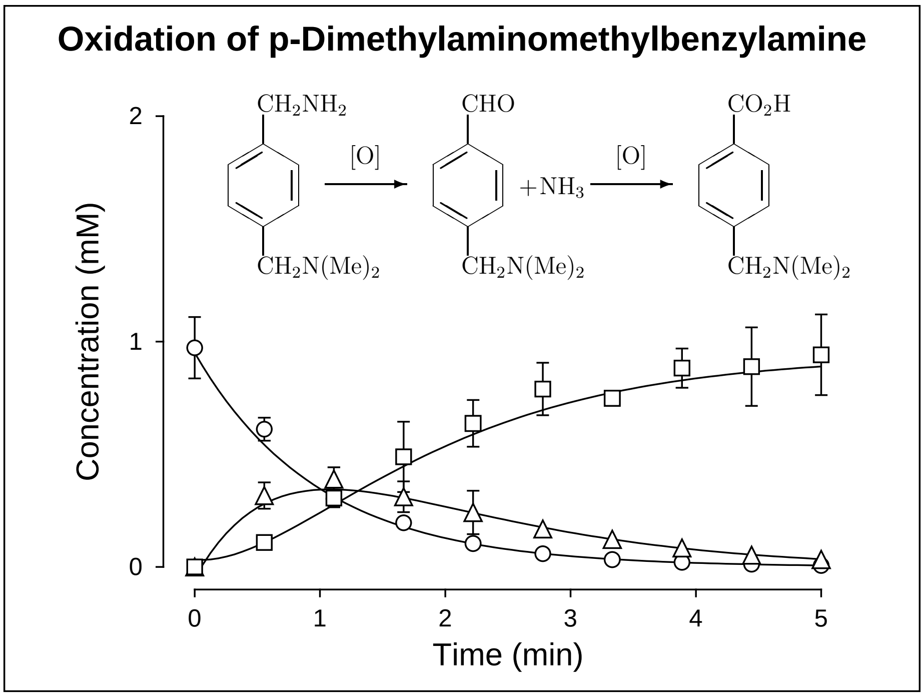
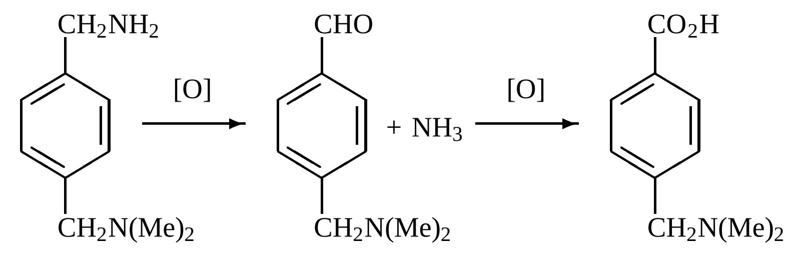


Figure 20.18: Plotting chemical structures

20.3.3 Composite graphs

The technique used to combine sub-graphs into a composite graph is easy (see page 364). First use your drawing or painting program to save the figures of interest in the form of eps files. Then the SimFIT graphs and any component eps files are read into **editps** to move them and scale them until the desired effect is achieved. In figure 20.19, data were generated using **deqsol**, error was added using **adderr**, the simulated experimental data were fitted using **deqsol**, the plot was made using **simplot**, the chemical formulae and mathematical equations were generated using LaTeX and the final graph was composed using **editps**.

A kinetic study of the oxidation of *p*-Dimethylaminomethylbenzylamine



$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -k_{+1} & k_{-1} & 0 \\ k_{+1} & (-k_{-1}-k_{+2}) & k_{-2} \\ 0 & k_{+2} & -k_{-2} \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix},\ \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

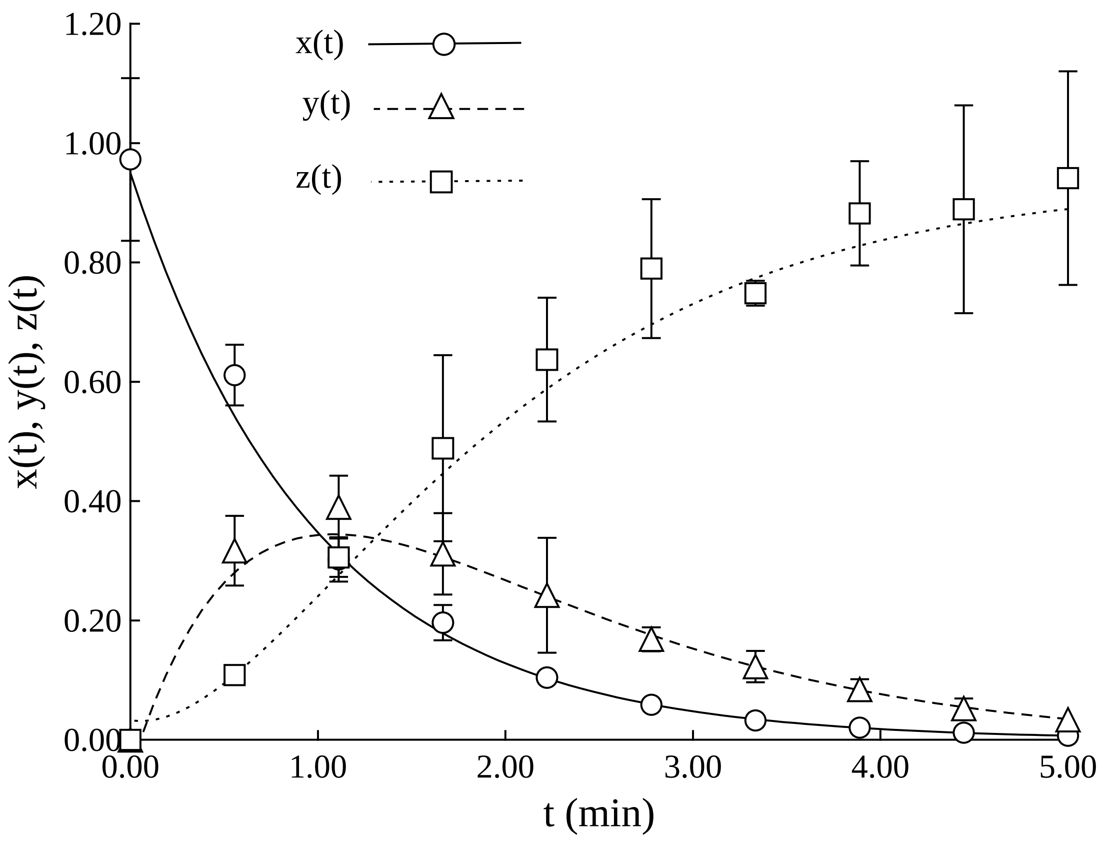
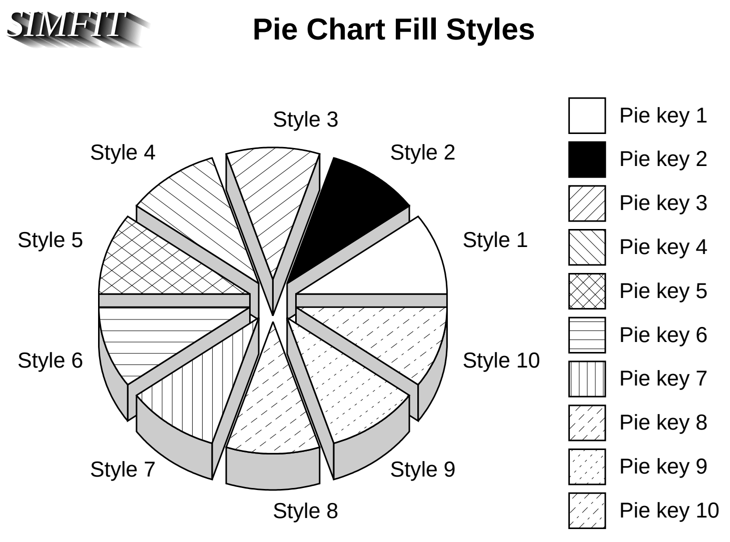


Figure 20.19: Chemical formulas



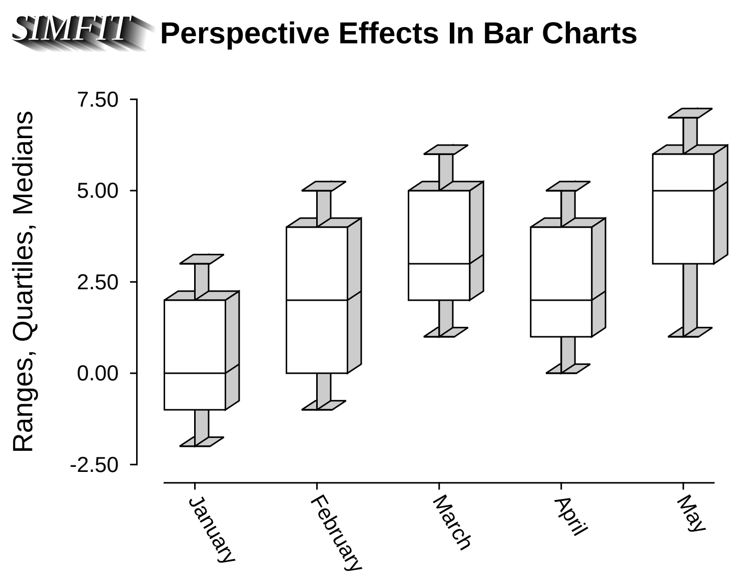


Figure 20.20: Perspective in barcharts, box and whisker plots and piecharts

20.4 Bar charts and pie charts

20.4.1 Perspective effects

Perspective can be useful in presentation graphics but it must be realized that, when pie chart segments are moved centrifugally, spaces adjacent to large segments open more than those adjacent to small sections. This creates an impression of asymmetry to the casual observer, but it is geometrically correct. Again, diminished curvature compared to the perimeter as segments move out becomes more noticeable where adjacent segments have greatly differing sizes so, to minimize this effect, displacements can be adjusted individually. A PostScript special (page 381) has been used to display the logo in figure 20.20.

20.4.2 Advanced barcharts

SimFIT can plot barcharts directly from data matrices, using the exhaustive analysis of a matrix procedure in **simstat**, but there is also an advanced barchart file format which gives users complete control over every individual bar, etc. as now summarized and illustrated in figure 20.21.

- Individual bars can have arbitrary position, width, fill style and color.
- Bars can be overlapped, grouped, formed into hanging groups or stacked vertically.
- Error bars can be capped or straight, and above or below symbols.
- Extra features such as curves, arrows, panel identifiers or extra text can be added.

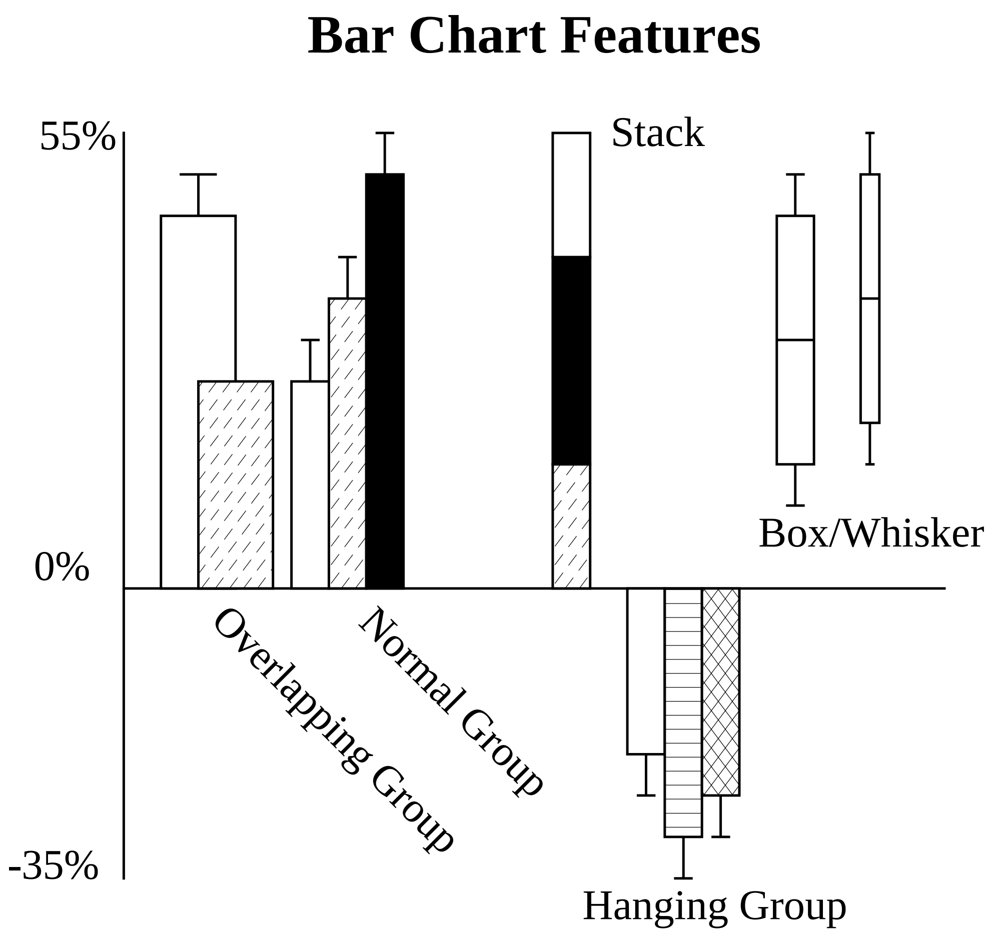


Figure 20.21: Advanced bar chart features

Of course the price that must be paid for such versatility is that the file format is rather complicated and the best way to understand it is to consult the `w_readme` files for details, browse the test files `barchart.tf?`, then read them into **simplot** to observe the effect produced before trying to make your own files. Labels can be added automatically, appended to the data file, edited interactively, or input as simple text files.

20.4.3 Three dimensional barcharts

The SimFIT surface plotting function can be used to plot three dimensional bars as, for example, using the test file `barcht3d.tf1` to generate figure 20.22. Blank rows and shading can also be added to enhance the three dimensional effect.

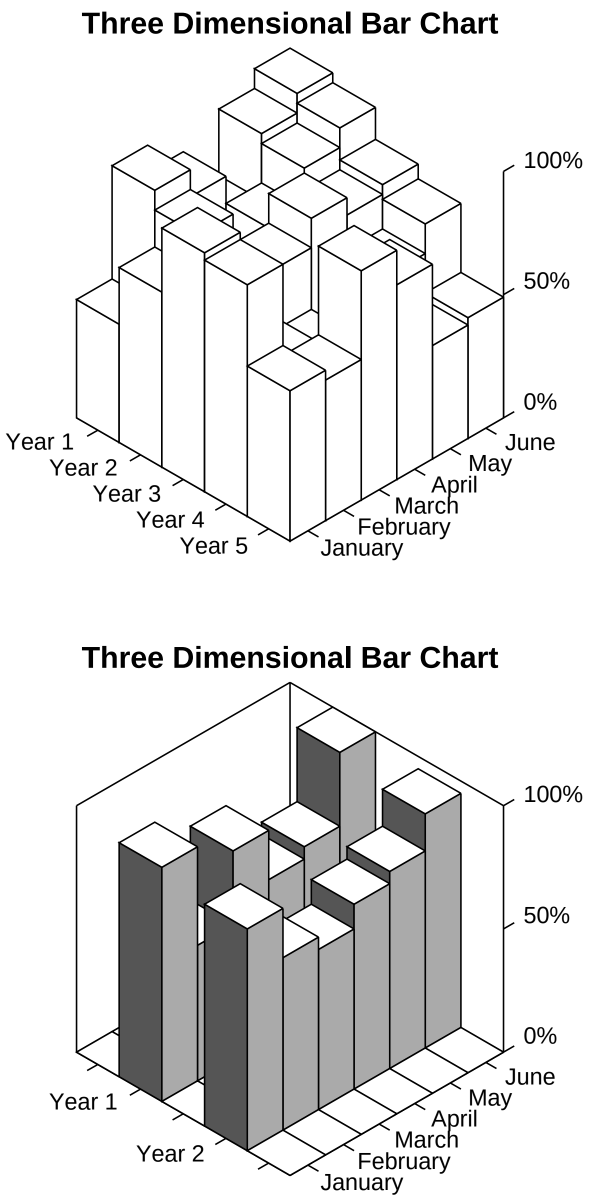


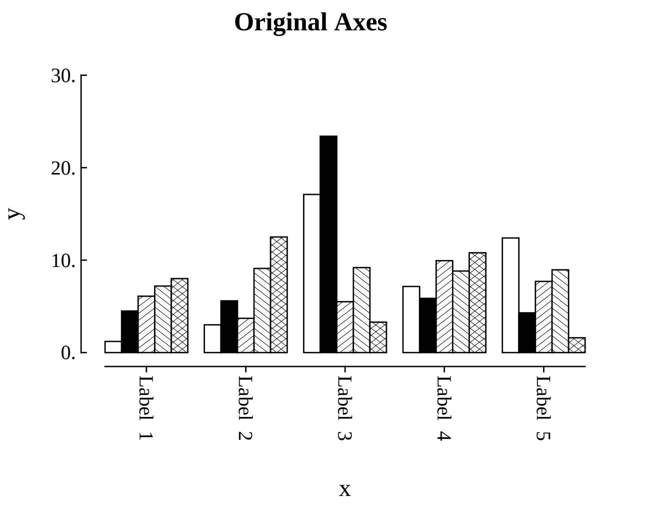
Figure 20.22: Three dimensional barcharts

Such plots can be created from n by m matrix files, or special vector files, e.g. with n values for x and m values for y a $nm + 6$ vector is required with n, then m, then the range of x and range of y, say $(0, 1)$ and $(0, 1)$ if arbitrary, followed by values of $f(x, y)$ in order of increasing x at consecutive increasing values of y.

20.5 Error bars

20.5.1 Error bars with barcharts

Barcharts can be created interactively from a table of values. For example, figure 20.23 was generated by the exhaustive analysis of a matrix procedure in **simstat** from `matrix.tf1`.



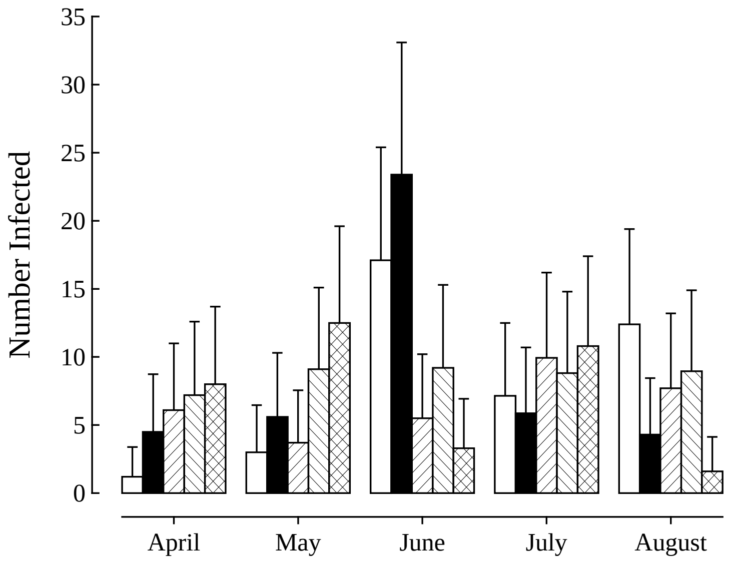
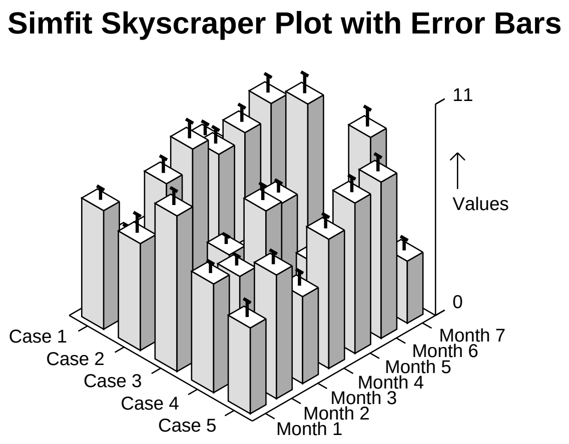


Figure 20.23: Error bars 1: barcharts

If the elements are measurements, the bars would be means, while error bars should be calculated as 95% confidence limits, i.e. assuming a normal distribution. Often one standard error of the mean is used instead of confidence limits to make the data look better, which is dishonest. If the elements are counts, approximate error bars should be added to the matrix file in **simplot** from a separate file, using twice the square root of the counts, i.e. assuming a Poisson distribution. After creating barcharts from matrices, the temporary advanced barchart files can be saved.

20.5.2 Error bars with skyscraper and cylinder plots

Barcharts can be created for tables, $z(i, j)$ say, where cells are values for plotting as a function of x (rows) and y (columns). The x, y values are not required, as such plots usually require labels not numbers. Figure 20.24 shows the plot generated by **simplot** from the test file `matrix.tf2`.



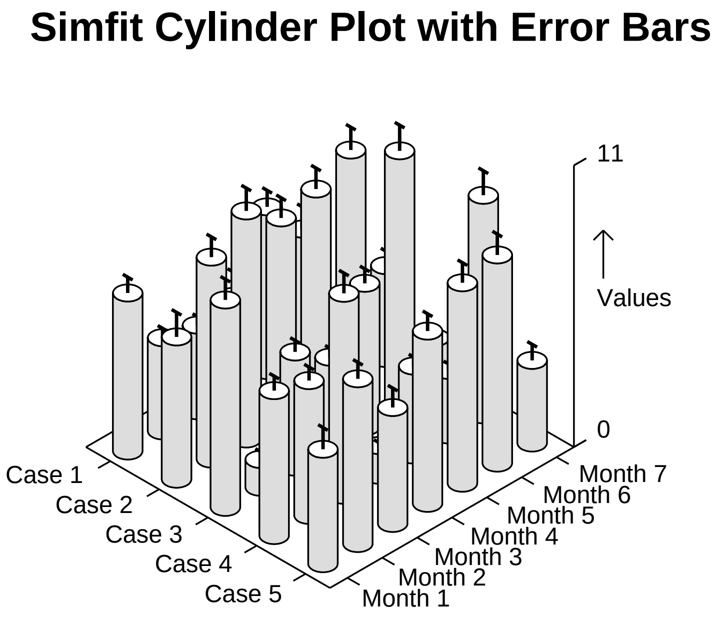


Figure 20.24: Error bars 2: skyscraper and cylinder plots

Errors are added from a file, and are calculated according to the distribution assumed. They could be twice square roots for Poisson counts, binomial errors for proportions or percentages, or they could be calculated from sample standard deviations using the t distribution for means. As skyscraper plots with errors are dominated by vertical lines, error bars are plotted with thickened lines, but a better solution is to plot cylinders instead of skyscrapers, as illustrated.

20.5.3 Slanting and multiple error bars

Error bar files can be created by program **editfl** after editing curve fitting files with all replicates, and such error bars will be symmetrical, representing central confidence limits in the original(x, y) space. But, note that these error bars can become unsymmetrical or slanting as a result of a transformation, e.g. $\log(y)$ or Scatchard, using program **simplot**. Program **binomial** will, on the other hand, always generates noncentral confidence limits, i.e. unsymmetrical error bars for binomial parameter confidence limits, and Log-Odds plots.

However, sometimes it is necessary to plot asymmetrical error bars, slanting error bars or even multiple error bars. To understand this, note that the standard error bar test file `errorbar.tf1` contains four columns with the x coordinate for the plotting symbol, then the y-coordinates for the lower bar, middle bar and upper bar. However, the advanced error bar test file `errorbar.tf2` has six columns, so that the (x_1, y_1), (x_2, y_2), (x_3, y_3) coordinates specified, can create any type of error bar, even multiple error bars, as will be seen in figure 20.25.

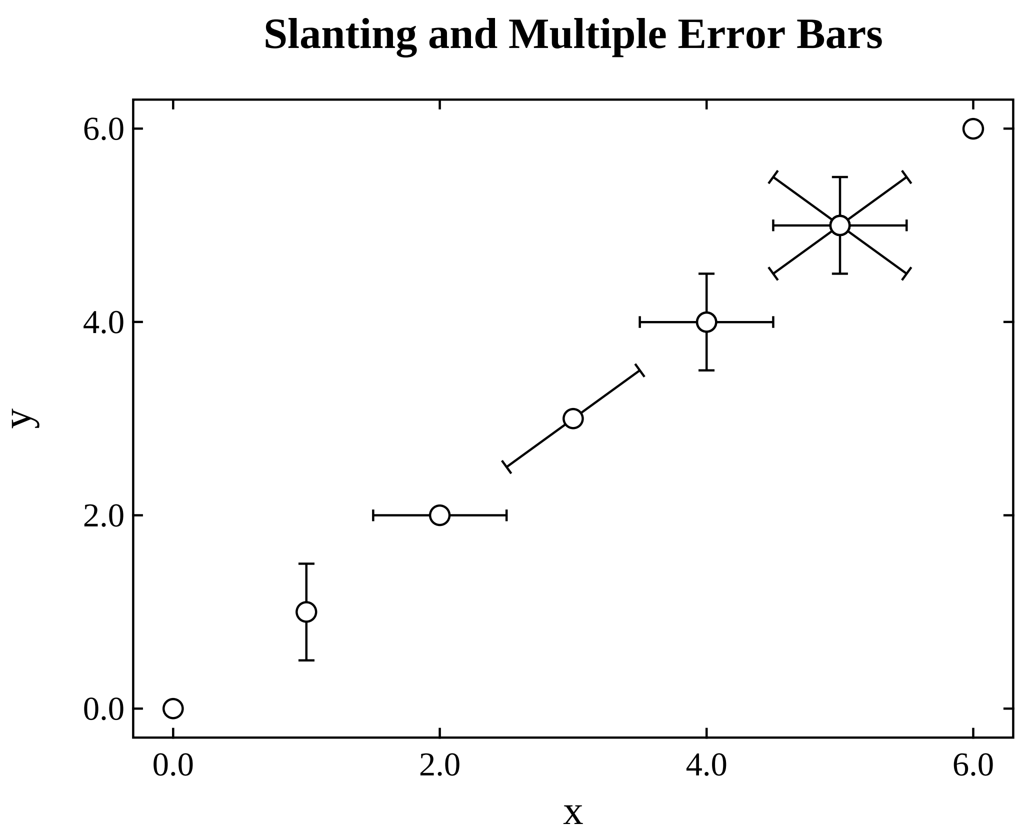
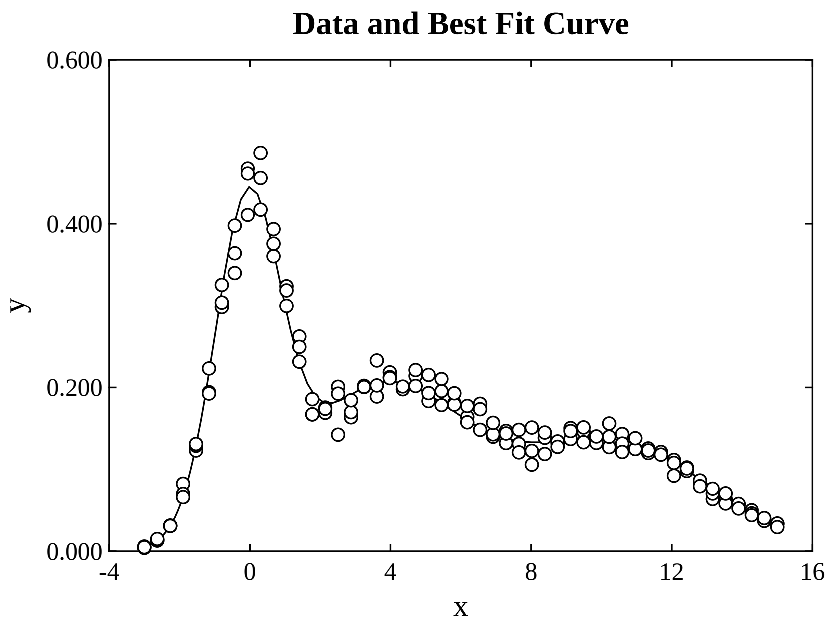


Figure 20.25: Error bars 3: slanting and multiple

Note that the normal error bar files created interactively from replicates by **editfl**, **qnfit**, **simplot**, or **compare** will only have four columns, like `errorbar.tf1`, with x, y_1, y_2, y_3, in that order. The six-column files like `errorbar.tf2` required for multiple, slanting, or unsymmetrical error bars must be created as matrix files with the columns containing $x_1, x_2, x_3, y_1, y_2, y_3$, in that order.

20.5.4 Calculating error bars interactively

Figure 20.26 shows the best fit curve estimated by **qnfit** when fitting a sum of three Gaussians to the test file `gauss3.tf1` using the expert mode. Note that all the data must be used for fitting, not means. **editfl** can generate error bar plotting files from such data files with replicates, but error bars can also be calculated interactively after fitting, as illustrated for 95% confidence limits.



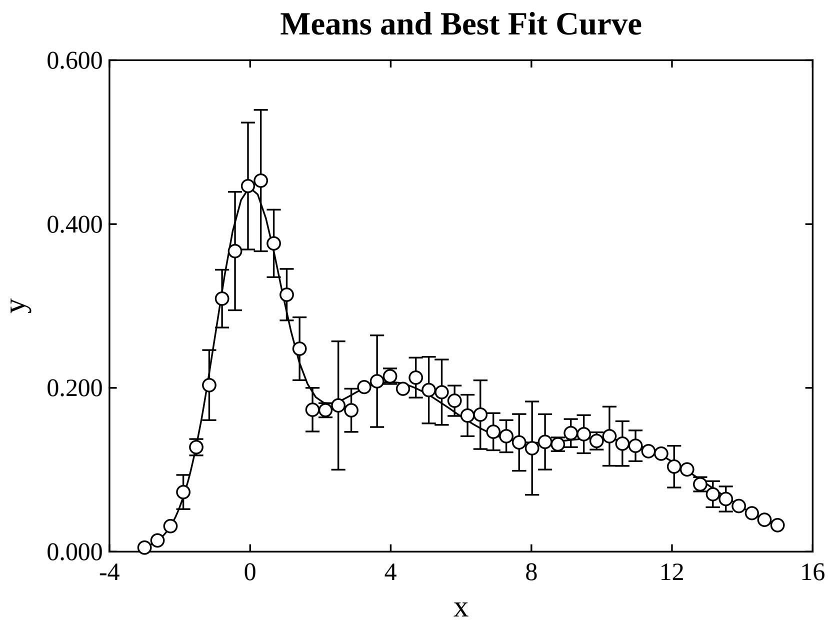


Figure 20.26: Error bars 4: calculated interactively

20.6 Three dimensional plotting

20.6.1 Surfaces and contours

SIMFIT uses isometric projection, but surfaces can be viewed from any corner, and data can be shown as a surface, contours, surface with contours, or 3-D bar chart as in figure 20.27.

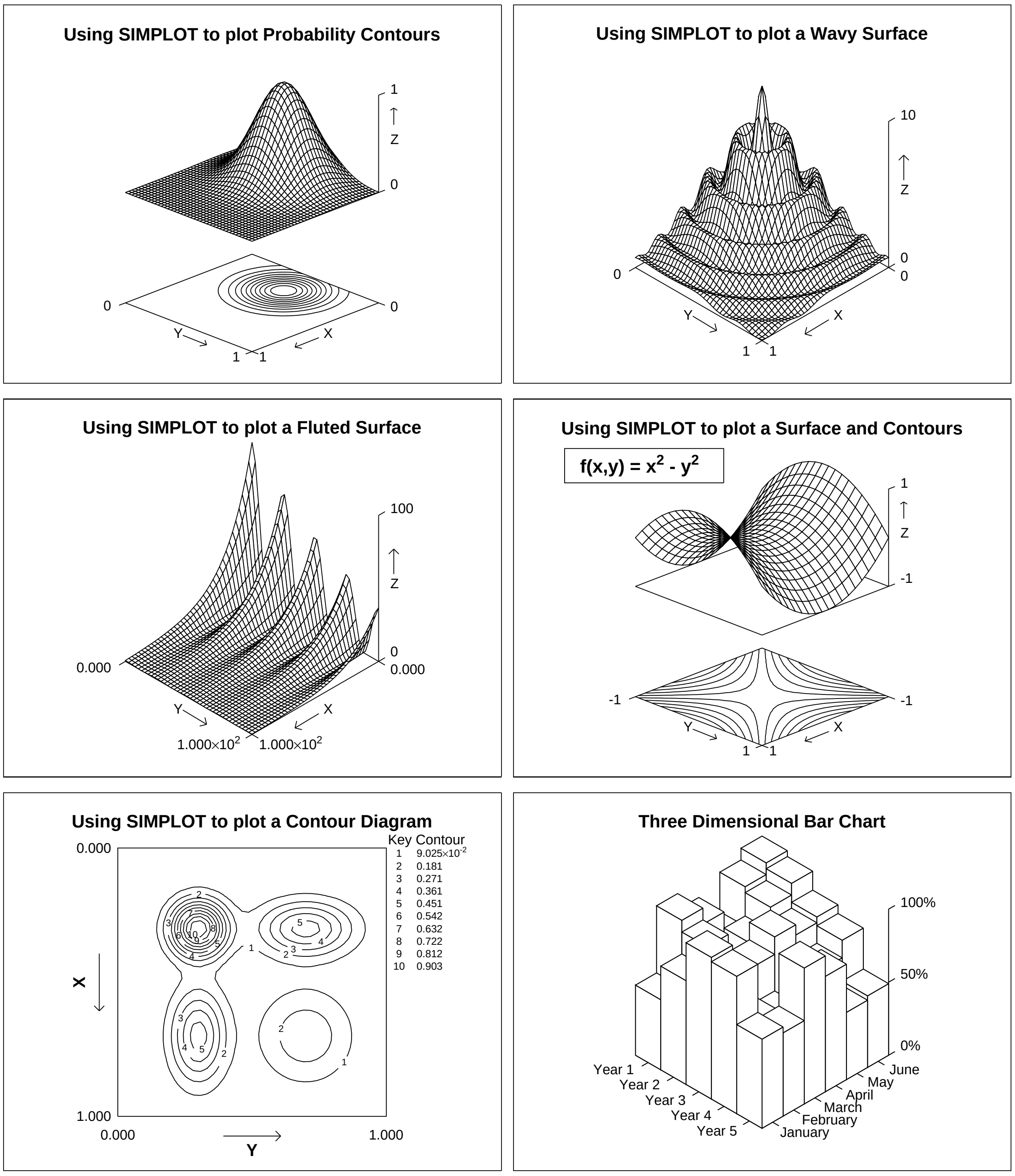


Figure 20.27: Three dimensional plotting

20.6.2 Three dimensional space curves

Sets of x, y, z coordinates can be plotted in three dimensional space to represent either an arbitrary scatter of points, a surface, or a connected space curve. Arbitrary points are best plotted as symbols such as circles or triangles, surfaces are usually represented as a mesh of orthogonal space curves, while single space curves can be displayed as symbols or may be connected by lines. For instance, space curves of the form

$$x = x(t), y = y(t), z = z(t)$$

can be plotted by generating x, y, z data for constant increments of t and joining the points together to create a smooth curve as in figure 20.28.

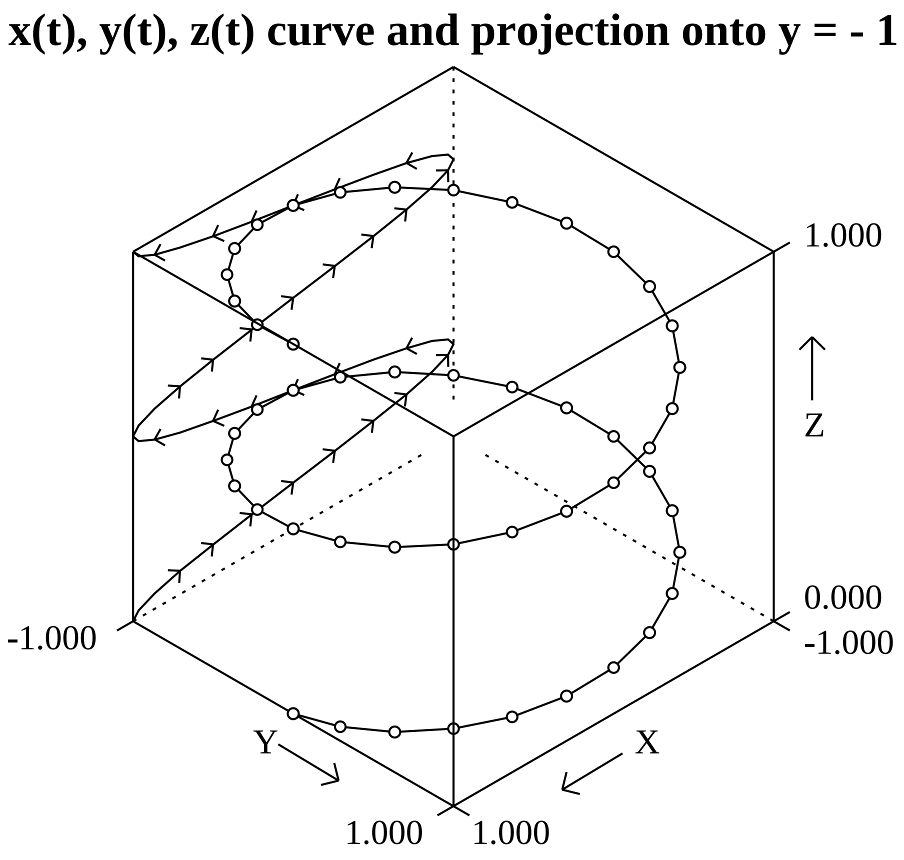


Figure 20.28: Space curves and projections

Such space curves can be generated quite easily by preparing data files with three columns of x, y, z data values, then displaying the data using the space curve option in **simplot**. However users can also generate space curves from $x(t), y(t), z(t)$ equations, using the option to plot parametric equations in **simplot** or **usermod**. The test file `helix.mod` shows you how to do this for a three dimensional helix. Note how the rear (x, y) axes have been subdued and truncated just short of the origin, to improve the three dimensional effect. Also, projections onto planes are generated by setting the chosen variable to a constant, or by writing model files to generate x, y, z data with chosen coordinates equal to the value for the plane.

20.6.3 Projecting space curves onto planes

Sometimes it is useful to project space curves onto planes for purposes of illustration. Figure 20.29 shows a simulation using **usermod** with the model file `twister.mod`. The parametric equations are

$$x = t\cos t, y = t\sin t, z = t^2$$

and projections are created by fixing one the variables to a constant value.

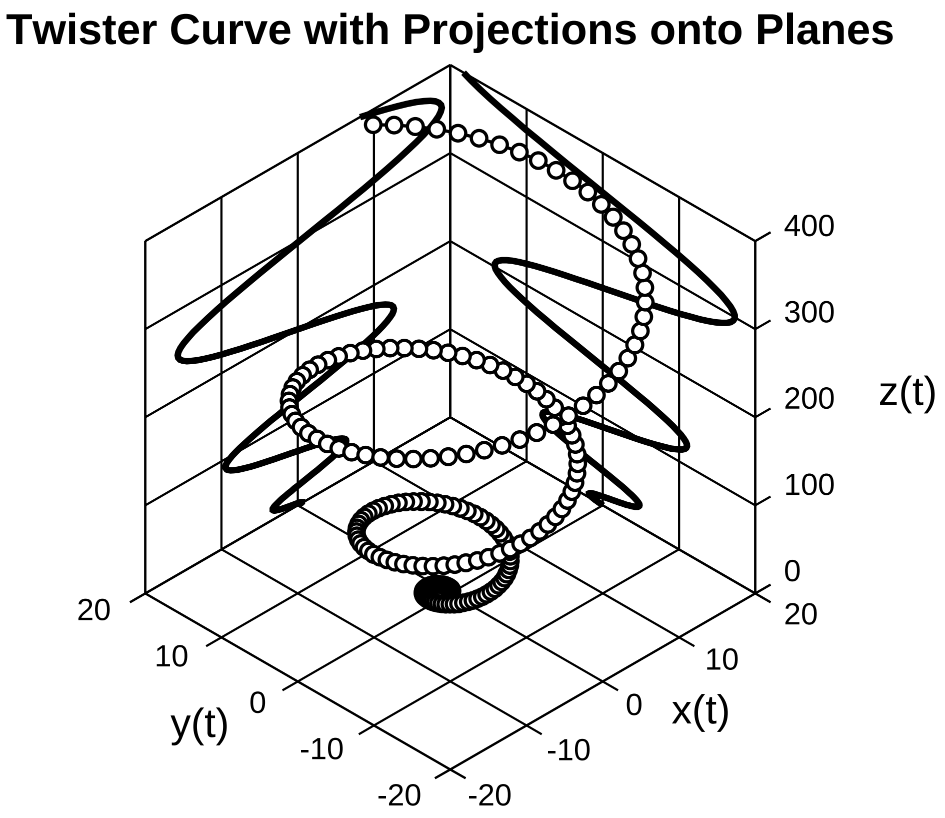


Figure 20.29: Projecting space curves onto planes

Note the following about the model file `twister.mod`.

- There are 3 curves so there are 9 functions of 1 variable
- The value of x supplied is used as the parameter t
- Functions $f(1), f(4), f(7)$ are the $x(t)$ profiles
- Functions $f(2), f(5), f(8)$ are the $y(t)$ profiles
- Functions $f(3), f(6), f(9)$ are the $z(t)$ profiles

Also observe that the model parameters fix the values of the projection planes just outside the data range, at

$$p(1) = 20, p(2) = 20.$$

20.6.4 Three dimensional scatter diagrams

Often it is necessary to plot sets of x, y, z coordinates in three dimensional space where the coordinates are arbitrary and are not functions of a parameter t. This is the case when it is wished to illustrate scattering by using different symbols for subsets of data that form clusters according to some distance criteria. For this type of plotting, the sets of x, y, z triples, say principal components, are collected together as sets of three column matrices, preferably referenced by a library file, and a default graph is first created. The usual aim would be to create a graph looking something like figure 20.30.

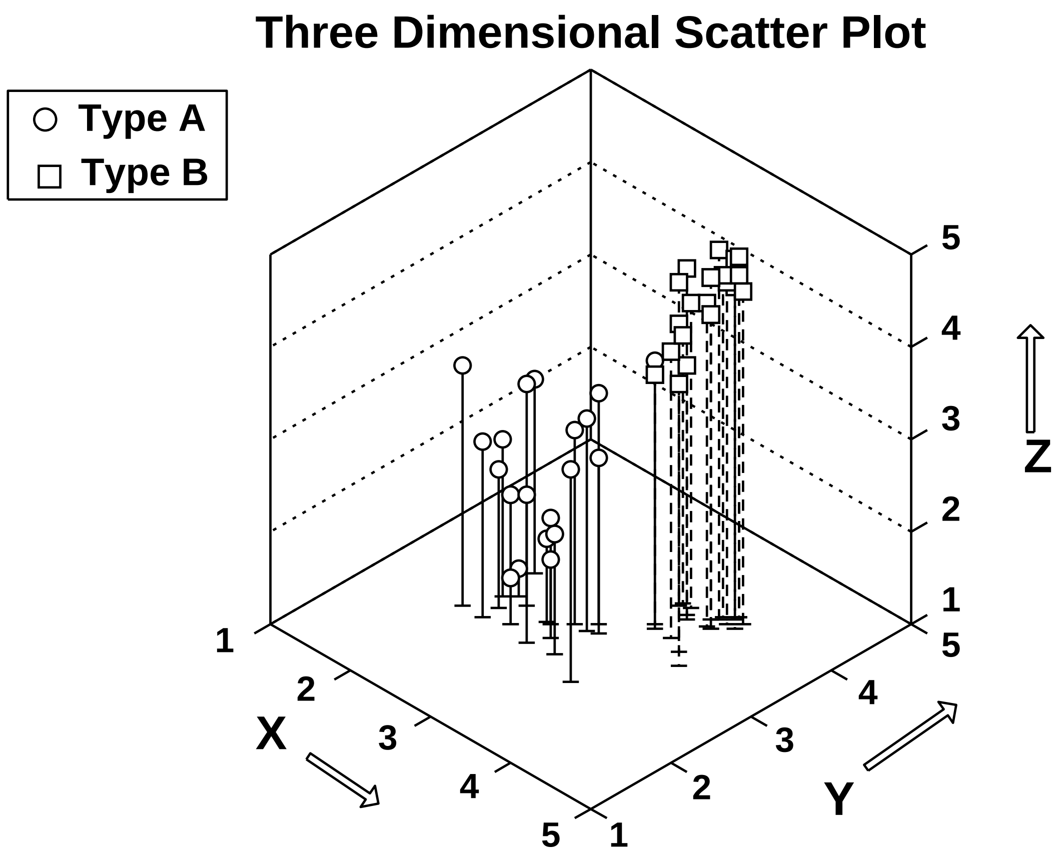


Figure 20.30: Three dimensional scatter plot

In this graph, the front axes have been removed for clarity, a subdued grid has been displayed on the vertical axes, but not on the base and perpendiculars have been dropped from the plotting symbols to the base of the plot, in order to assist in the identification of clusters.

Note that plotting symbols, minus signs in this case, have been added to the foot of the perpendiculars to assist in visualizing the clustering. Also, note that distinct data sets, requiring individual plotting symbols, are identified by a simple rule; data values in each data file are regarded as representing the same cluster, i.e. each cluster must be in a separate file.

20.6.5 Two dimensional families of curves

Users may need to plot families of curves indexed by parameters. For instance, diffusion of a substance from an instantaneous plane source is described by the equation

$$f(x) = \frac{1}{2\sqrt{\pi D t}} \exp\left(\frac{-x^2}{4Dt}\right)$$

which is, of course, a normal distribution with $\mu = 0$ and $\sigma^2 = 2Dt$, where D is the diffusion constant and t is time, so that $2Dt$ is the mean square distance diffused by molecules in time t. Now it is easy to plot the concentration $f(x)$ predicted by this equation as a function of distance x and time t given a diffusion constant D, by simulating the equation using **makdat**, saving the curves to a library file or project archive, then plotting the collected curves. However, there is a much better way using program **usermod** which has the important advantage that families of curves indexed by parameters can be plotted interactively. This is a more powerful technique which provides numerous advantages and convenient options when simulating systems to observe the behavior of the profiles as the indexing parameters vary.

Figure 20.31 shows the above equation plotted (in arbitrary units) using the model parameters

$$p_i = 2Dt_i, \text{ for } i = 1, 2, 3, 4$$

to display the diffusion profiles as a function of time. The plot was created using the model file `family2d.mod`, which simply defines four identical equations corresponding to the diffusion equation but with four different parameters p_i. Program **usermod** was then used to read in the model, simulate it for the parameter values indicated, then plot the curves simultaneously.

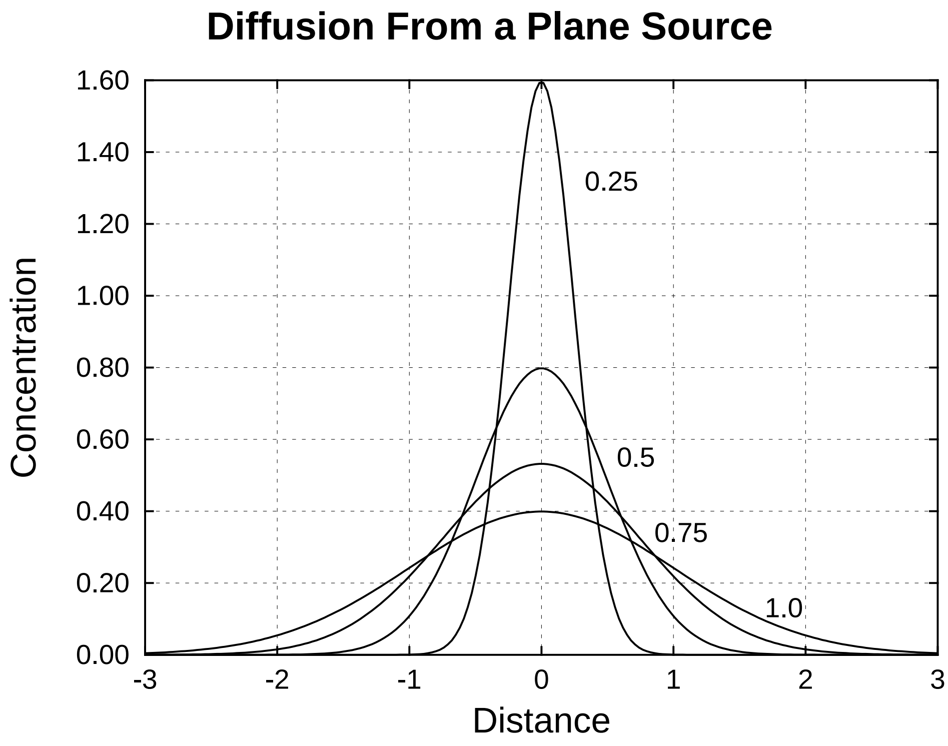


Figure 20.31: Two dimensional families of curves

20.6.6 Three dimensional families of curves

Users may need to plot families of curves indexed by parameters in three dimensions. To show how this is done, the diffusion equation dealt with previously (page 354) is reformulated, using $y = \sqrt{2Dt}$, as

$$z(x, y) = \frac{1}{y\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x}{y}\right)^2\right\}$$

and is plotted in figure 20.32 for the same parameter values used before, but now as sections through the surface of a function of two variables.

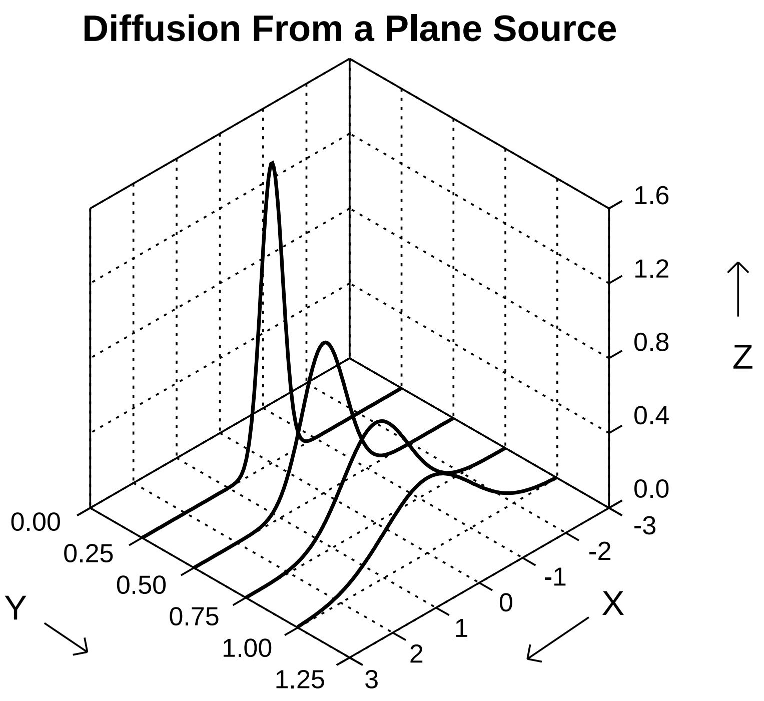


Figure 20.32: Three dimensional families of curves

This is, of course, a case of a family of parametric space curves projected onto the fixed values of y. Now the model file `family3d.mod` was used by program **usermod** to create this figure, using the option to plot n sets of parametric space curves, but you should observe a number of important facts about this model file before attempting to plot your own families of space curves.

- There are 4 curves so there are 12 functions of 1 variable
- Functions $f(1), f(4), f(7), f(10)$ are the parameter t, i.e. x
- Functions $f(2), f(5), f(8), f(11)$ are the y values, i.e. $\sqrt{2Dt}$
- Functions $f(3), f(6), f(9), f(12)$ are the z values, i.e. the concentration profiles

Finally, it is clear that n space curves require a model file that specifies $3n$ equations, but you should also realize that space curves cannot be plotted if there is insufficient variation in any of the independent variables, e.g. if all $y = k$, for some fixed parameter k.

20.7 Specialized techniques

20.7.1 Segmented models with cross-over points

Often segmented models are required with cross-over points where the model equation swaps over at one or more values of the independent variable. In figure 20.33, for instance, data are simulated then fitted using the model `updown.mod`, showing how the three way get command `get3(.,.,.)` can be used to swap over from one model to another at a fixed critical point.

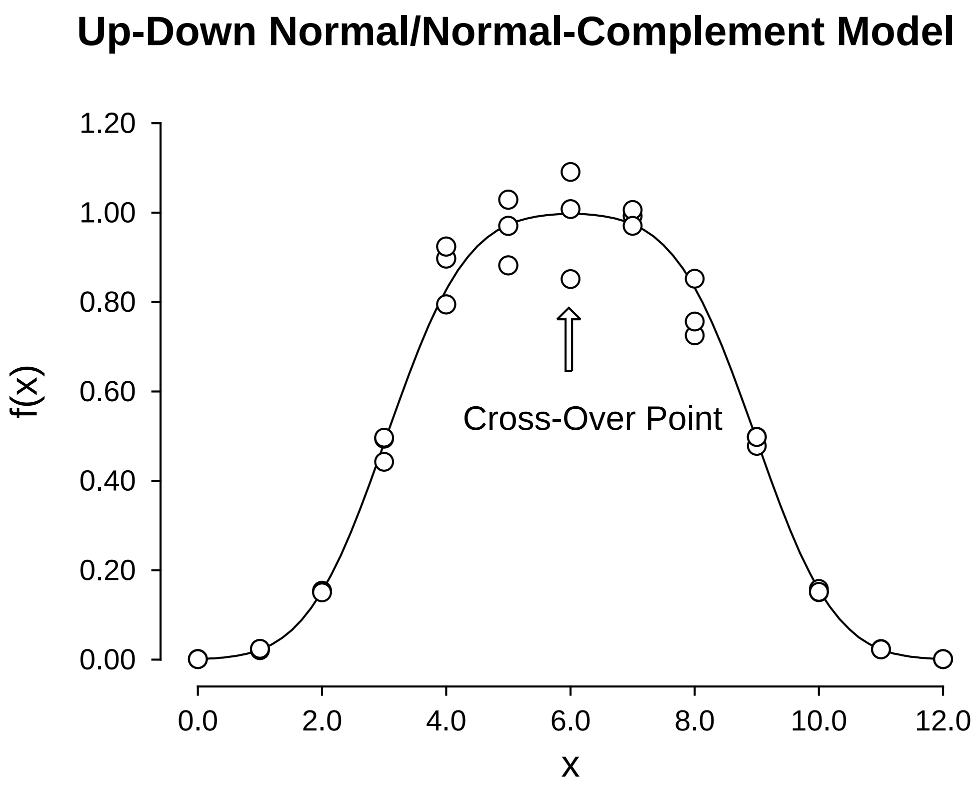


Figure 20.33: Models with cross over points

The model defined by `updown.mod` is

$$\begin{aligned} f(x) &= \Phi((x - p_1)/p_2) \text{ for } x \le 6 \\ &= 1 - \Phi((x - p_3)/p_4) \text{ otherwise} \end{aligned}$$

where $\Phi(.)$ is the cumulative normal distribution, and this is the relevant swap-over code.

```
x
6
subtract
get3(1,1,2)
f(1)
```

The `get3(1,1,2)` command pops the $x - 6$ off the stack and uses `get(1)` or `get(2)` depending on the magnitude of x, since a `get3(i,j,k)` command simply pops the top value off the stack and then uses `get(i)` if this is negative, `get(j)` if this is zero (to machine precision), or `get(k)` otherwise. The cross-over point can also be fixed using an extra parameter that is then estimated, but this can easily lead to ill-determined parameters and a rank deficient covariance matrix if the objective function is insensitive to small variations in the extra parameter.

20.7.2 Plotting single impulse functions

Plotting single impulse functions, as in figure 20.34, sometimes requires care due to the discontinuities.

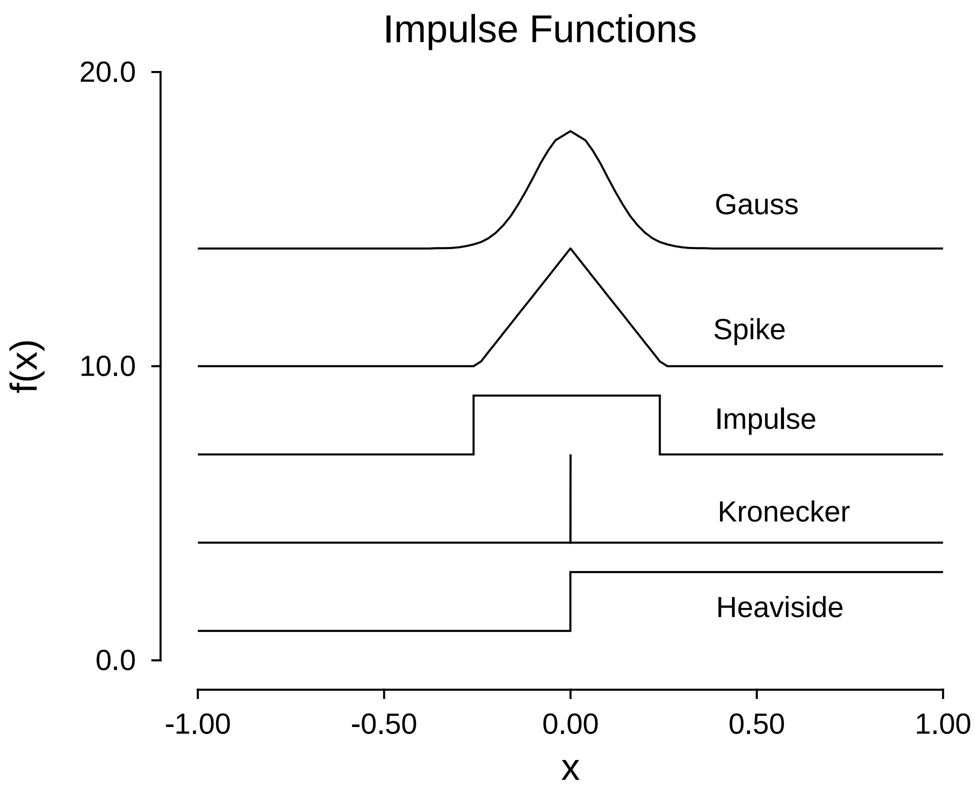


Figure 20.34: Plotting single impulse functions

These graphs were created using program **usermod** together with the model file `impulse.mod`, which defines the five impulse functions of one variable described on page 430, and uses $a = p(1) > 0$ to fix the location, and $b = p(2) > 0$ to set the pulse width where necessary.

- The Heaviside unit function $h(x - a)$. A pdf or survival curve stepped line type is required in order to plot the abrupt step at $x = a$.
- The Kronecker delta symbol δ_{ij}. The x-coordinate data were edited interactively in program **usermod** in order to plot the vertical signal when $i = j$ as a distinct spike. After editing there was one x value at precisely $x = a$, where the function value is one, and one at a short distance either side, where the function values are zero.
- The square impulse function of unit area. Again a stepped line type is necessary to plot the abrupt increase and decrease of this discontinuous function, and it should be noted that, by decreasing the pulse width, the Dirac delta function can be simulated.
- The triangular spike function of unit area is straightforward to simulate and plot as long as the three x-coordinates for the corners of the triangles are present.
- The Gauss function of unit area is easy to plot.

Note that the model file `impulse.mod` uses scaling factors and additive constants so that all five functions can be displayed in a convenient vertically stacked format.

20.7.3 Plotting periodic impulse functions

Plotting periodic impulse functions, as in figure 20.35, sometimes requires care due to the discontinuities.

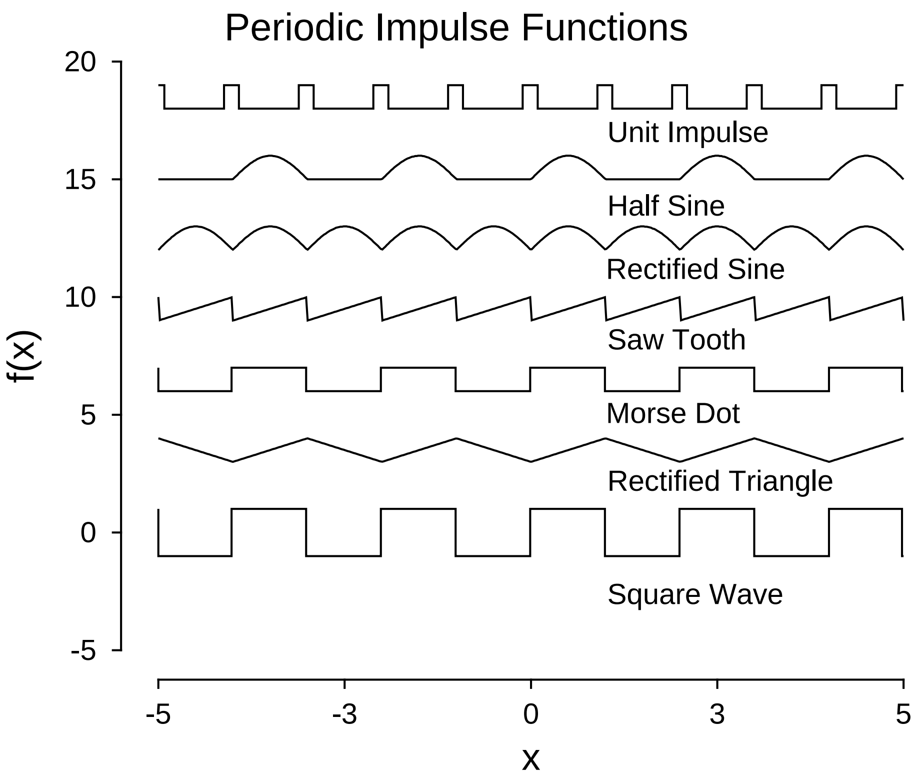


Figure 20.35: Plotting periodic impulse functions

These graphs were created using program **usermod** together with the model file `periodic.mod`, which defines the seven impulse functions of one variable described on page 430, and uses $a = p(1) > 0$ to fix the period and $b = p(2) > 0$ to set the width where required.

- The square wave function oscillates between plus and minus one, so a pdf or survival curve stepped line type is required in order to plot the abrupt step at $x = \lambda a$, for positive integer λ.
- The rectified triangular wave plots perfectly as long as the x-coordinate data are edited interactively in program **usermod** to include the integer multiples of a.
- The Morse dot is just the positive part of the square wave, so it also requires a stepped line type.
- The sawtooth function is best plotted by editing the x-coordinate data to include a point immediately either side of the multiples of a.
- The rectified sine wave and half-wave merely require sufficient points to create a smooth curve.
- The unit impulse function requires a second parameter to define the width b, and this is best plotted using a stepped line type.

Note that the model file `periodic.mod` uses scaling factors and additive constants so that all seven functions can be displayed in a convenient vertically stacked format.

20.7.4 Subsidiary figures as insets

This is easily achieved using **editps** with the individual PostScript files (see page 364) as shown in figure 20.36.

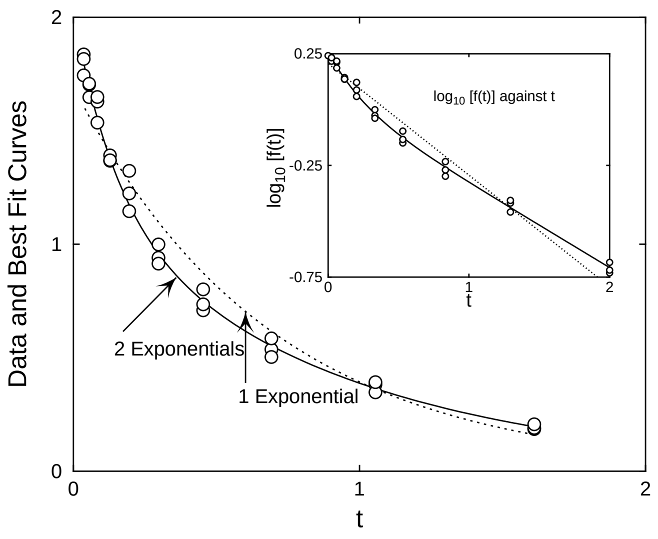


Figure 20.36: Subsidiary figures as insets

20.7.5 Nonlinear growth curves

Figure 20.37 illustrates the use of male and female plotting symbols to distinguish experimental data, which are also very useful when plotting correlation data for males and females.

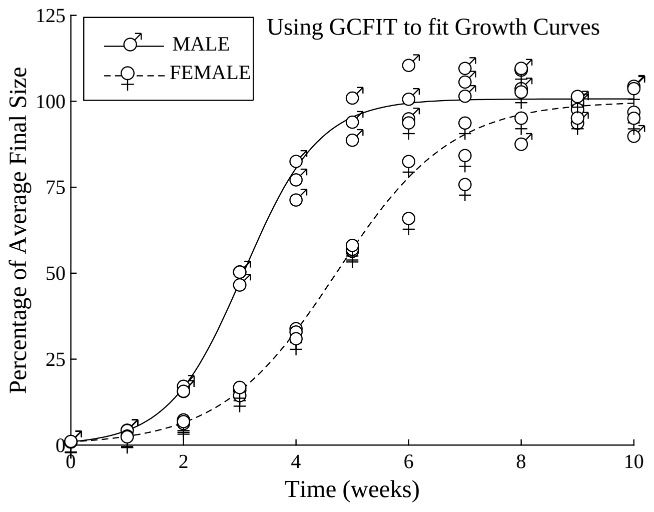


Figure 20.37: Growth curves

20.7.6 Immunoassay and dose-response dilution curves

Antibodies are used in bioassays in concentrations known up to arbitrary multiplicative factors, and dose response curves are constructed by dilution technique, usually 1 in 2, 1 in 3, 1 in 10 or similar. By convention, plots are labelled in dilutions, or powers of the dilution factor and, with this technique, affinities can only be determined up to the unknown factor. Figure 20.38 was constructed using **makfil** in dilution mode with dilutions 1, 2, 4, 8, 16 and 32 to create a data file with concentrations 1/32, 1/16, 1/8, 1/4, 1/2, 1. **hlfit** fitted response as a function of concentration and a dilution curve was plotted.

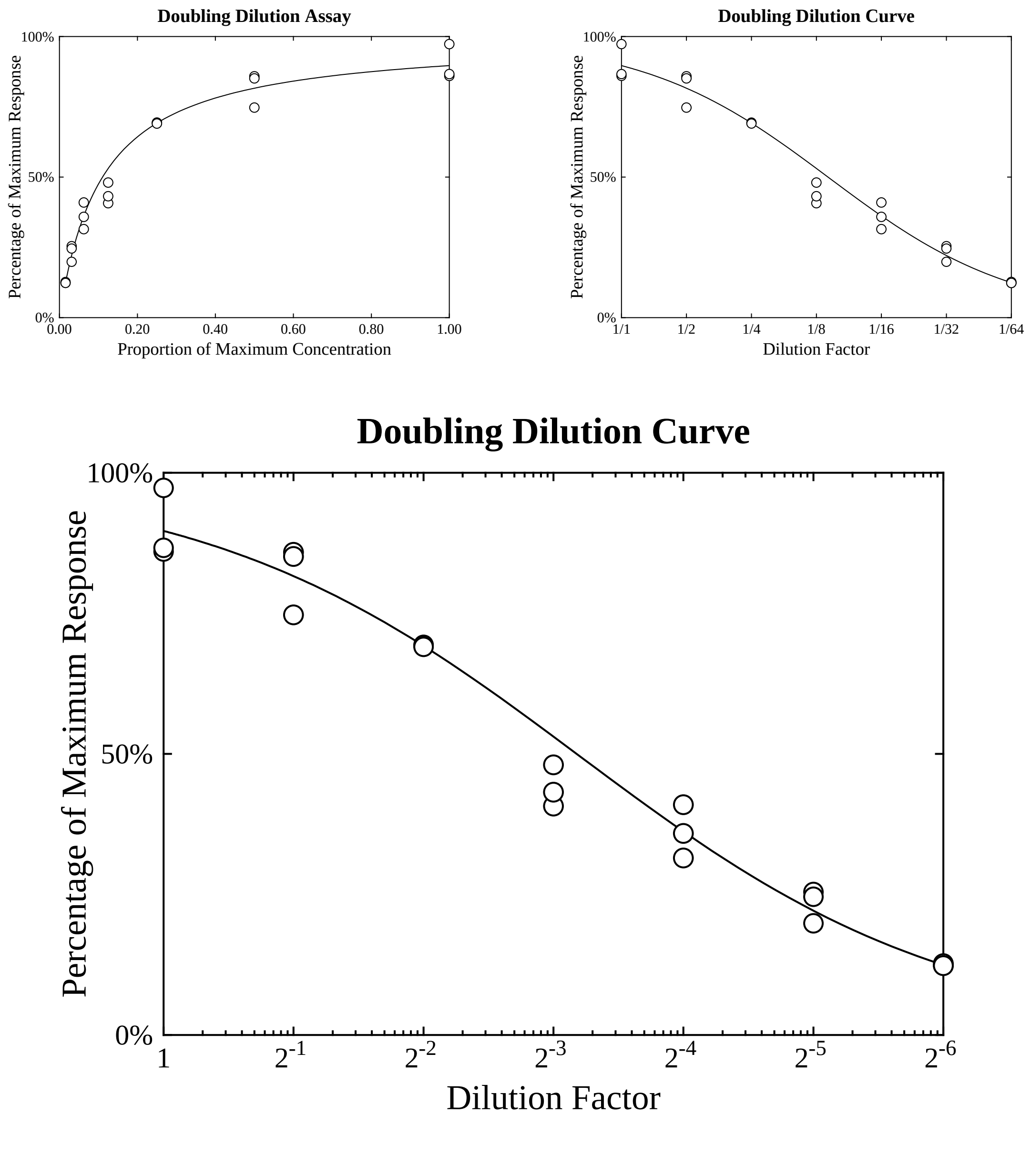


Figure 20.38: Immunoassay and dose-response dilution curves

The transformation is equivalent to plotting log of reciprocal concentration (in arbitrary units) but this is not usually appreciated. SIMPLOT can plot $\log(1/x)$ to bases 2, 3, 4, 5, 6, 7, 8 and 9 as well as e and ten, allowing users to plot trebling, quadrupling dilutions, etc. To emphasize this, intermediate gradations can be added and labeling can be in powers of the base, as now shown.

20.7.7 Information panels

Figure 20.39 illustrates techniques for adding information panels to identify the data plotted. To avoid having to input text interactively where a sequence of similar data sets are to be plotted, default panel labels should be set up using the configuration options.

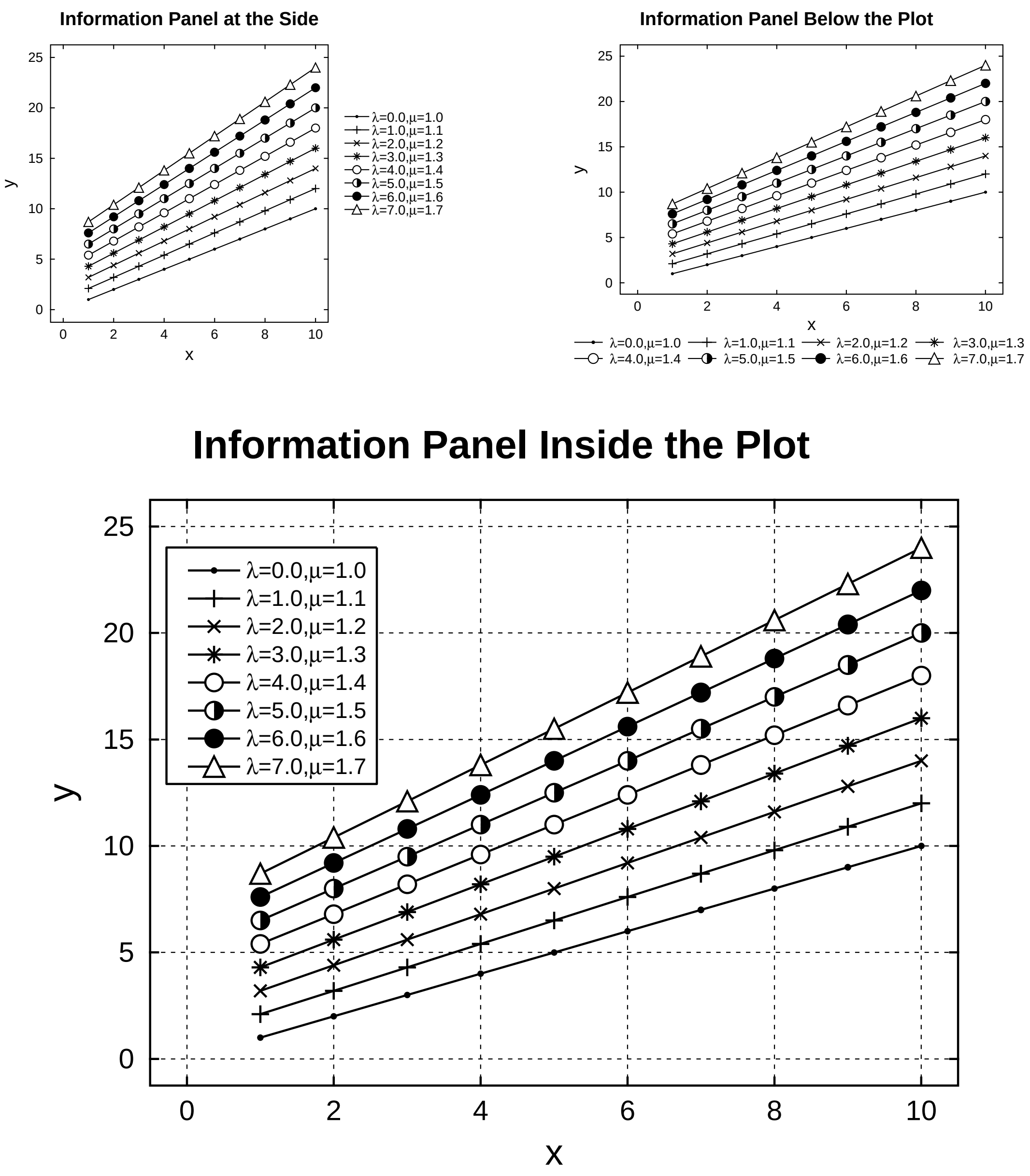


Figure 20.39: Information panels

The first figure illustrates the standard way to display a panel at the right hand side. Up to twenty lines and symbols can be included at the side and it may be more convenient to use a square 1:1 aspect ratio to make space available. The second graph illustrates the option to display the panel below the graph, but this is limited to ten items. The third figure illustrates the effect of using a 4:3 aspect ratio but then dragging and dropping the panel inside the graph. This third example illustrates two further ways to enhance the presentation of graphs with panels: the font size is reduced, and the panel is surrounded by an opaque rectangle which was configured to lie between the axes and data, so obliterating the underlying grid lines.

20.8 Parametric curves

Figure 20.28 and figure 20.31 are examples for parametric curves of the form $x(t), y(t)$, while figure 20.29 and figure 20.32 are for $x(t), y(t), z(t)$. Examples for $r(\theta)$ follow.

20.8.1 $r = r(\theta)$ parametric plot 1: Eight leaved rose

Figure 20.40, for example, was generated using the SimFIT model file `rose.mod` from **usermod** to define an eight leaved rose in $r = r(\theta)$ form using the following code.

```
 %
Example: Eight leaved rose
r = A*sin(4*theta): where theta = x, r = f(1) and A = p(1)
 %
 1 equation
 1 variable
 1 parameter
 %
 x
 4
 multiply
 sin
 p(1)
 multiply
 f(1)
 %
```

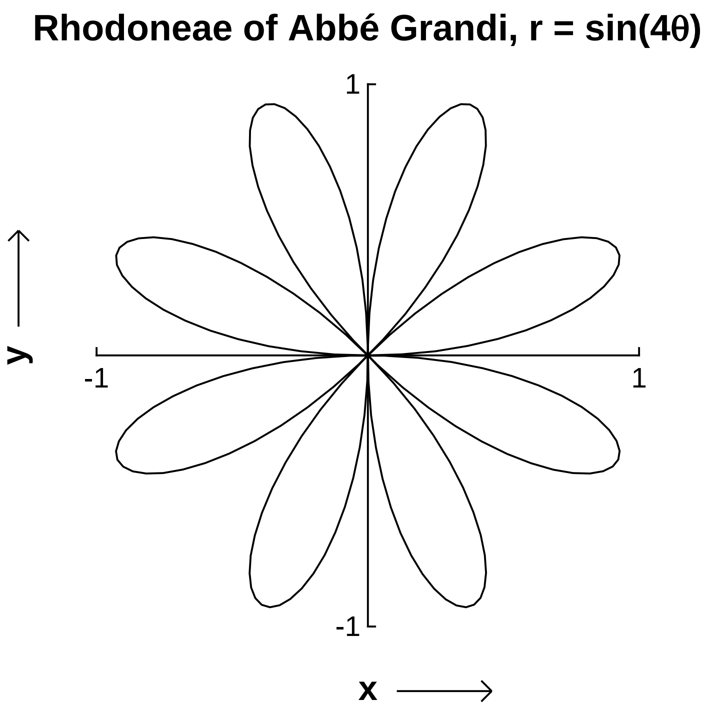


Figure 20.40: $r = r(\theta)$ parametric plot 1. Eight leaved Rose

20.8.2 $r = r(\theta)$ parametric plot 2: Logarithmic spiral with tangent

Figure 20.41 illustrates the logarithmic spiral $r(\theta) = A\exp(\theta\cot\alpha)$, defined in SimFIT model file `camalot.mod` for $A = 1, p(1) = \alpha, x = \theta, r = f(1)$ as follows.

```
 1
p(1)
tan
divide
x
multiply
exp
f(1)
```

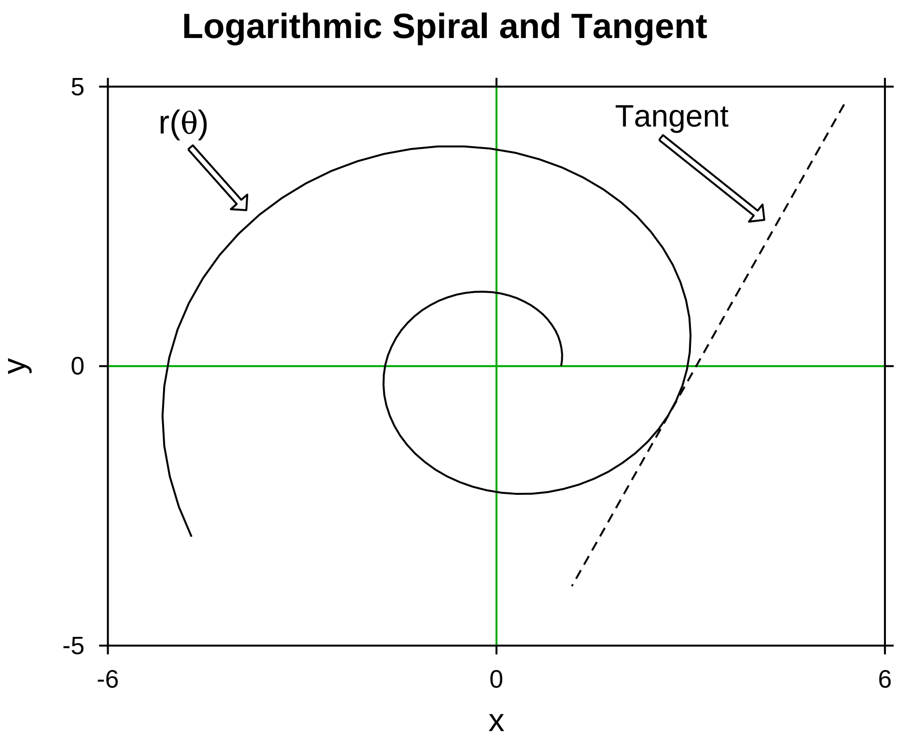


Figure 20.41: $r = r(\theta)$ parametric plot 2. Logarithmic Spiral with Tangent

This profile is used in camming devices such as Camalots and Friends to maintain a constant angle α between the radius vector for the spiral and the tangent to the curve, defined in `tangent.mod` as

$$r = \frac{A\exp(\theta_0\cot\alpha)\,[\sin\theta_0 - \tan(\theta_0+\alpha)\cos\theta_0]}{\sin\theta - \tan(\theta_0+\alpha)\cos\theta}.$$

Figure 20.41 used $\alpha = p(1) = 1.4, \theta_0 = P(2) = 6$ and **usermod** to generate individual figures over the range $0 \le \theta = x \le 10$, then **simplot** plotted the ASCII text coordinates simultaneously, a technique that can be used to overlay any number of curves.

Part 21

PostScript procedures

21.1 Encapsulated PostScript files

The best way to use SimFIT graphics is to archive standard sized SimFIT PostScript files in portrait orientation and then, when required, manipulate them, followed by printing hardcopy, or by transforming into other formats, such as .png, for pasting into documents. Most simple editing operations can be done using a text editor as described later, but for more extensive manipulations program **editps** can be used as now described.

21.1.1 Using editps to manipulate PostScript files

An encapsulated PostScript file (*.eps) is a special type of self-contained one page PostScript file containing a BoundingBox with dimensions, so that the file can be easily manipulated for re-sizing and inclusion within documents. All PostScript files created by SimFIT adhere to this convention. Program **editps** will accept any such files, but some features will only work with SimFIT .eps files.

21.1.2 Editing SimFIT Postscript files

After a SimFIT file has been loaded into **editps** it is possible to search for such items as the title, or legends, etc., and edit as required. However, as described later, it is much easier to do such editing using a simple text editor. Further, this type of editing is restricted to SimFIT PostScript files.

21.1.3 Rotating, re-sizing, and changing aspect ratios.

Figure 20.12 illustrates the effects of simple rotation, but a word of caution is required concerning re-scaling of axes combined with rotations. Figure 20.13 illustrates how shearing effects can result when axes are scaled differentially, and this is combined with rotation, as these operations do not commute.

21.1.4 Creating simple collages

Figure 1.1 illustrates a simple collage created using **editps** from a set of SimFIT PostScript files assembled into a library file. If required, tiles, labels, and extra text can be added by program **editps** to identify the sub-graphs or add further details. To create such collages it is advisable that all the files supplied should have the same dimensions and orientation, as this facility can only generate collages with fixed cell dimensions.

21.1.5 Creating freestyle collages

Figure 1.2 and figure 1.3 show how a collage can be assembled using **editps** in freestyle mode with a set of graphs that can have arbitrary dimensions and rotations. In addition to being able to move the sub-graphs into any positions, this procedure also allows differential re-sizing of individual graphs. There is an extremely important point to remember when creating freestyle collages: it is possible to create PostScript files from

SimFIT where the background, if white, can be either transparent or opaque. Note that PostScript files with opaque white backgrounds will obscure any graphs they overlay. Of course, sometimes this is desired, but sometimes a transparent white background may be preferred.

21.1.5.1 Creating insets

Figure 20.36 illustrates a freestyle collage with a sub-graph inside a parent graph. It is best to enlarge fonts and increase line thicknesses when a sub-graph is going to be reduced in size in this way, and it is always important to remember the effects of opaque and transparent backgrounds just discussed. As this is such a frequently used procedure, the steps required to create figure 20.36 will be described.

First the plots in figure 21.1 were created using **exfit** with test file `exfit.tf4` after fitting 1 exponential then 2 exponentials. Note that the line thickness and font size have been increased in the transformed plot to be reduced in the inset. Figure 21.2 was then created by **editps** using the option to create a freestylecollage.

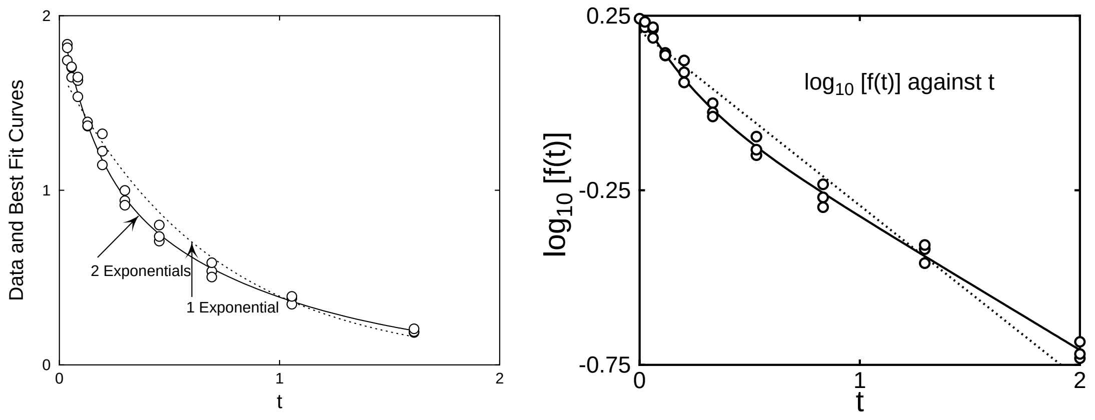


Figure 21.1: Insets 1: Exponential fitting and semilog transforms

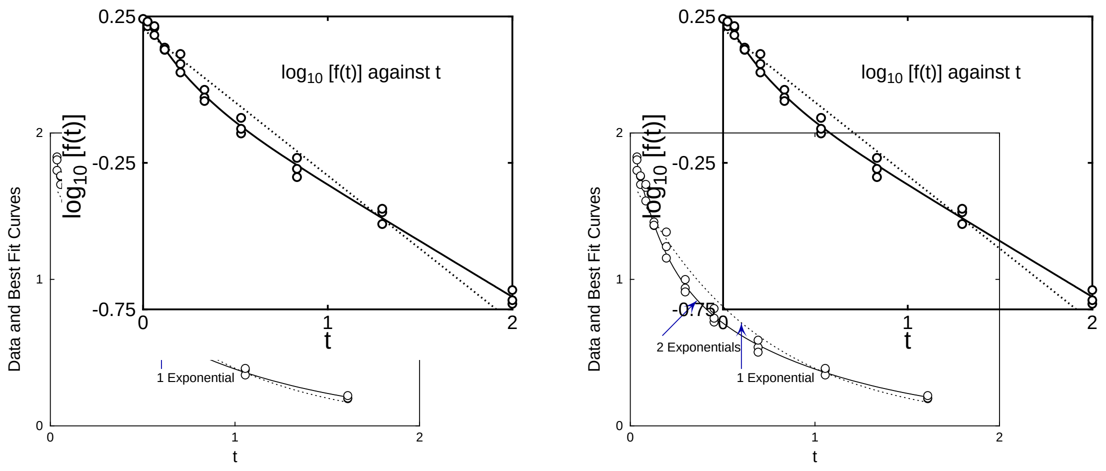


Figure 21.2: Insets 2: Opaque and transparent backgrounds in insets

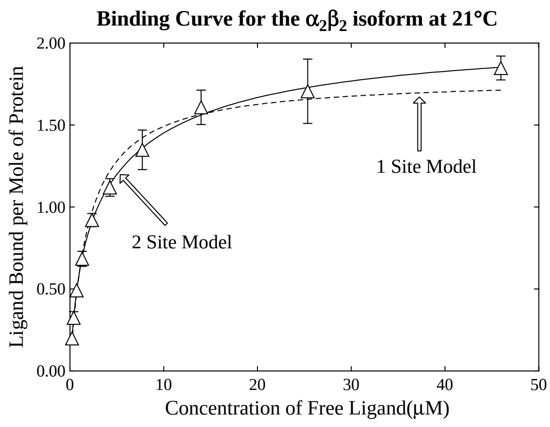
Note how, in the left hand plot the option to plot an opaque background even when white was selected and the transformed plot obscures the underlying main plot. In the right hand plot the option for a transparent background was used so that the main plot was not obscured. Both techniques are valuable when creating insets, and all that is now necessary to create figure 20.36 is to shrink the transformed plot and translate it to a more convenient location. A further point to note is that SimFIT plots have a border, which is obscuring more of the left hand main figure in figure 21.2 than seems necessary. When subsidary figures are going to be used in this way it is often advisable to use the option to clip the plot to trim away extra white space, or else use GsView to calculate a new BoundingBox in a transparent subsidiary plot by transforming ps into eps.

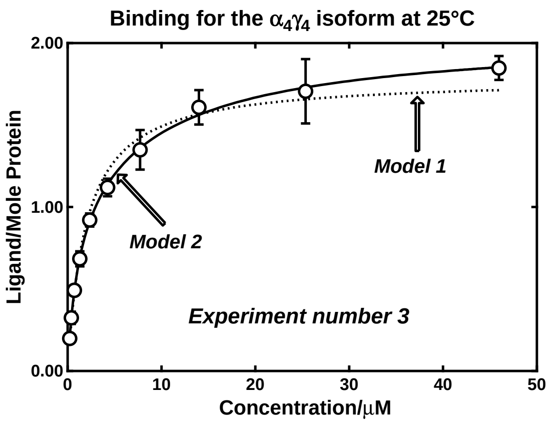
21.1.5.2 Creating split graphs

Figure 20.15 shows an extreme type of freestyle collage, when it is necessary to split a graph into sections and recombine them using **editps**. By using the facility to clip a graph into sections when the graph is first created, there is no limit to the number of sections that can be re-combined in this way.

21.2 The format of SimFIT PostScript files

One of the unique features of SimFIT PostScript files is that the format is designed to make retrospective editing easy. A typical example of when this could be useful would be when a graph needs to be changed for some reason. Typically an experimentalist might have many plots stored as .eps files and want to alter one for publication or presentation. SimFIT users are strongly recommended to save all their plots as .ps or .eps files, so that they can be altered in the way to be described. Even if you do not have a PostScript printer it is still best to save as .ps, then use program **GSview** to print or transform into another graphics format. Consider these next two figures, showing how a graph can be transformed by simple editing in a text editor, e.g. NOTEPAD.





This type of editing should always be done if you want to use one figure as a reduced size inset figure inside another, or when making a slide, otherwise the SimFIT default line thickness will be too thin. Note that most of the editing to be described below can actually be done at the stage of creating the file, or by using program EDITPS. In this hypothetical example, we shall suppose that the experimentalist had realized that the title referred to the wrong isoform and temperature, and also wanted to add extra detail, but simplify the graph in order to make a slide using thicker lines and a bolder font. In the following sections the editing required to transform the SimFIT example file `simfig1.ps` will be discussed, following a preliminary warning.

21.2.1 Advice about editing PostScript files

In the first place the technique to be described can only be done with SimFIT PostScript files, because the format was developed to facilitate the sort of editing that scientists frequently need to perform. Secondly, it must be realized that PostScript files must conform to a very strict set of rules. If you violate these rules, then GSview will warn you and indicate the fault. Unfortunately, if you do not understand PostScript, the warning will be meaningless. So here are some rules that you must keep in mind when editing.

❏ Always keep a backup copy at each successful stage of the editing.

❏ All text after a single percentage sign `%` to the line end is ignored in PostScript.

- ❏ Parentheses must always be balanced as in (figure 1(a)) not as in (figure 1(a).
- ❏ Fonts must be spelled correctly, e.g. Helvetica-Bold and not helveticabold.
- ❏ Character strings for displaying must have underneath them a vector index string of EXACTLY the same length.
- ❏ When introducing non-keyboard characters each octal code represents one byte.
- ❏ The meaning of symbols and line types depends on the function, e.g. `da` means dashed line while `do` means dotted line.

A review of the PostScript colours, fonts and conventions is also in the `w_readme` files. In the next sections it will be assumed that are running SIMFIT and have a renamed copy of `simfig1.ps` in your text editor (e.g. notepad), and after each edit you will view the result using program **GSview**. Any errors reported when you try to view the edited file will be due to violation of a PostScript convention. The most usual one is to edit a text string without correctly altering the index below it to have exactly the same number of characters.

21.2.1.1 The percent-hash escape sequence

Later versions of SIMFIT create PostScript files that can be edited by a stretch, clip, slide procedure, which relies on each line containing coordinates being identified by a comment line starting with %#. All text extending to the right from the first character of this sequence can safely be ignored and is suppressed for clarity in the following examples.

21.2.1.2 Changing line thickness and plot size

The following text will be observed in the original `simfig1.ps` file.

```
72.00  252.00 translate    0.07    0.07 scale    0.00 rotate
   11.00 setlinewidth 0 setlinecap 0 setlinejoin [] 0 setdash
    2.50 setmiterlimit
```

The postfix argument for setlinewidth alters the line width globally. In other words, altering this number by a factor will alter all the linewidths in the figure by this factor, irrespective on any changes in relative line thicknesses set when the file was created. The translate, scale and rotate are obvious, but perhaps best done by program EDITPS. Here is the same text edited to increase the line thickness by a factor of two and a half.

```
72.00  252.00 translate    0.07    0.07 scale    0.00 rotate
   27.50 setlinewidth 0 setlinecap 0 setlinejoin [] 0 setdash
    2.50 setmiterlimit
```

21.2.1.3 Changing PostScript fonts

In general the Times-Roman fonts may be preferred for readability in diagrams to be included in books, while Helvetica may look better in scientific publications. For making slides it is usually preferable to use Helvetica-Bold. Of course any PostScript fonts can be used, but in the next example we see how to change the fonts in `simfig1.ps` to achieve the effect illustrated.

```
/ti-font /Times-Bold  D%plot-title
/xl-font /Times-Roman D%x-legend
/yl-font /Times-Roman D%y-legend
/zl-font /Times-Roman D%z-legend
/tc-font /Times-Roman D%text centred
/td-font /Times-Roman D%text down
/tl-font /Times-Roman D%text left to right
/tr-font /Times-Roman D%text right to left
/ty-font /Times-Roman D%text right y-mid
/tz-font /Times-Roman D%text left  y-mid
```

The notation is obvious, the use indicated being clear from the comment text following the percentage sign % at each definition, denoted by a D. This is the editing needed to bring about the font substitution.

```
/ti-font /Helvetica-Bold D%plot-title
/xl-font /Helvetica-Bold D%x-legend
/yl-font /Helvetica-Bold D%y-legend
/zl-font /Helvetica-Bold D%z-legend
/tc-font /Helvetica-Bold D%text centred
/td-font /Helvetica-Bold D%text down
/tl-font /Helvetica-Bold D%text left to right
/tr-font /Helvetica-Bold D%text right to left
/ty-font /Helvetica-Bold D%text right y-mid
/tz-font /Helvetica-Bold D%text left  y-mid
```

Observing the scheme for colours (just before the fonts in the file) and text sizes (following the font definitions) will make it obvious how to change colours and text sizes.

21.2.1.4 Changing title and legends

Observe the declaration for the title and legends in the original file.

```
(Binding Curve for the a2b2 isoform at 21@C) 3514 4502 ti
(000000000000000000000061610000000000000060) fx
(Concentration of Free Ligand(lM)) 3514 191 xl
(00000000000000000000000000000300) fx
(Ligand Bound per Mole of Protein) 388 2491 yl
(00000000000000000000000000000000) fx
```

Note that, for each of the text strings displayed, there is a corresponding index of font substitutions. For example a zero prints the letter in the original font, a one denotes a subscript, while a six denotes bold maths. Since the allowed number of index keys is open-ended, the number of potential font substitutions is enormous. You can have any accent on any letter, for instance. This is the editing required to change the text. However, note that the positions of the text do not need to be changed, the font display functions work out the correct position to centre the text string.

```
(Binding for the a4c4 isoform at 25@C) 3514 4502 ti
(0000000000000000061610000000000000060) fx
(Concentration/lM) 3514 191 xl
(0000000000000030) fx
(Ligand/Mole Protein) 388 2491 yl
(0000000000000000000) fx
```

Note that the \ character is an escape character in PostScript so, if you want to have something like an unbalanced parenthesis, as in `Figure 1 a)` you would have to write `Figure 1a\)`. When you create a PostScript file from SɪᴍFɪT it will prevent you from writing a text string that violates PostScript conventions but, when you are editing, you must make sure yourself that the conventions are not violated, e.g. use `c:\\simfit` instead of `c:\simfit`.

21.2.1.5 Deleting graphical objects

It is very easy to delete any text or graphical object by simply inserting a percentage sign % at the start of the line to be suppressed. In this way an experimental observation can be temporarily suppressed, but it is still in the file to be restored later if required. Here is the PostScript code for the notation on the left hand vertical, i.e. y axis in the file `simfig1.ps`.

```
910 1581 958 1581 li
6118 1581 6070 1581 li
(0.50) 862 1581 ty
(0000) fx
910 2491 958 2491 li
6118 2491 6070 2491 li
(1.00) 862 2491 ty
(0000) fx
910 3401 958 3401 li
6118 3401 6070 3401 li
(1.50) 862 3401 ty
(0000) fx
```

This is the text, after suppressing the tick marks and notation for $y = 0.5$ and $y = 1.5$ by inserting a percentage sign. Note that the index must also be suppressed as well as the text string.

```
%910 1581 958 1581 li
 %6118 1581 6070 1581 li
 %(0.50) 862 1581 ty
 %(0000) fx
910 2491 958 2491 li
6118 2491 6070 2491 li
(1.00) 862 2491 ty
(0000) fx
 %910 3401 958 3401 li
 %6118 3401 6070 3401 li
 %(1.50) 862 3401 ty
 %(0000) fx
```

21.2.1.6 Changing line and symbol types

This is simply a matter of substituting the desired line or plotting symbol key.

```
Lines     : li (normal) da (dashed) do (dotted) dd (dashed dotted) pl (polyline)
Circles   : ce (empty)  ch (half)   cf(full)
Triangles: te (empty)  th (half)   tf (full)
Squares   : se (empty)  sh (half)   sf (full)
Diamonds  : de (empty)  dh (half)   df (full)
Signs     : ad (add)    mi (minus)  cr (cross)  as (asterisk)
```

Here is the original text for the dashed line and empty triangles.

```
5697 3788 120 da
933 1032 72 te
951 1261 72 te
984 1566 73 te
1045 1916 72 te
1155 2346 72 te
1353 2708 73 te
1714 3125 72 te
2367 3597 72 te
3551 3775 72 te
5697 4033 72 te
```

Here is the text edited for a dotted line and empty circles.

```
5697 3788 120 do
933 1032 72 ce
951 1261 72 ce
984 1566 73 ce
1045 1916 72 ce
1155 2346 72 ce
1353 2708 73 ce
1714 3125 72 ce
2367 3597 72 ce
3551 3775 72 ce
5697 4033 72 ce
```

21.2.1.7 Adding extra text

Here is the original extra text section.

```
/font /Times-Roman D /size   216 D
GS font F size S   4313  2874 M    0 rotate
(1 Site Model)
(000000000000) fx
/font /Times-Roman D /size   216 D
GS font F size S   1597  2035 M    0 rotate
(2 Site Model)
(000000000000) fx
```

Here is the above text after changing the font.

```
/font /Helvetica-BoldOblique D /size   216 D
GS font F size S   4313  2874 M    0 rotate
(Model 1)
(0000000) fx
/font /Helvetica-BoldOblique D /size   216 D
GS font F size S   1597  2035 M    0 rotate
(Model 2)
(0000000) fx
```

Here is the additional code required to add another label to the plot.

```
/font /Helvetica-BoldOblique D /size   240 D
GS font F size S   2250  1200 M    0 rotate
(Experiment number 3)
(0000000000000000000) fx
```

21.2.1.8 Changing colors

Colors c0 to c71 in the file header can be edited, but c0 and c15 should be black and white. For instance, changing

```
c4
(Survival Analysis) 3195 4467 ti%#title
(00000000000000000) fx
```

into

```
c0
(Survival Analysis) 3195 4467 ti%#title
(00000000000000000) fx
```

would change a red title into a black title.

21.3 Standard fonts

All PostScript printers have a basic set of 35 fonts and it can be safely assumed that graphics using these fonts will display in program **GSview** and print on all except the most primitive PostScript printers. Of course there may be a wealth of other fonts available. The Times and Helvetica fonts are well known, and the monospaced Courier family of typewriter fonts are sometimes convenient for tables.

Times-Roman
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Times-Bold
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Times-Italic
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Times-BoldItalic
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica-Bold
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica-Oblique
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

Helvetica-BoldOblique
!"#$%&’()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_‘
abcdefghijklmnopqrstuvwxyz{|}~

21.3.1 Decorative fonts

Sometimes decorative or graphic fonts are required, such as pointing hands or scissors. It is easy to include such fonts using program **simplot**, although the characters will be visible only if the plot is inspected using program **GSview**.

Symbol
!∀#∃%&∋()∗+,−./0123456789:;<=>?≅ ‾
ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ[]⊥_ ‾
αβχδεφγηιϕκλμνοπθρστυϖωξψζ{|}∼

ZapfDingbats
✁✂✃✄☎✆✇✈✉☛☞✌✍✎✏✐✑✒✓✔✕✖✗✘✙✚✛✜✝✞✟✠
✡✢✣✤✥✦✧★✩✪✫✬✭✮✯✰✱✲✳✴✵✶✷✸✹✺✻✼✽✾✿❀
❁❂❃❄❅❆❇❈❉❊❋●❍■❏❐❑❒▲▼◆❖◗❘❙❚❛❜❝❞

ZapfChancery-MediumItalic
!"#$%&'()+,-./0123456789:;<=>?@*
ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_'
abcdefghijklmnopqrstuvwxyz{|}~

Some extra characters in Times, Helvetica, etc.
æ(361)•(267)†(262)‡(263)¡(241)ƒ(246)œ(372)¿(277)˚(312)§(247)£(243)

Some extra characters in Symbol
∠(320)〈(341)〉(361)≈(273)↔(253)⇔(333)⇐(334)⇒(336)←(254)→(256)|(174)
⊗(304)⊕(305)°(260)÷(270)∈(316)…(274)∅(306)≡(272)ƒ(246)∇(321)≥(263)
∞(245)∫(362)≤(243)×(264)≠(271)∏(325)∂(266)±(261)√(326)∑(345)∪(310)

21.3.2 Plotting characters outside the keyboard set

To use characters outside the keyboard set you have to use the corresponding octal codes. Note that these codes represent just one byte in PostScript so, in this special case, four string characters need only one key character. For example, such codes as `\277` for an upside down question mark in standard encoding, or `\326` for a square root sign in Symbol, only need one index key. You might wonder why, if Simplot can put any accent on any character and there are maths and bold maths fonts, you would ever want alternative encodings, like the ISOLatin1Encoding. This is because the ISOLatin1Encoding allows you to use specially formed accented letters, which are more accurately proportioned than those generated by program Simplot by adding the accent as a second over-printing character, e.g. using `\361` for n tilde is more professional than overprinting.
All the characters present in the coding vectors to be shown next can be used by program Simplot, as well as a special Maths/Greek font and a vast number of accented letters and graphical objects, but several points must be remembered.

All letters can be displayed using program ***GSview*** *and then* ***Adobe Acrobat Reader*** *after distilling to pdf. Although substitutions can be made interactively from Simplot, you can also save a .eps file and edit it in a text editor. When using an octal code to introduce a non-keyboard character, only use one index key for the four character code. If you do not have a PostScript printer, save plots as .eps files and print from program* ***GSview*** *or transform into graphics files to include in documents.*

Some useful codes follow, then by examples to clarify the subject. You will find it instructive to view `simfonts.ps` in the SimFIT viewer and display it in program **GSview**.

21.3.2.1 The StandardEncoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	$	%	&	’
\05x	(	)	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	_
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	
\20x								
\21x								
\22x								
\23x								
\24x		¡	¢	£	⁄	¥	ƒ	§
\25x	¤	'	“	«	‹	›	ﬁ	ﬂ
\26x		–	†	‡	·		¶	•
\27x	‚	„	”	»	…	‰		¿
\30x		`	´	ˆ	˜	¯	˘	˙
\31x	¨		˚	¸		˝	˛	ˇ
\32x	—							
\33x								
\34x		Æ		ª				
\35x	Ł	Ø	Œ	º				
\36x		æ				ı		
\37x	ł	ø	œ	ß				

21.3.2.2 The ISOLatin1Encoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	$	%	&	’
\05x	(	)	*	+	,	−	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	_
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	
\20x								
\21x								
\22x	ı	`	´	ˆ	˜	¯	˘	˙
\23x	¨		˚	¸		˝	˛	ˇ
\24x		¡	¢	£	¤	¥	¦	§
\25x	¨	©	ª	«	¬	-	®	¯
\26x	°	±	²	³	´	µ	¶	·
\27x	¸	¹	º	»	¼	½	¾	¿
\30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
\31x	È	É	Ê	Ë	Ì	Í	Î	Ï
\32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×
\33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
\34x	à	á	â	ã	ä	å	æ	ç
\35x	è	é	ê	ë	ì	í	î	ï
\36x	ð	ñ	ò	ó	ô	õ	ö	÷
\37x	ø	ù	ú	û	ü	ý	þ	ÿ

21.3.2.3 The SymbolEncoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	∀	#	∃	%	&	∋
\05x	(	)	∗	+	,	−	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	≅	Α	Β	Χ	Δ	Ε	Φ	Γ
\11x	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
\12x	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω
\13x	Ξ	Ψ	Ζ	[	∴	]	⊥	_
\14x	‾	α	β	χ	δ	ε	ϕ	γ
\15x	η	ι	φ	κ	λ	μ	ν	ο
\16x	π	θ	ρ	σ	τ	υ	ϖ	ω
\17x	ξ	ψ	ζ	{	|	}	∼	
\20x								
\21x								
\22x								
\23x								
\24x		ϒ	′	≤	⁄	∞	ƒ	♣
\25x	♦	♥	♠	↔	←	↑	→	↓
\26x	°	±	″	≥	×	∝	∂	•
\27x	÷	≠	≡	≈	…	⏐	⎯	↵
\30x	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩
\31x	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
\32x	∠	∇	®	©	™	∏	√	⋅
\33x	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
\34x	◊	〈	®	©	™	∑	⎛	⎜
\35x	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
\36x		〉	∫	⌠	⎮	⌡	⎞	⎟
\37x	⎠	⎤	⎥	⎦	⎫	⎬	⎭	

21.3.2.4 The ZapfDingbatsEncoding Vector

octal	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		✁	✂	✃	✄	☎	✆	✇
\05x	✈	✉	☛	☞	✌	✍	✎	✏
\06x	✐	✑	✒	✓	✔	✕	✖	✗
\07x	✘	✙	✚	✛	✜	✝	✞	✟
\10x	✠	✡	✢	✣	✤	✥	✦	✧
\11x	★	✩	✪	✫	✬	✭	✮	✯
\12x	✰	✱	✲	✳	✴	✵	✶	✷
\13x	✸	✹	✺	✻	✼	✽	✾	✿
\14x	❀	❁	❂	❃	❄	❅	❆	❇
\15x	❈	❉	❊	❋	●	❍	■	❏
\16x	❐	❑	❒	▲	▼	◆	❖	◗
\17x	❘	❙	❚	❛	❜	❝	❞	
\20x	❨	❩	❪	❫	❬	❭	❮	❯
\21x	❰	❱	❲	❳	❴	❵		
\22x								
\23x								
\24x		❡	❢	❣	❤	❥	❦	❧
\25x	♣	♦	♥	♠	①	②	③	④
\26x	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷
\27x	❸	❹	❺	❻	❼	❽	❾	❿
\30x	➀	➁	➂	➃	➄	➅	➆	➇
\31x	➈	➉	➊	➋	➌	➍	➎	➏
\32x	➐	➑	➒	➓	➔	→	↔	↕
\33x	➘	➙	➚	➛	➜	➝	➞	➟
\34x	➠	➡	➢	➣	➤	➥	➦	➧
\35x	➨	➩	➪	➫	➬	➭	➮	➯
\36x	✐	➱	➲	➳	➴	➵	➶	➷
\37x	➸	➹	➺	➻	➼	➽	➾	

21.3.3 SɪᴍFɪT character display codes

0	Standard font
1	Standard font subscript
2	Standard font superscript
3	Maths/Greek
4	Maths/Greek subscript
5	Maths/Greek superscript
6	Bold Maths/Greek
7	ZapfDingbats (PostScript) Wingding (Windows)
8	ISOLatin1Encoding (PostScript), Standard (Windows, almost)
9	Special (PostScript) Wingding2 (Windows)
A	Grave accent
B	Acute accent
C	Circumflex/Hat
D	Tilde
E	Macron/Bar/Overline
F	Dieresis
G	Maths/Greek-hat
H	Maths/Greek-bar
I	Bold maths/Greek-hat
J	Bold Maths/Greek-bar
K	Symbol font
L	Bold Symbol font

You will need non-keyboard characters from the standard font for such characters as a double dagger (‡) or upside down question mark (¿), e.g. typing `\277` in a text string would generate the upside down question mark (¿) in the PostScript output. If you want to include a single backslash in a text string, use `\\`, and also cancel any unpaired parentheses using `\(` and `\)`. Try it in program SIMPLOT and it will then all make sense. The ISOLatin1Encoding vector is used for special characters, such as `\305` for Angstrom (Å), `\361` for n-tilde (ñ), or `\367` for the division sign (÷), and, apart from a few omissions, the standard Windows font is the same as the ISOLatin1Encoding. The Symbol and ZapfDingbats fonts are used for including special graphical characters like scissors or pointing hands in a text string.

A special font is reserved for PostScript experts who want to add their own character function. Note that, in a document with many graphs, the prologue can be cut out from all the graphs and sent to the printer just once at the start of the job. This compresses the PostScript file, saves memory and speeds up the printing. Examine the manuals source code for this technique.

If you type four character octal codes as character strings for plotting non-keyboard characters, you do not have to worry about adjusting the character display codes, program SIMPLOT will make the necessary corrections. The only time you have to be careful about the length of character display code vectors is when editing in a text editor. If in doubt, just pad the character display code vector with question marks until it is the same length as the character string.

21.3.4 editps text formatting commands

Program **editps** uses the SimFIT convention for text formatting characters within included SimFIT .eps files but, because this is rather cumbersome, a simplified set of formatting commands is available within **editps** whenever you want to add text, or even create PostScript files containing text only. The idea of these formatting commands is to allow you to introduce superscripts, subscripts, accented letters, maths, dashed lines or plotting symbols into PostScript text files, or into collage titles, captions, or legends, using only ASCII text controls. To use a formatting command you simply introduce the command into the text enclosed in curly brackets as in: {raise}, {lower}, {newline}, and so on. If {anything} is a recognized command then it will be executed when the .eps file is created. Otherwise the literal string argument, i.e. anything, will be printed with no inter-word space. Note that no {commands} add interword spaces, so this provides a mechanism to build up long character strings and also control spacing; use {anything} to print anything with no trailing inter-word space, or use { } to introduce an inter-word space character. To introduce spaces for tabbing, for instance, just use {newline}{   }start-of-tabbing, with the number of spaces required inside the { }. Note that the commands are both spelling and case sensitive, so, for instance, {21}{degree}{C} will indicate the temperature intended, but {21}{degrees}{C} will print as 21degreesC while {21}{Degree}{C} will produce 21DegreeC.

21.3.4.1 Special text formatting commands, e.g. left

{left}	. . . use {left} to print a {
{right}	. . . use {right} to print a }
{%!command}	. . . use {%!command} to issue command as raw PostScript

The construction {%!command} should only be used if you understand PostScript. It provides PostScript programmers with the power to create special effects. For example {%!1 0 0 setrgbcolor}, will change the font colour to red, and {%!0 0 1 setrgbcolor} will make it blue, while {%!2 setlinewidth} will double line thickness. In fact, with this feature, it is possible to add almost any conceivable textual or graphical objects to an existing .eps file.

21.3.4.2 Coordinate text formatting commands, e.g. raise

{raise}	. . . use {raise} to create a superscript or restore after {lower}
{lower}	. . . use {lower} to create a subscript or restore after {raise}
{increase}	. . . use {increase} to increase font size by 1 point
{decrease}	. . . use {decrease} to decrease font size by 1 point
{expand}	. . . use {expand} to expand inter-line spacing by 1 point
{contract}	. . . use {contract} to contract inter-line spacing by 1 point

21.3.4.3 Currency text formatting commands, e.g. dollar

{dollar} $	{sterling} £	{yen} Y

21.3.4.4 Maths text formatting commands, e.g. divide

{divide} ÷	{multiply} ×	{plusminus} ±

21.3.4.5 Scientific units text formatting commands, e.g. Angstrom

{Angstrom} Å	{degree} $\circ$	{micron} μ

21.3.4.6 Font text formatting commands, e.g. roman

{roman}	{bold}	{italic}	{helvetica}
{helveticabold}	{helveticaoblique}	{symbol}	{zapfchancery}
{zapfdingbats}	{isolatin1}		

Note that you can use octal codes to get extra-keyboard characters, and the character selected will depend on whether the StandardEncoding or IOSLatin1Encoding is current. For instance, \ 361 will locate an {ae} character if the StandardEncoding Encoding Vector is current, but it will locate a {ñ} character if the ISOLatin1Encoding Encoding Vector is current, i.e. the command {isolatin1} has been used previously. The command {isolatin1} will install the ISOLatin1Encoding Vector as the current Encoding Vector until it is cancelled by any font command, such as {roman}, or by any shortcut command such as {ntilde} or {alpha}. For this reason, {isolatin1} should only be used for characters where shortcuts like {ntilde} are not available.

21.3.4.7 Poor man's bold text formatting command, e.g. pmb?

The command {pmb?} will use the same technique of overprinting as used by the Knuth TEX macro to render the argument, that is ? in this case, in bold face font, where ? can be a letter or an octal code. This is most useful when printing a boldface character from a font that only exists in standard typeface. For example, {pmbb} will print a boldface letter b in the current font then restore the current font, while {symbol}{pmbb}{roman} will print a boldface beta then restore roman font. Again, {pmb\ 243} will print a boldface pound sign.

21.3.4.8 Punctuation text formatting commands, e.g. dagger

{dagger} †	{daggerdbl} ‡	{paragraph} ¶	{subsection} §
{questiondown} ¿			

21.3.4.9 Letters and accents text formatting commands, e.g. Aacute

{Aacute} Á	{agrave} à	{aacute} á	{acircumflex} â
{atilde} ã	{adieresis} ä	{aring} å	{ae} æ
{ccedilla} ç	{egrave} è	{eacute} é	{ecircumflex} ê
{edieresis} ë	{igrave} ì	{iacute} í	{icircumflex} î
{idieresis} ï	{ntilde} ñ	{ograve} ò	{oacute} ó
{ocircumflex} ô	{otilde} õ	{odieresis} ö	{ugrave} ù
{uacute} ú	{ucircumflex} û	{udieresis} ü	

All the other special letters can be printed using {isolatin1} (say just once at the start of the text) then using the octal codes, for instance {isolatin1}{\ 303} will print an upper case ntilde.

21.3.4.10 Greek text formatting commands, e.g. alpha

{alpha} α	{beta} β	{chi} χ	{delta} δ
{epsilon} ϵ	{phi} ϕ	{gamma} γ	{eta} η
{kappa} κ	{lambda} λ	{mu} μ	{nu} ν
{pi} π	{theta} θ	{rho} ρ	{sigma} σ
{tau} τ	{omega} ω	{psi} ψ	

All the other characters in the Symbol font can be printed by installing Symbol font, supplying the octal code, then restoring the font, as in {symbol}{\ 245}{roman} which will print infinity, then restore Times Roman font.

21.3.4.11 Line and Symbol text formatting commands, e.g. ce

{li} = line
{da} = dashed line
{do} = dotted line
{dd} = dashed dotted line
{ce}, {ch}, {cf} = circle (empty, half filled, filled)
{te}, {th}, {tf} = triangle (empty, half filled, filled)
{se}, {sh}, {sf} = square (empty, half filled, filled)

{de}, {dh}, {df} = diamond (empty, half filled, filled)

These line and symbol formatting commands can be used to add information panels to legends, titles, etc. to identify plotting symbols.

21.3.4.12 Examples of text formatting commands

{TGF}{beta}{lower}{1}{raise} is involved
TGFβ_1 is involved

y = {x}{raise}{2}{lower} + 2
$y = x^2 + 2$

The temperature was {21}{degree}{C}
The temperature was 21°C

{pi}{r}{raise}{decrease}{2}{increase}{lower} is the area of a circle
πr^2 is the area of a circle

The {alpha}{lower}{2}{raise}{beta}{lower}{2}{raise} isoform
The $\alpha_2\beta_2$ isoform

{[Ca}{raise}{decrease}{++}{increase}{lower}{]} = {2}{mu}{M}
$[Ca^{++}] = 2\mu M$

21.4 PostScript specials

SimFIT PostScript files are designed to faciltate editing, and one important type of editing is to be able to specify text files, known as specials, that can modify the graph in an almost unlimited number of ways. This technique will now be described but, if you want to do it and you are not a PostScript programmer, do not even think about it; get somebody who has the necessary skill to do what you want. An example showing how to display a logo will be seen on page 343 and further details follow.

21.4.1 What specials can do

First of all, here are some examples of things you may wish to do with SimFIT PostScript files that would require specials.

- ❏ Replace the 35 standard fonts by special user-defined fonts.
- ❏ Add a logo to plots, e.g. a departmental heading for slides.
- ❏ Redefine the plotting symbols, line types, colours, fill styles, etc.
- ❏ Add new features, e.g. outline or shadowed fonts, or clipping to non-rectangular shapes.

When SimFIT PostScript files are created, a header subsection, called a prologue, is placed at the head of the file which contains all the definitions required to create the SimFIT dictionary. Specials can be added, as independent text files, to the files after these headings in order to re-define any existing functions, or even add new PostScript plotting instructions. The idea is is very simple; you can just modify the existing SimFIT dictionary, or even be ambitious and add completely new and arbitrary graphical objects.

21.4.2 The technique for defining specials

Any SimFIT PostScript file can be taken into a text editor in order to delete the existing header in order to save space in a large document, as done with the SimFIT manual, or else to paste in a special. However, this can also be done interactively by using the font option, accessible from the SimFIT PostScript interface. Since this mechanism is so powerful, and could easily lead to the PostScript graphics being permanently disabled by an incorrectly formatted special, SimFIT always assumes that no specials are installed. If you want to use a special, then you simply install the special and it will be active until it is de-selected or replaced by another special. Further details will be found in the on-line documentation and `w_readme` files, and examples of specials are distributed with the SimFIT package to illustrate the technique. You should observe the effect of the example specials before creating your own. Note that any files created with specials can easily be restored to the default configuration by cutting out the special. So it makes sense to format your specials like the SimFIT example specials `pspecial.1`, etc. to facilitate such retrospective editing. The use of specials is controlled by the file `pspecial.cfg` as now described. The first ten lines are Booleans indicating which of files 1 through 10 are to be included. The next ten lines are the file names containing the special code. There are ten SimFIT examples supplied, and it is suggested that line 1 of your specials should be in the style of these examples. You simply edit the file names in `pspecial.cfg` to install your own specials. The Booleans can be edited interactively from the advanced graphics PS/Fonts option. Note that any specials currently installed are flagged by the SimFIT program manager and specials only work in advanced graphics mode. In the event of problems with PostScript printing caused by specials, just delete `pspecial.cfg`. To summarise.

- ❏ Create the special you want to insert.
- ❏ Edit the file `psecial.cfg` in the SimFIT folder.
- ❏ Attach the special using the Postscript Font option.

21.4.3 Examples of PostScript specials

To clarify the structure of SIMFIT PostScript specials, just consider the code for the first three examples distributed with the SIMFIT package. The file `psecial.1` simply adds a monochrome logo, the file `psecial.2` shows how to add color, while the file `psecial.3` makes more sweeping changes to the color scheme by reversing the definitions for black and white.

❏ The PostScript special `pspecial.1`

```
%file = pspecial.1: add monochrome simfit logo to plot
gsave
/printSIMFIT {0 0 moveto (SIMFIT) show} def
/Times-Italic findfont 300 scalefont setfont
300 4400 translate
.95 -.05 0
{setgray printSIMFIT -10 5 translate} for
1 1 1 setrgbcolor printSIMFIT
grestore
%end of pspecial.1
```

❏ The PostScript special `pspecial.2`

```
%file = pspecial.2: add yellow simfit logo to plot
gsave
/printSIMFIT {0 0 moveto (SIMFIT) show} def
/Times-Italic findfont 300 scalefont setfont
300 4400 translate
.95 -.05 0
{setgray printSIMFIT -10 5 translate} for
0 0 moveto (SIMFIT) true charpath gsave 1 1 0 setrgbcolor fill grestore
grestore
%end of pspecial.2
```

❏ The PostScript special `pspecial.3`

```
%file = pspecial.3: yellow-logo/blue-background/swap-black-and-white
/background{.5 .5 1 setrgbcolor}def
background
0 0 0 4790 6390 4790 6390 0 4 pf
/c0{1 1 1 setrgbcolor}def
/c15{0 0 0 setrgbcolor}def
/foreground{c0}def
gsave
/printSIMFIT {0 0 moveto (SIMFIT) show} def
/Times-Italic findfont 300 scalefont setfont
300 4400 translate
.95 -.05 0
{setgray printSIMFIT -10 5 translate} for
0 0 moveto (SIMFIT) true charpath gsave 1 1 0 setrgbcolor fill grestore
grestore
%end of pspecial.3
```

Remember, the effects of these specials are only visible in the PostScript files created by SIMFIT and not in any direct Windows quality hardcopy.

Part 22

Scalable vector graphics (SVG)

22.1 SVG: introduction

In order to appreciate the nature and use of scalable vector graphics files (*.SVG) it is useful to review the two document types, namely bitmap and vector, that are used to archive graphs and include them into computer generated documents or web pages.

22.1.1 Bitmaps

These files (*.BMP) contain the raw information for every pixel captured by a digital photograph or displayed on a computer screen. They have the following properties.

- The larger the number of pixels then the greater the detail recorded.
- Such files are very easy to display, include in documents, or print but, unless there is a large number of pixels, then lines and curves will appear stepped and fonts will pixelate.
- For complicated portraits, landscapes, capture of microscope fields, or 3D display of molecules etc. requiring shading such files are indispensable.
- Where there are appreciable homogeneous patches such as blue sky in a landscape then considerable compression into formats such those of the joint photographic expert group (*.JPG) or portable network graphics (*.PNG) can result in smaller files, but compression is not always lossless.
- Where a bitmap has been created using antialiasing to smooth out polygons and polylines as in text, curves, or lines then compression can lead to fuzzy curves or distorted characters. Traditional computer graphics hardly ever looked good on computer screens because hard edges at an angle would show as a series of steps (a distortion known as aliasing). This made even the simplest graph – such as a sine curve, look ugly. This problem has been largely eliminated by anti-aliasing (automatically employed by SIMFIT). However a bitmap of a specific size generated using anti-aliasing, never looks its best if it is ultimately displayed at another resolution. SVG completely sidesteps this problem because the necessary anti-aliasing is performed as an image is displayed (or printed), so that data are displayed correctly. This property of being device–independent is something that sets aside vector graphics from bitmap graphics.

22.1.2 Vector graphics

Scientific graphs largely consist of axes, curves, and plotting symbols, with small amounts of text, and vector graphic files simply contain the mathematical data such as coordinates necessary to reproduce the graph at any degree of magnification or compression without loss of information. Here is a summary of properties for the two main vector graphics files; encapsulated PostScript (*.EPS), and scalable vector graphics (*.SVG).

- The files are in text format, which means they can easily be edited retrospectively in text editors such as **notepad** in order to change, titles, legends, line types, plotting symbols, or colours.
- They are device independent so there is no loss of information on expansion or compression.
- They can be imported into LaTeX documents or include LaTeX code, so that high quality mathematical formulas and chemical structures can be incorporated.
- The free program **Ghostscript** can be used to convert EPS files into other formats, and similarly **Inkscape** can be used to visualize and transform SVG files.
- While EPS is the main import format for LaTeX documents, SVG is the recommended format for scientific graphs on the internet.

22.1.3 Bogus vector files

Note that many applications claim to transform bitmap and compressed bitmap files into vector files without loss of significant information, but this is almost impossible except for fairly simple images. Such applications usually just exploit a weakness in vector files that allows bitmaps to be inserted giving bogus vector files that are wrappers containing bitmaps. Similarly portable document files (*.PDF) were developed from PostScript and retain many PostScript features that can be exploited. For instance, using the SIMFIT interface to GhostScript to transform SIMFIT EPS files into PDF files yields PDF files that are effectively device independent, whereas using Windows to distil files into PDF merely creates bitmap files.

22.1.4 Using SVG files in SIMFIT

The SVG format is very comprehensive as it has been specifically developed to be versatile for web use. For that reason few applications implement the whole standard and SIMFIT is no exception, so it must be emphasized that SIMFIT will only accept and manipulate SVG files according to the graph plotting functionality provided by Silverfrost FTN95 Clearwin+. This interface was written by David Bailey and it provides the following SVG file functionality for SIMFIT users.

- The SVG files can be used on the web and opened by browsers such as **firefox**.
- The SVG files can be displayed and edited retrospectively by the program **EditSVG**.
- The SVG files created by other applications may cause warning messages, but in some circumstances files may use acceptable subsets of SVG facilities. To explore this option just open the SVG file with a text editor and splice the option Clearwin_output="1" into the line beginning <svg. It is easy to get this wrong, so it is probably best to file the result to a different file name.

22.1.5 Editing SVG files in SIMFIT

The functionality provided by procedure **EditSVG** is now listed.

1. The only file types that can be used in this program are:

   (A) *.SVG files created by SIMFIT, and other Clearwin+ output from Silverfrost FTN95 programs.

   (B) *.TEX files describing mathematical equations or chemical formulas. Such *.TEX files can be used to generate internal *.SVG files if **latex.exe**, **dvips.exe**, and **dvisvgm.exe** are on the path. For instance if users have a recent version of MikTeX available.

2. Files can be input from the console, by drag and drop, or by using library files.
3. Images can be enlarged or reduced by "right clicking" on the image.
4. Images can be freely positioned by dragging a window from any point on its surface. Using the view menu it is possible to add a graticule with optional "snap to nearest graticule intersection" facility. The graticule does not remain on the finished image.

5. After manipulating a file or a set of files the resulting composite image can be written out to a *.SVG, or *.PNG, file, or even to a *.ISVG rebuild–image file.

6. It can be used to create strict collages where every image is snapped to the nearest grid point, freestyle collages where images can be arranged in arbitrary positions, or overlays where smaller images can be inserted into larger ones.

22.1.6 Using LaTeX

Some examples of how to use these procedures within the SimFIT package from version 7.5.0 onwards using the test files provided follow. However the procedure used to create the SVG files using LaTeX should be noted.

To enlarge on the use of LaTeX it must be emphasized that the procedure to use LaTeX depends on whether the user has a fully functioning LaTeX installation on their machine. So there are three distinct cases.

1. **The direct method**

   There is a LaTeX installation so the user prefers to input a *.TEX file directly into program **EditSVG** whereupon the *.TEX file will be processed and the image will appear in the main window.

2. **The indirect method**

   There is a LaTeX installation but the user prefers to use the command line technique described subsequently to transform the *.TEX file into *.DVI then use **dvisvgm** to transform this into a stand alone *.SVG file. Note that filenames used in this procedure must be local files with no spaces in the file name.

3. **The remote user method**

   There is no LaTeXinstallation so a known LaTeX user will have to perform the transformation. Alternatively, a stand–alone *.SVG file, such as the demonstration files distributed with SimFIT will have to be used.

LaTeX is designed to create documents and, because of this, care is needed to remove much of the header information in order to create simple images that can be imported into **EditSVG**. This is much the same as the steps required to create a *.EPS from from a *.PS file but using the following commands.

To make DVI: use `latex myfile.tex` to create `myfile.dvi`.

To make PS: use `dvips myfile.dvi` to create `myfile.ps`.

To make SVG: use `dvisvgm -E myfile.ps` to create `myfile.svg`.

The argument `-E` indicates that a PostScript file is to be input. Note that white space can be trimmed from the resulting SVG file by **EditSVG**, or alternatively by using **inkscape** or **GSview** (e.g. Version 5) to transform `myfile.ps` into `myfile.eps`, where the `BoundingBox` will automatically remove white space.

22.1.7 Important differences between EPS and SVG files

From within any SimFIT graph it is possible to create *.EPS files by first selecting [PS] then choosing [File], or to create *.SVG files by first selecting [Win] then choosing the [SVG] option. Any *.SVG file created in this way will be a fairly accurate representation of the display, as will any *.EPS files, except that there will be small but significant differences between them. That is unavoidable because the fonts used may differ slightly as the *.SVG file will use Windows fonts whereas the *EPS file will use PostScript fonts. In addition SimFIT allows users to set different global line thickness for EPS and SVG files as line thickness do not scale in exactly the same way in EPS and SVG files. Nevertheless it is useful to know how to create a *.SVG file from a *.EPS file and vice versa.

Fortunately such transformations can readily be carried out due to the widespread availability of Open Source programs such as **Inkscape** and **Cairo**. The usefulness of such transformations will be explained in subsequent tutorial sections.

22.2 SVG: Importing LaTeX maths equations

Sometimes it is required to use LaTeX to display a mathematical equation inside a scientific plot, and this document describes how to do this for the normal distribution cumulative distribution function $\Phi(x)$. Note that all the files mentioned in this document are distributed as SimFIT test files so that users simply wishing to create the final composed document can proceed directly to the last section describing how to use **EditSVG**.

22.2.1 The TEX source

This is the code contained in the file `latex_maths_equation.tex`

```
\documentclass[12pt]{article}
\usepackage{amsmath,bm}
\pagestyle{empty}
\begin{document}
\Large
\[
 \frac{1}{\sigma \sqrt{2\pi}}
   \int\limits_{-\infty}^x
   \exp -\left\{
  \frac{1}{2} \left( \frac{t-\mu}{\sigma} \right)^2 \right\}\,dt
\]
\end{document}
```

which displays the mathematical definition of $\Phi(x)$ as follows.

$$\frac{1}{\sigma \sqrt{2\pi}} \int\limits_{-\infty}^x \exp -\left\{ \frac{1}{2} \left(\frac{t-\mu}{\sigma} \right)^2 \right\}\,dt$$

In order to import this formula into a graph using **EditSVG** this code must be used to create the corresponding SVG file `latex_maths_equation.svg`, the overall process being the following sequence of commands.

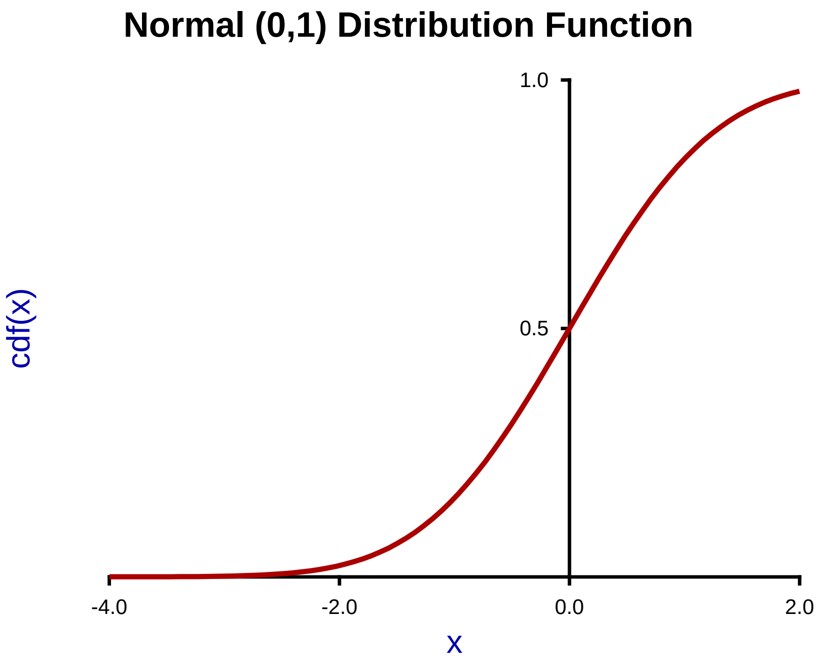
- **latex** `latex_maths_equation.tex`
- **dvips** `latex_maths_equation.dvi`
- **dvisvgm** `-E latex_maths_equation.ps`

The file `latex_maths_equation.svg` created is then ready to be imported into **EditSVG** but, alternatively, the source file `latex_maths_equation.tex` can be opened in or dragged and dropped directly onto **EditSVG** if there is a local installation of LaTeX.

It should be realized that, when using LaTeX in this way to create a SVG file, the command line must be used from a folder containing the *.TEX file required as a local file and not as a fully qualified path–filename to a remote source file. The program **EditSVG** circumvents this issue when importing LaTeX source by creating local copies of all files.

22.2.2 Creating the plot file

The file `latex_maths_plot.svg` with the $\Phi(x)$ profile to be used looks like this before the equation is added.



This figure was created using **makmat** by selecting to display the normal cumulative distribution $\Phi(x)$ with $\mu = 0$ and $\sigma^2 = 1$ for $-4 \le x \le 2$, and then transferring the resulting plot into the program **simplot** using the [Advanced] option to manipulate the title, legends, line–widths, and colors, etc.

Users wishing to avoid this process can simply read the SimFIT metafile `latex_maths_plot.metafile` directly into the SimFIT program **simplot**, or the SimDEM program **simdem70**.

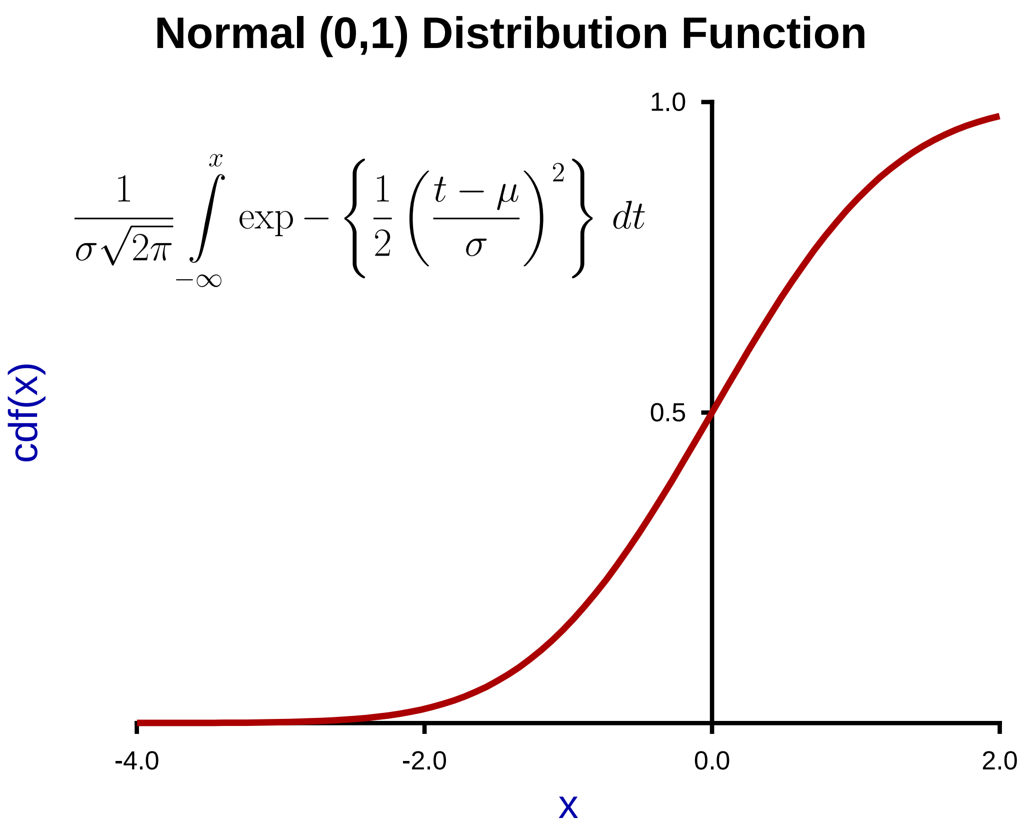
In either case the file is then saved as `latex_maths_plot.svg` using the [Win] or [SVG] option.

22.2.3 Joining the SVG files using EditSVG

First open program **EditSVG** then input the test file `latex_maths_plot.svg` to act as a background, then there two possible options.

1. Input the test file `latex_maths_equation.svg` directly; or
2. read in the test file `latex_maths_equation.tex` which will then be used by LaTeX to generate an internal copy of `latex_maths_equation.svg`.

Finally, just use the mouse to move the equation into position and alter the scaling as required to obtain the final plot saved as `latex_maths.svg` and shown next.



22.2.4 Summary of files described in this section

The programs referred to in this document are as follows.

1. **InkScape** is an OpenSource program that takes in SVG files and can write out EPS and other files.

2. **EditSVG** is a SIMFIT and SIMDEM program that takes in SVG or TEX files and writes out SVG and other files.

3. **editPS** is a SIMFIT and SIMDEM program that takes in EPS files and writes out only EPS files.

4. The SIMFIT program **simplot** and the SIMDEM program **simdem70** take in SIMFIT metafiles and write out either SVG or EPS files.

Further, the SIMFIT test files (*.TEX and *.SVG) described in this document that can be used by program **EditSVG**, and those (*.EPS) that can be used by program **editPS** are now listed.

File name	Data included
`latex_maths_plot.metafile`	SIMFIT or SIMDEM metafile to create the plot without any equation
`latex_maths_equation.tex`	LaTeX source file for the maths equation with no plot
`latex_maths_equation.svg`	SVG file containing the formula only
`latex_maths_plot.svg`	SVG file containing the plot only
`latex_maths.svg`	SVG file containing both the equation and plot
`latex_maths_equation.eps`	EPS file containing the formula only
`latex_maths_plot.eps`	EPS file containing the plot only
`latex_maths.eps`	EPS file containing both the equation and plot

22.3 SVG: Importing LaTeX chemical formulas

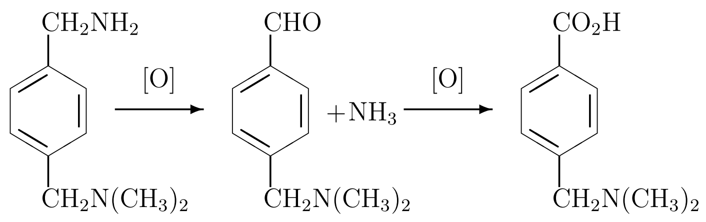
Sometimes it is required use LaTeX to display chemical structures inside a scientific plot, and this document describes how to do this using a condensed scheme for the oxidation of p-dimethylaminomethylbenzylamine. Note that all the files mentioned in this document are distributed as SimFIT test files so that users simply wishing to create the final composed document can proceed directly to the last section describing how to use **EditSVG**

22.3.1 The TEX source

This is the code contained in the file `latex_chemical_formula.tex`

```
\documentclass[12pt]{article}
\usepackage{carom}
\pagestyle{empty}
\begin{document}
{\begin{picture}(3000,600)(0,0)
\thicklines
\put(0,0){\bzdrv{1==CH$_{2}$NH$_{2}$;4==CH$_{2}$N(CH$_{3}$)$_{2}$}}
\put(700,450){\vector(1,0){400}}
\put(820,550){[O]}
\put(1000,0){\bzdrv{1==CHO;4==CH$_{2}$N(CH$_{3}$)$_{2}$}}
\put(1650,400){+}
\put(1750,400){NH$_{3}$}
\put(2000,450){\vector(1,0){400}}
\put(2120,550){[O]}
\put(2300,0){\bzdrv{1==CO$_{2}$H;4==CH$_{2}$N(CH$_{3}$)$_{2}$}}
\end{picture}}
\end{document}
```

which displays like this.



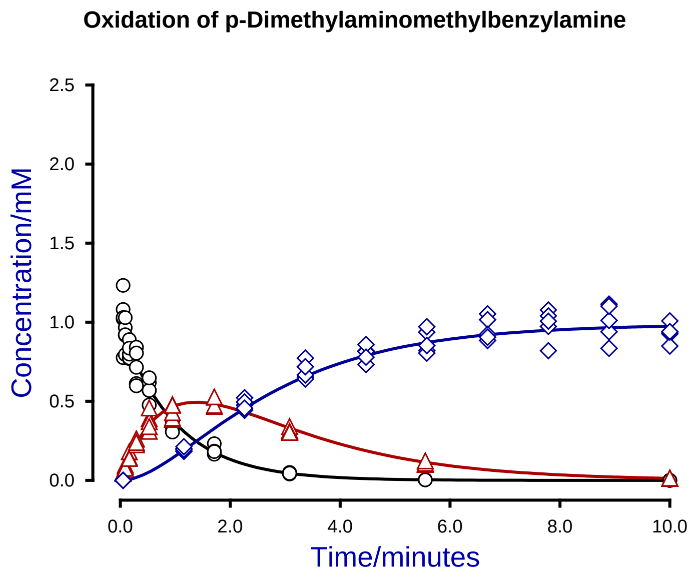
To import this formula into a graph using **EditSVG**, `latex_maths_equation.svg` can be made using the following commands, or `latex_maths_equation.tex` can be input directly into **EditSVG**.

- `latex latex_chemical_formula.tex`
- `dvips latex_chemical_formula.dvi`
- `dvisvgm -E latex_chemical_formula.ps`

The file `latex_chemical_formula.svg` created is then ready to be imported into **EditSVG** but, alternatively, the source file `latex_chemical_formula.tex` can be opened in or dragged and dropped directly onto **EditSVG** if there is a local installation of LaTeX. It should be realized that, when using LaTeX in this way to create a SVG file, the command line must be used from a folder containing the *.TEX file required as a local file and not as a fully qualified path–filename to a remote source file. The program **EditSVG** circumvents this issue when importing LaTeX source by creating local copies of all files.

22.3.2 Creating the plot file

The file `latex_chemical_plot.svg` with the time course data to be used looks like this before the equation is added.



This figure was created using **qnfit** fit three data sets for the consecutive reaction scheme

$$A \rightarrow B \rightarrow C$$

in the SIMFIT test library file `consec3.tfl`, then fitted using the model in the model file `consec3.mod`.

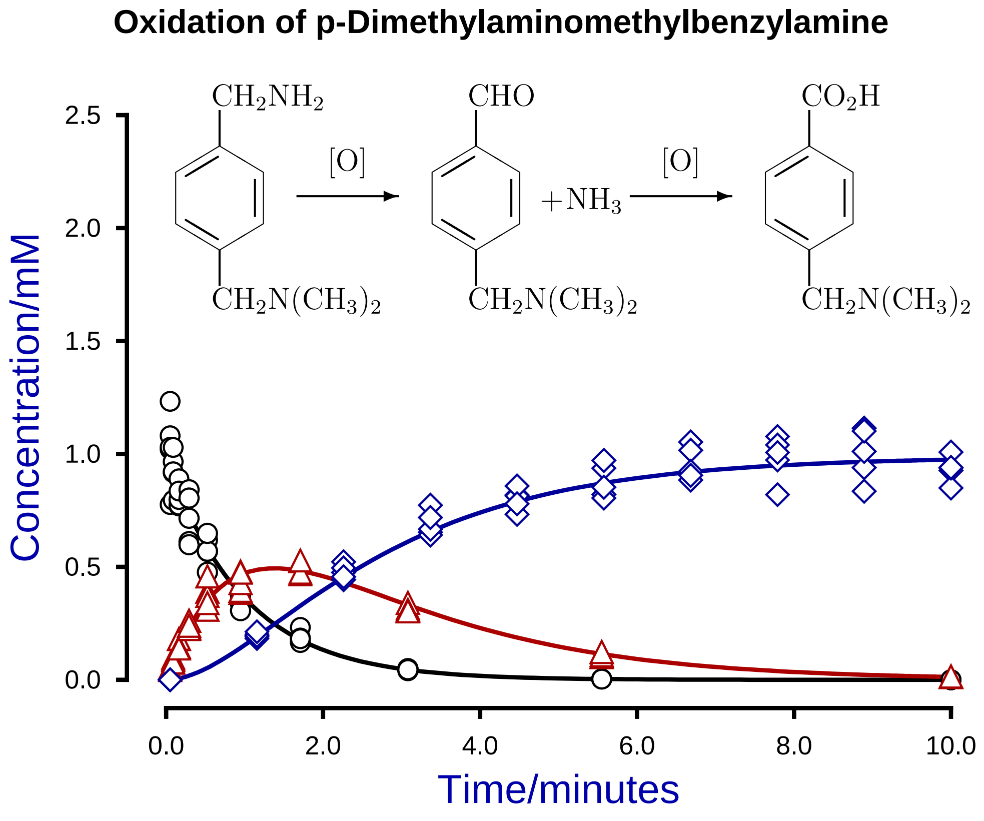
After manipulating the line thicknesses, title, legend, and colors, the files `latex_chemical_plot.svg`, and `latex_chemical_plot.eps` were created to archive the graph. In addition the SIMFIT metafile `latex_chemical_plot.metafile` was saved so that users wishing generate this plot can easily do so using the SIMFIT program **simplot** or the SIMDEM program **simdem70**. Users wishing to avoid this process can simply read the SIMFIT metafile `latex_maths_plot.metafile` directly into the SIMFIT program **simplot**, or the SIMDEM program `simdem70`.

22.3.3 Joining the SVG files using EditSVG

Open program **EditSVG** then input the test file `latex_chemical_plot.svg`. Then there are two options.

1. Input the test file `latex_chemical_formula.svg` directly; or
2. read in the test file `latex_chemical_formula.tex` which will then be used by LaTeX to generate an internal copy of `latex_chemical_formula.svg`.

Finally, just use the mouse to move the equation into position and alter the scaling as required to obtain the final plot saved as `latex_chemistry.svg` and shown next.



22.3.4 Summary of files used in this section

The programs referred to in this document are as follows.

1. **InkScape** is an OpenSource program that takes in SVG files and can write out EPS and other files.
2. **EditSVG** is a SimFIT and SimDEM program that takes in SVG or TEX files and writes out SVG and other files.
3. **editPS** is a SimFIT and SimDEM program that takes in EPS files and writes out only EPS files.
4. The SimFIT program **simplot** and the SimDEM program **simdem70** take in SimFIT metafiles and write out either SVG or EPS files.

Further, the SimFIT test files (*.TEX and *.SVG) described in this document that can be used by program **EditSVG**, and those (*.EPS) that can be used by program **editPS** are now listed.

File name	Data included
`latex_chemical_plot.metafile`	SimFIT or SimDEM metafile to create the plot without any equation
`latex_chemical_formula.tex`	LaTeX source file for the maths equation with no plot
`latex_chemical_formula.svg`	SVG file containing the formula only
`latex_chemical_plot.svg`	SVG file containing the plot only
`latex_chemistry.svg`	SVG file containing both the formula and plot
`latex_chemical_formula.eps`	EPS file containing the formula only
`latex_chemical_plot.eps`	EPS file containing the plot only
`latex_chemistry.eps`	EPS file containing both the formula and plot

22.4 SVG: Importing SVG files into SVG files

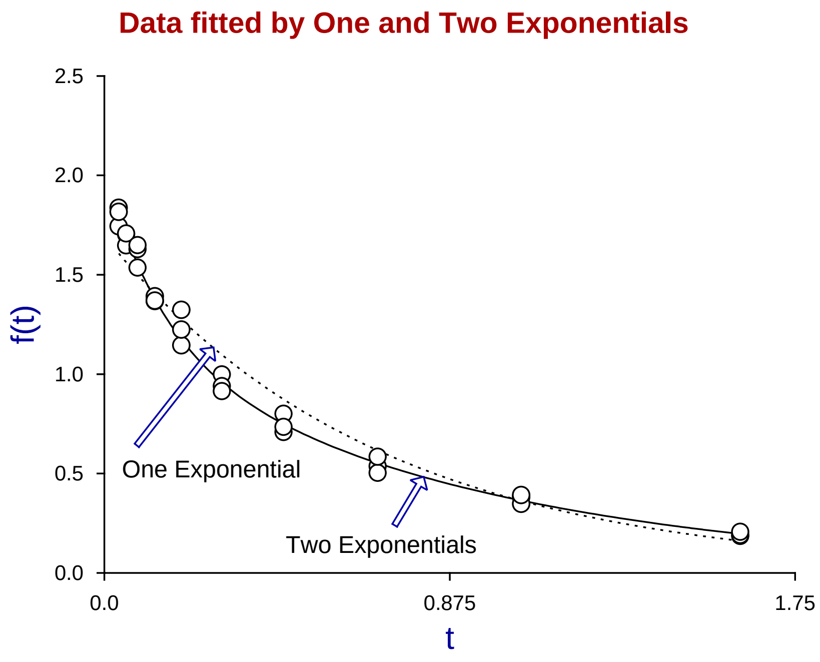
Sometimes it is required to import one SVG file into another SVG file,for example to create overlays, insets, or collages. This document describes how to do this using program **EditSVG** when fitting one then two exponential functions to a data set and plotting the best–fit curves to evaluate the improvement in fit.

22.4.1 Fitting exponential functions

Using SIMFIT program **exfit** in the default mode to analyze data in the test file `exfit.tf4` for models of orders 1 and 2 fits the following exponential functions sequentially.

$$f_1(t) = Ae^{-Bt}$$
$$f_2(t) = \alpha_1 e^{-k_1 t} + \alpha_2 e^{-k_2 t}$$

Program **exfit** then outputs goodness of fit criteria and statistical tests to see if there is sufficient statistical evidence to accept the need to fit the additional parameters required by the two exponential model. In addition the following graph is plotted to illustrate the goodness of fit for both models.



The following SIMFIT test files are provided to reproduce this fit.

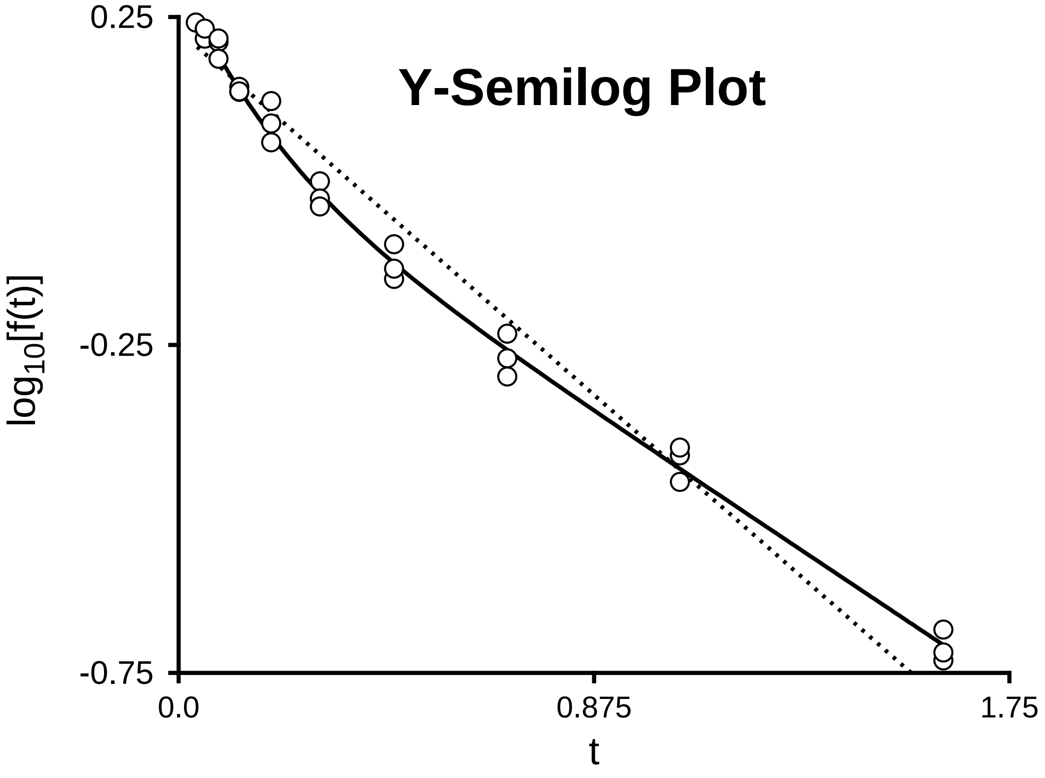
- `exfit.t4f`, the data for input into program **exfit**
- `exfit_normalplot.eps`, the above plot in EPS format for input into program **editps**
- `exfit_normalplot.svg`, the above plot in SVG format for input into program **EditSVG**
- `exfit_normalplot.metafile`, a metafile to create this plot in SIMFIT program **simplot** or SIMDEM program **simdem70**

The file `exfit_normalplot.svg` was also created at this stage to archive this plot.

22.4.2 Creating the log transform

Of course the previous plot was created from within program **exfit** by simply transferring the plot into SIMFIT advanced graphics mode then editing. Now one technique that can be used to check the relative fit for two exponentials as opposed to one is to plot a semilogarithm plot, which results in linearizing the single exponential function but not the double exponential model.

Within the advanced graphics environment this transformation was selected to produce the next plot.



As the intention was to insert this graph into the previous one the graph was edited as follows

- Suppressing the title
- Changing the legends
- Adding a subsidiary title within the graph
- Making the lines thicker so they do not look too narrow when the plot is reduced in size. As the figure is to be reduced by a factor of two, the line thicknesses were doubled. Sometimes it is useful to enlarge the legends or to replace using a bold font, and even the numbers can be enlarged if required.

The following SIMFIT test files are provided to reproduce this fit.

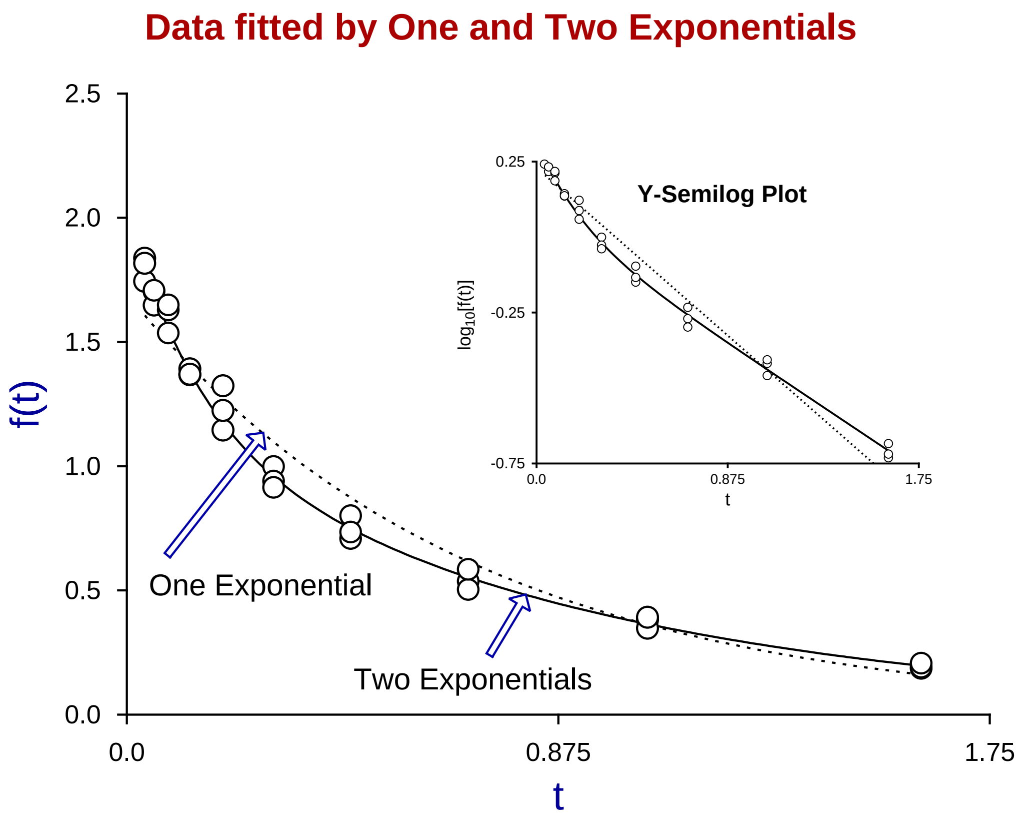
- `exfit_logplot.eps`, the above plot in EPS format for input into program **editps**
- `exfit_logplot.svg`, the above plot in SVG format for input into program **EditSVG**
- `exfit_logplot.metafile`, a metafile to create this plot in SIMFIT program **simplot** or SIMDEM program **simdem70**

The file `exfit.svg` was also created at this stage to archive the compound plot, as described next.

22.4.3 Joining the SVG files using EditSVG

Open program **EditSVG** then input the test file `exfit_normalplot.svg`.

Now input the file `exfit_logplot.svg` then just use the mouse to move the equation into position and alter the scaling as required to obtain the final plot saved as `exfit.svg` and shown next.



22.4.4 Summary of files used in this section

Finally, the SIMFIT test files described in this document that can be used to create this plot are now listed.

File name	Data included
`exfit_normalplot.metafile`	SIMFIT or SIMDEM metafile to create the normal plot
`exfit_logplot.metafile`	SIMFIT or SIMDEM metafile to create the log plot
`exfit_normal.svg`	SVG file containing the normal plot
`exfit_logplot.svg`	SVG file containing the log plot
`exfit.svg`	SVG file containing both the final compound plot
`exfit_normalplot.eps`	EPS file containing the normal plot
`exfit_logplot.eps`	EPS file containing the log plot
`exfit.eps`	EPS file containing the final compound plot
`exfit.tf4`	Test file containing the data for fitting

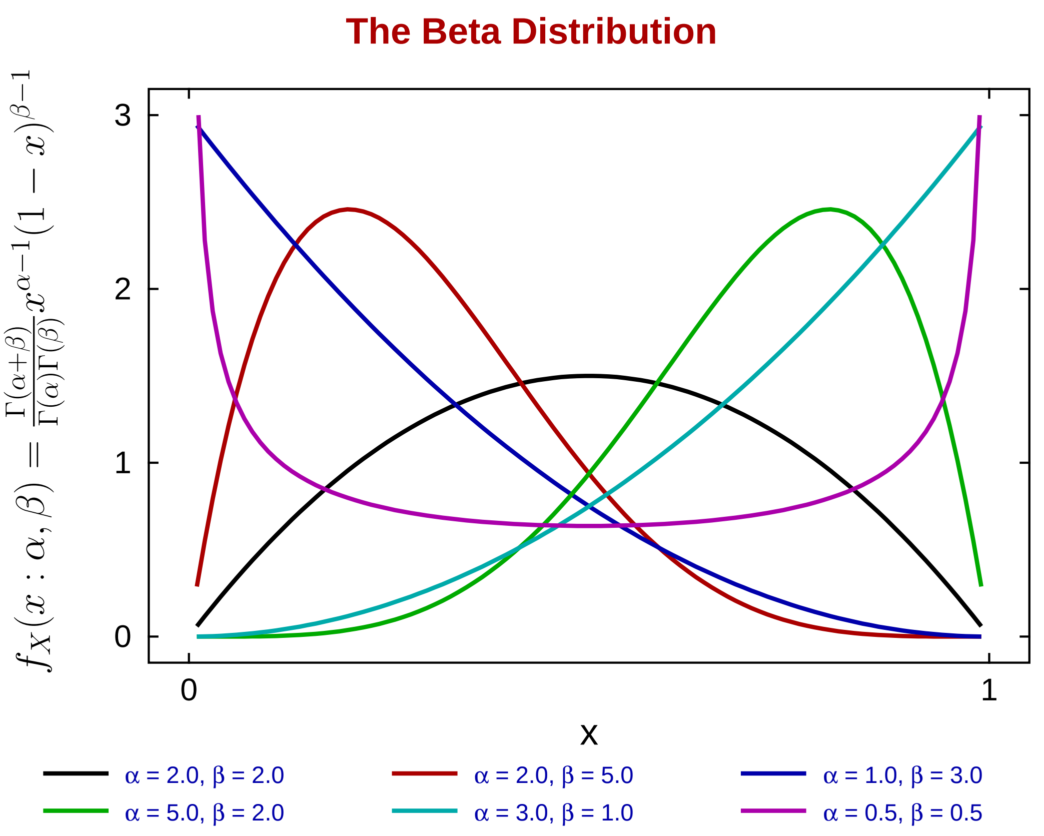
22.5 SVG: Using LaTeX to label SVG y axes

Sometimes it is required to use LaTeX to display a mathematical equation but with the formula rotated so it can be used as the y axis label inside a scientific SVG plot, and this document describes how to do this using the beta probability distribution as an example.

Note that all the files mentioned in this document are distributed as SimFIT test files so that users simply wishing to create the final composed document can proceed directly to the last section describing how to use **EditSVG**.

22.5.1 The beta probability density function

Consider, for example, the wide variety of shapes possible for the beta probability distribution as the two positive parameters α and β are varied as shown next.



This distribution is widely used in data analysis where a unimodal distribution is required as an empirical equation to model data as positive frequencies for a variable x that can be scaled into the range $0 \le x \le 1$.

The great advantage of this distribution is that for positive parameters α and β a great variety of shapes can be generated to illustrate and quantify skew and kurtosis with frequency histograms.

22.5.2 The LaTeX source

This is the LaTeX code contained in the file `latex_beta_pdf.tex` to generate the rotated formula.

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\usepackage{graphicx}
\pagestyle{empty}
\begin{document}
\Large
\rotatebox{90}{$ f_X(x:\alpha,\beta) = \frac{\Gamma(\alpha +
\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha - 1}(1 - x)^{\beta - 1}$}
\end{document}
```

which displays the mathematical definition of the beta function (shown before rotation) as follows.

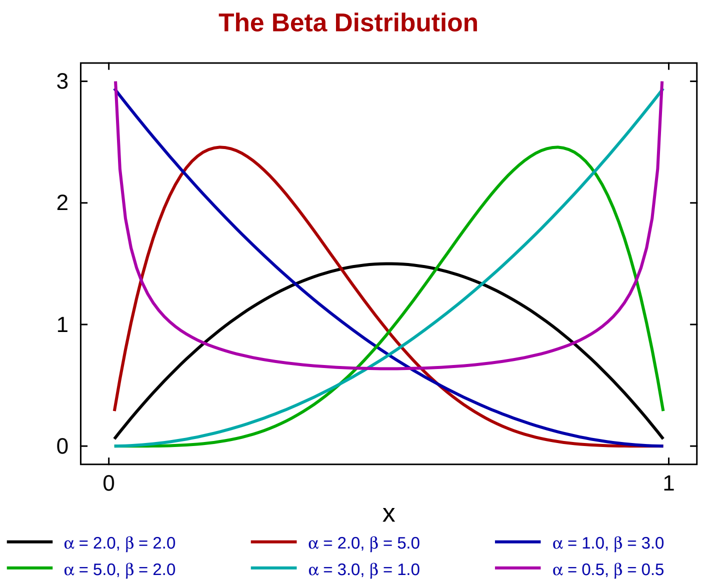
$$f_X(x:\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$

In order to import this formula into a graph using **EditSVG** the code must be used to create the corresponding SVG file `latex_beta_pdf.svg`, the overall process being the following sequence of commands.

- **latex** `latex_beta_pdf.tex`
- **dvips** `latex_beta_pdf.dvi`
- **dvisvgm** `-E latex_beta_pdf.ps`

The file `latex_beta_pdf.svg` created is then ready to be imported into **EditSVG** but, alternatively, the source file `latex_beta_pdf.tex` can be opened in or dragged and dropped directly onto **EditSVG** if there is a local installation of LaTeX. When using LaTeX in this way to create a SVG file, the command line must be used from a folder containing the *.TEX file required as a local file and not as a fully qualified path–filename to a remote source file. The program **EditSVG** circumvents this issue when importing LaTeX source by creating local copies of all files.

22.5.3 Creating the plot file



The file `beta_pdf_plot.svg` with the $f_X(x : \alpha, \beta)$ to be used looks like the previous figure before the equation is added.

This figure was created using the SimFIT program **makmat** by selecting to display the beta distribution $f_X(x : \alpha, \beta))$ with various values for the positive parameters α and β over the range $0.01 \le x \le 0.99$ so as to avoid the poles at either extreme. Users wishing to avoid this process can simply read the SimFIT metafile `beta_pdf_plot.metafile` directly into the SimFIT program **simplot**, or the SimDEM program **simdem70**. In either case the file is then saved as `beta_pdf_plot.svg` using the [Win] or [SVG] option.

22.5.4 Joining the SVG files using EditSVG

First open program **EditSVG** then input the test file `beta_pdf_plot.svg` to act as a main plot, then there are two possible options.

1. Input the test file `latex_beta_pdf.svg` directly; or
2. read in the test file `latex_beta_pdf.tex` which will then be used by LaTeX to generate an internal copy of `latex_beta_pdf.svg`.

Finally, just use the mouse to move the equation into position and alter the scaling as required to obtain the final plot saved as `beta_pdf_with_equation.svg` shown previously at the start of this document.

22.5.5 Summary

The programs referred to in this document are as follows.

1. **EditSVG** is a SimFIT and SimDEM program that takes in SVG or TEX files and writes out SVG and other files.
2. The SimFIT program **simplot** and the SimDEM program **simdem70** take in SimFIT metafiles and write out either SVG or EPS files.

Further, the SimFIT test files (*.TEX and *.SVG) described in this document that can be used by program **EditSVG**, and those (*.EPS) that can be used by program **editPS** are now listed.

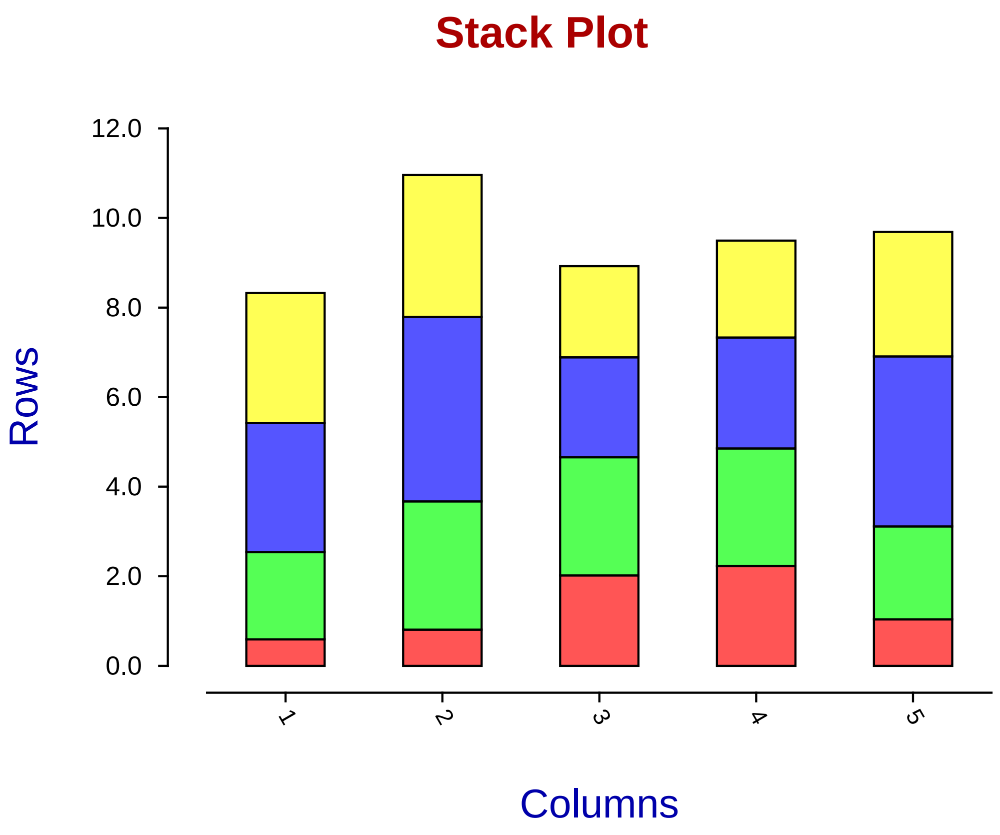
File name	Data included
`beta_pdf_plot.metafile`	SimFIT or SimDEM metafile to create the plot without any equation
`latex_beta_pdf.tex`	LaTeX source file for the beta_pdf equation with no plot
`latex_beta_pdf.svg`	SVG file containing the formula only
`beta_pdf_plot.svg`	SVG file containing the plot only
`beta_pdf_with_equation.svg`	SVG file containing both the equation and plot
`beta_pdf_with_equation.eps`	EPS file containing both the equation and plot only
`latex_beta_pdf.eps`	EPS file containing formula only
`beta_pdf_plot.eps`	EPS file containing the plot only

22.6 SVG: Editing using text editors, e.g., Notepad

It is frequently convenient to edit SVG files retrospectively, usually in order to change sizes, titles, legends, labels, line–types, line–widths, and colors, etc. Fortunately, SVG files, like EPS files, are in ASCII text format so they can be edited using any text editor that supports UTF8 characters, such as the Windows program **notepad** or better **notepad++**. The actual format is in XML which is similar to HTML but, in addition, the SVG files created by SIMFIT have been designed with editing in mind. For instance, each individual markup code starts on a new line.

22.6.1 Titles and Legends

As a simple example consider the following stacked bar chart created by importing the SIMFIT metafile `stack_plot.metafile` into SIMFIT program **simplot** or SIMDEM program **simdem70**.



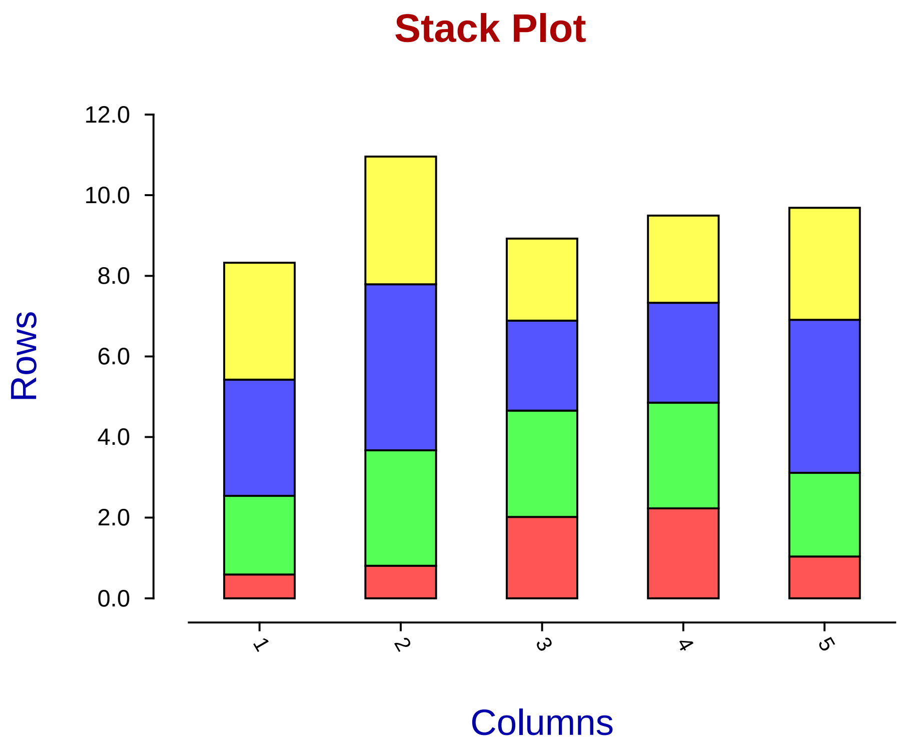
On searching for the string `"Stack Plot"` in the file `stack_plot.svg` using a text editor we find the following markup at line 34, where the tokens have been shown on separate lines for clarity.

```
<text
x="420.00" y="44.72"
font-family="ARIAL"
font-size="47px"
font-weight="700"
fill="rgb(170,0,0)">
Stack Plot</text>
```

Note the key markup codes used in this line.

- <text ... ></text>
- font-family=
- font-size=
- font-weight=
- fill=

We might think the title could be more explanatory and would be better using a more subdued coloring given the amount of color already associated with the stack segments. So, after some editing in **notepad++**, re-displaying using **firefox** gives the next graph.



Lines 34, 35, and 36 now look like this (with line numbers added for clarity).

```
34 <text x="330.00" y="44.72" font-family="ARIAL" font-size="47px"
    font-weight="700" fill="rgb(0,0,170)">Bar Chart in Stacked Format</text>
35 <text x="504.00" y="788.92" font-family="ARIAL" font-size="42px"
    font-weight="400" fill="rgb(0,170,0)">Columns</text>
36 <text x="68.00" y="449.44" font-family="ARIAL" font-size="44px"
    font-weight="400" transform="rotate(270.00,68.00,460.00)"
    fill="rgb(0,170,0)">Rows</text>
```

The following editing will be apparent on inspection.

Line 34

The original short title "Stack Plot" has been changed to the longer title "Bar Chart in Stacked Format", the color has been changed from red rgb(170,0,0) to blue rgb(0,0,170), and the x-coordinate has been changed from 420.0 to 330.00 (so that the title remains centralized over the plot).

Line 35

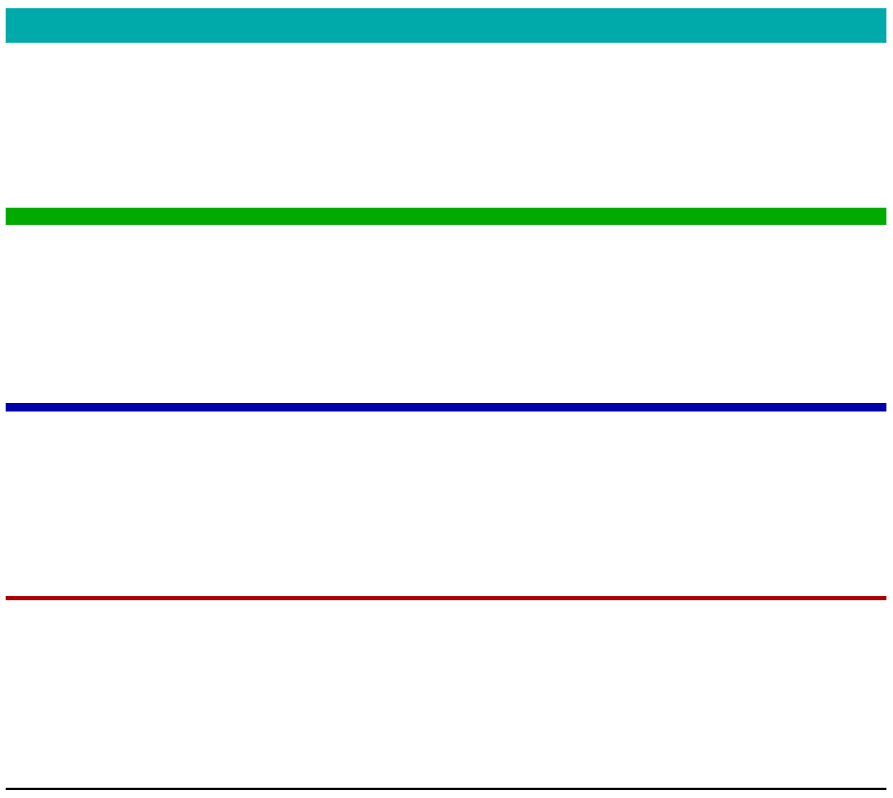
The x-legend color has been changed from blue rgb(0,0,170) to green rgb(0,170,0)

Line 36

The y-legend color has been changed from blue rgb(0,0,170) to green rgb(0,170,0)

22.6.2 Lines and Curves

Consider the next plot with five lines, each in a different color and line width.



This plot can easily be created by reading the SIMFIT metafile `lines.metafile` into SIMFIT program **simplot** or SIMDEM program **simdem70**.

Note that in this plot the lines were created by drawing a polyline between two points which could easily have been drawn as a simple line. However a polyline was drawn as it would be more usual to change the line type, thickness, and color for a smooth curve which would then have made it more difficult to comprehend with the presence of a large number of coordinates, whereas only the end points are needed for a straight line.

The point is that, in order to search for a graphical object in a SVG file, the the markup code would be used. For instance, searching for the text string `<polyline points=` in the file `lines.svg` locates the following code on a single line, but here broken up into separate tokens for clarity.

```
<polyline points="196.81,823.62 1095.69,823.62"
style="fill:none;stroke:rgb(0,0,0) ;stroke-width:1.84"
stroke-linecap="butt"
stroke-linejoin="round" />
```

To understand this code the following summary is presented.

1. `<polyline points=  ...  />`

   This markup section starting with < and ending with /> defines all the properties of the polyline.

2. `style=`

   Here all the details of the curve to be drawn are to be found.

3. `:rgb(0,0,0)`

   This color convention is exactly as used for HTML, where red, blue, and green color components are defined on a scale from 0 to 255.

4. `stroke width:`

   This defines the line width in pixels.

5. `stroke-linecap=`

   This defines the way the ends of curves are finished off.

6. `stroke-linejoin=`

   This is where the way that the sections of the polyline making up the curve are to be connected together is defined.

In fact the code for drawing all five lines is as follows , where dots . . . are used to indicate the text omitted for clarity.

```
... style="fill:none;stroke:rgb(0,0,0)     ;stroke-width:1.84"  ...
... style="fill:none;stroke:rgb(170,0,0)   ;stroke-width:2.00"  ...
... style="fill:none;stroke:rgb(0,0,170)   ;stroke-width:4.00"  ...
... style="fill:none;stroke:rgb(0,170,0)   ;stroke-width:8.00"  ...
... style="fill:none;stroke:rgb(0,170,170) ;stroke-width:16.00" ...
```

Note that the order of these commands is in the direction from bottom to top, so that the first line (stroke-width:1.84) refers to the bottom line, while the last command (stroke-width:16) refers to the top line.

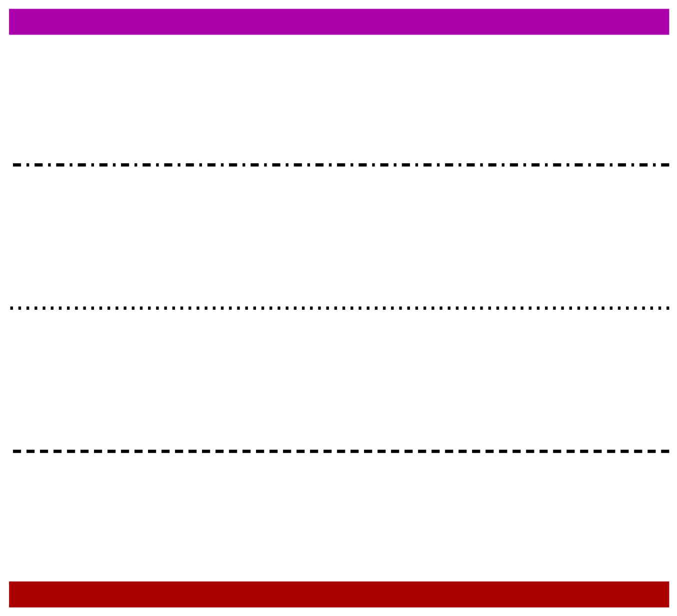
Another change that is often required is to swap between solid lines and dashed, dotted, or dash–dotted lines which requires the definition of a `stroke-dasharray` parameter as follows.

- `stroke-dasharray="18.00,12.00"` for dashed lines
- `stroke-dasharray="6.00,12.00"` for dotted lines
- `stroke-dasharray="18.00,12.00,6.00,12.00"` for dash–dotted lines

To demonstrate this technique, note that replacing this section of `lines.svg` by the next section

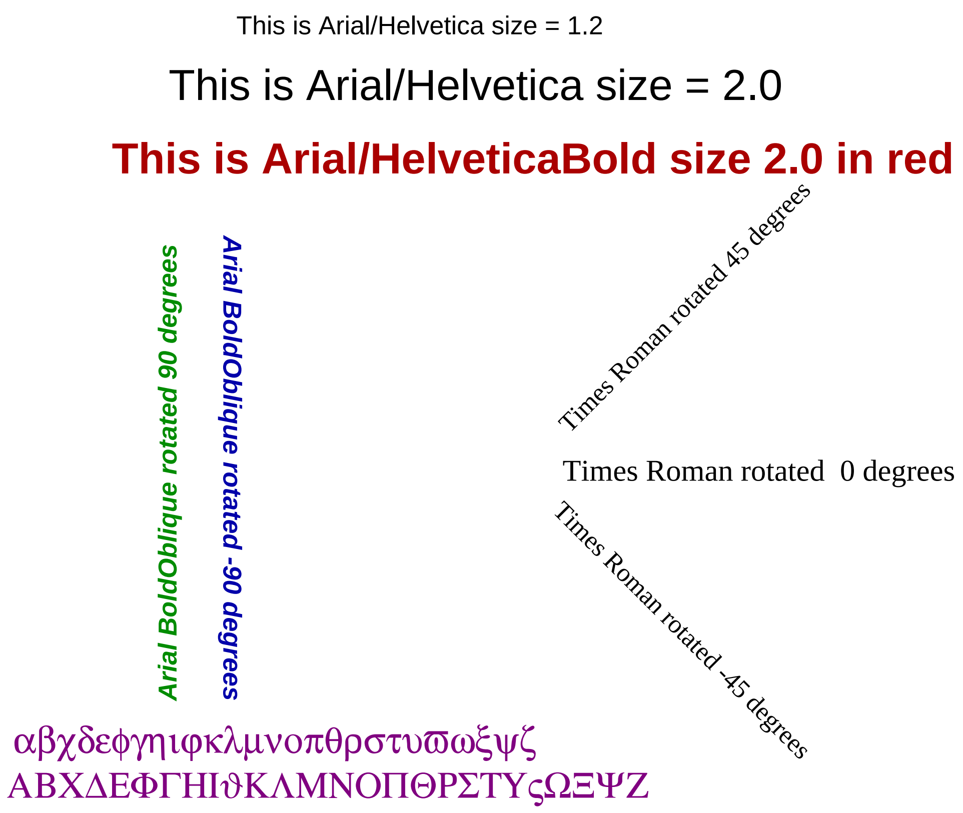
```
:rgb(170,0,0)   ;stroke-width:16.00"
:rgb(0,0,0)     ;stroke-width:2.00" stroke-dasharray="18.00,12.00"
:rgb(0,0,0)     ;stroke-width:2.00" stroke-dasharray="6.00,12.00"
:rgb(0,0,0)     ;stroke-width:2.00" stroke-dasharray="18.00,12.00,6.00,12.00"
:rgb(170,0,170) ;stroke-width:16.00"
```

generates the following graph with the dashed, dotted, and dash–dotted lines colored black.



22.6.3 Character Strings and Fonts

A common need is to reposition a character string, to edit the string, to change the font size, or to change the colour. Consider, for instance, the following diagram.


This is Arial/Helvetica size = 1.2
This is Arial/Helvetica size = 2.0
This is Arial/HelveticaBold size 2.0 in red
Arial BoldOblique rotated 90 degrees
Arial BoldOblique rotated -90 degrees
Times Roman rotated 45 degrees
Times Roman rotated 0 degrees
Times Roman rotated -45 degrees
αβχδεφγηιϕκλμνοπθρστυϖωξψζ
ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ

Here is the first character string with Arial/Helvetica at size 1.2, but broken into separate tokens for clarity.

```
<text x="289.00" y="62.48"
font-family="ARIAL"
font-size="23px"
font-weight="400"
fill="rgb(0,0,0)"
>This is Arial/Helvetica size = 1.2
</text>
```

If this is understood then editing such a SVG file will be simple. Here is what the rules are.

- `<text ...`

  This indicates the start of a new character string which includes several self–evident definitions such as these.

  – `font-family`
  – `font-size`
  – `font-weight`
  – `fill`

- `This is Arial/Helvetica size = 1.2`

  This is the actual string itself that is going to be displayed

- `</text>`

  This indicates the end of the instructions to display the string

The command to rotate 90 degrees, again with tokens separated for clarity, is as follows.

```
<text x="232.00" y="705.00"
font-family="ARIAL"
font-size="25px"
font-weight="700"
transform="rotate(270.00,232.00,711.00)"
font-style = "oblique"
fill="rgb(0,140,0)">
Arial BoldOblique rotated 90 degrees</text>
```

The following files are distributed with the SIMFIT package in order to understand the previous details.

1. `lines.metafile, new_lines.metafile, fonts.metafile`

   These can be used to generate the figures using the SIMFIT program **simplot** or the SIMDEM program **simdem70**

2. `lines.eps, new-lines.eps, fonts.eps`

   PostScript graphics files

3. `lines.svg, new-lines.svg, fonts.svg`

   SVG graphics files

Note that, to edit text strings containing non–ASCII characters as in the lower strings using Symbol font, an editor supporting UTF8 must be used.

22.7 SVG: Creating collages

Sometimes it is required to combine several SVG files together to create a collage, i.e., a single SVG file containing subgraphs arranged into fixed or arbitrary positions. To do this, subsidiary SVG files are input into program **EditSVG**, then rearranged and scaled as necessary. When satisfied, program **EditSVG** can be used to output the composite graph as a SVG file (*.SVG).

Note that, after constructing a collage, program **EditSVG** also provides the facility to output re–build files (*.ISVG). These simply contain a list of SVG files used to compose the collage along with positions, scaling, and flags to indicate if white space borders are to be clipped from graphs, etc.

Such collages can be classified into several types as follows.

- A fixed or strict collage

  Here all the subgraphs have the same size and shape. For instance, all square, or all in landscape format, or all in portrait aspect ratio.

- An arbitrary or free–style collage

  In this case the subgraphs can be of arbitrary size and shape.

- Using ribbon graphs

  On occasions graphs created in square, portrait, or landscape, format are too compressed and it is necessary to apply differential stretching into a non standard format. This involves scaling the length of lines without altering the aspect ratio of the fonts used in titles, legends or plot labels. For instance with dendrograms or forest plots.

Several collages from the SIMFIT tutorials section concerning SVG are now shown to illustrate some typical possibilities.

- Collage 1.

  Freestyle non–overlapping type showing a collection of arbitrary mathematical equations and chemical formulas.

- Collage 2.

  Freestyle inlay type illustrating how to combine LaTeX maths with visual display of data.

- Collage 3.

  Freestyle inlay type illustrating how to combine a LaTeX chemical scheme and graph with data and best-fit curves.

- Collage 4.

  Strict type displaying illustrations from the SIMFIT tutorials SVG section.

- Collage 5.

  Differential stretching type illustrating how to stretch the x or y axes while maintaining constant aspect ratios for characters required for ribbon graphs.

22.7.1 Collage 1: Miscellaneous LaTeX examples

$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix}$$

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad \begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix} \qquad \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}}$$

$$\iint\limits_V \mu(v,w)\,du\,dv$$

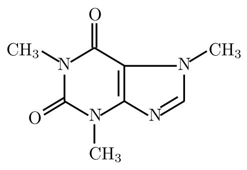
$$\iiint\limits_V \mu(u,v,w)\,du\,dv\,dw$$

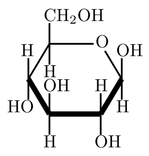
$$\int\cdots\int\limits_V \mu(z_1,\dots,z_k)\,\mathbf{dz}$$

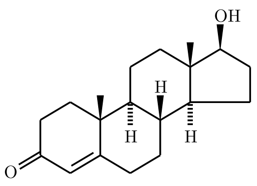
$$\lim_{x\to 0}\frac{\sin^2(x)}{x^2}=1$$

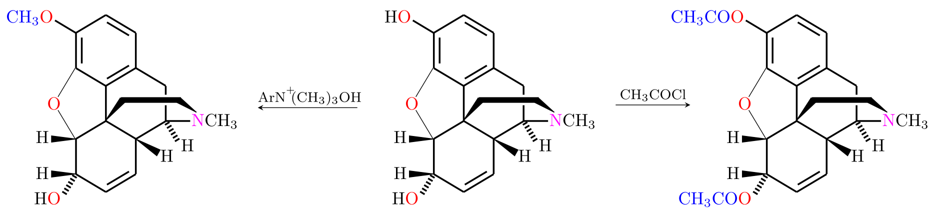
$$\varliminf_{n\to\infty}|a_{n+1}|/|a_n|=0$$

$$\varinjlim(m_i^\lambda\cdot M)^*\le\varprojlim_{A/p\to\lambda(A)}A_p\le 0$$


Caffeine


Glucose


Testosterone



Codeine Morphine Heroin

22.7.2 Collage 2: LaTeX maths

$$S_n = \sum_{k=1}^{n} X_i, \quad k = 0, 1, 2, \ldots, n$$

$$\text{Binomial Distribution: } P(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$E(S_n) = np$$

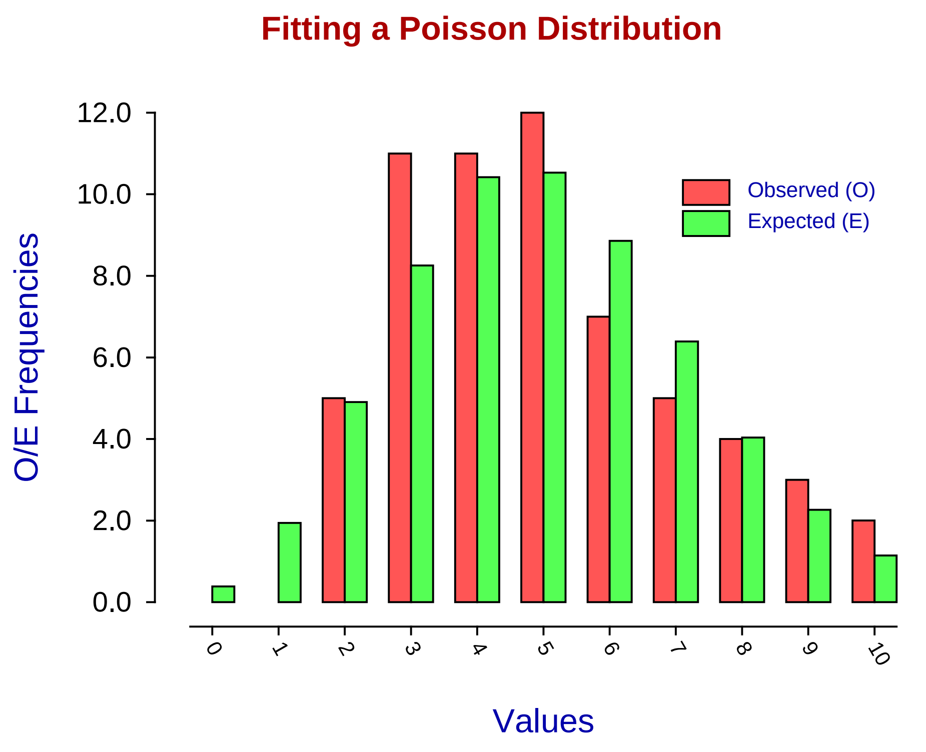
$$V(S_n) = np(1-p)$$

$$\text{Poisson Distribution: } P(Y = k) = \frac{\lambda^k}{k!} \exp(-\lambda), \quad y = 0, 1, 2, \ldots,$$

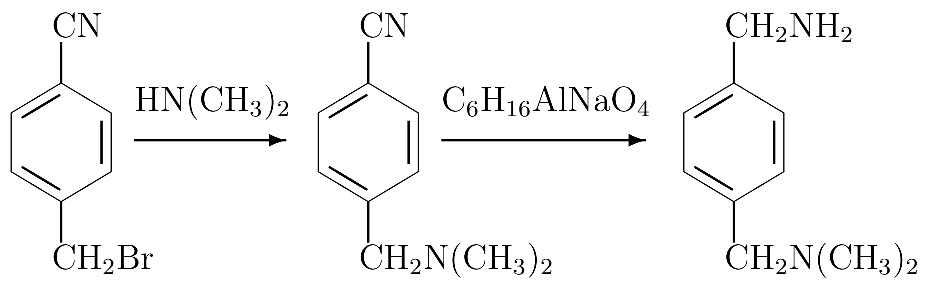
$$E(Y) = \lambda$$

$$V(Y) = \lambda$$

$$\lim_{n\to\infty, p\to 0} \binom{n}{k} p^k (1-p)^{n-k} = \frac{(np)^k}{k!} \exp(-np), \quad np > 0$$



22.7.3 Collage 3: LaTeX chemistry

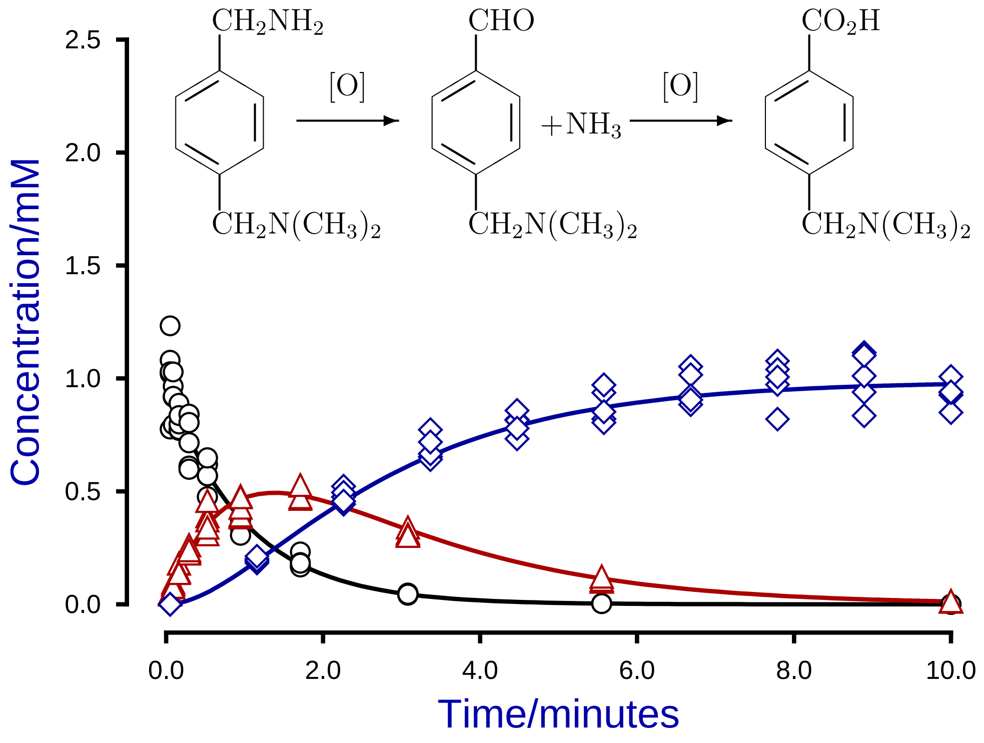


Chemical synthesis of p-dimethylaminomethlbenzylamine

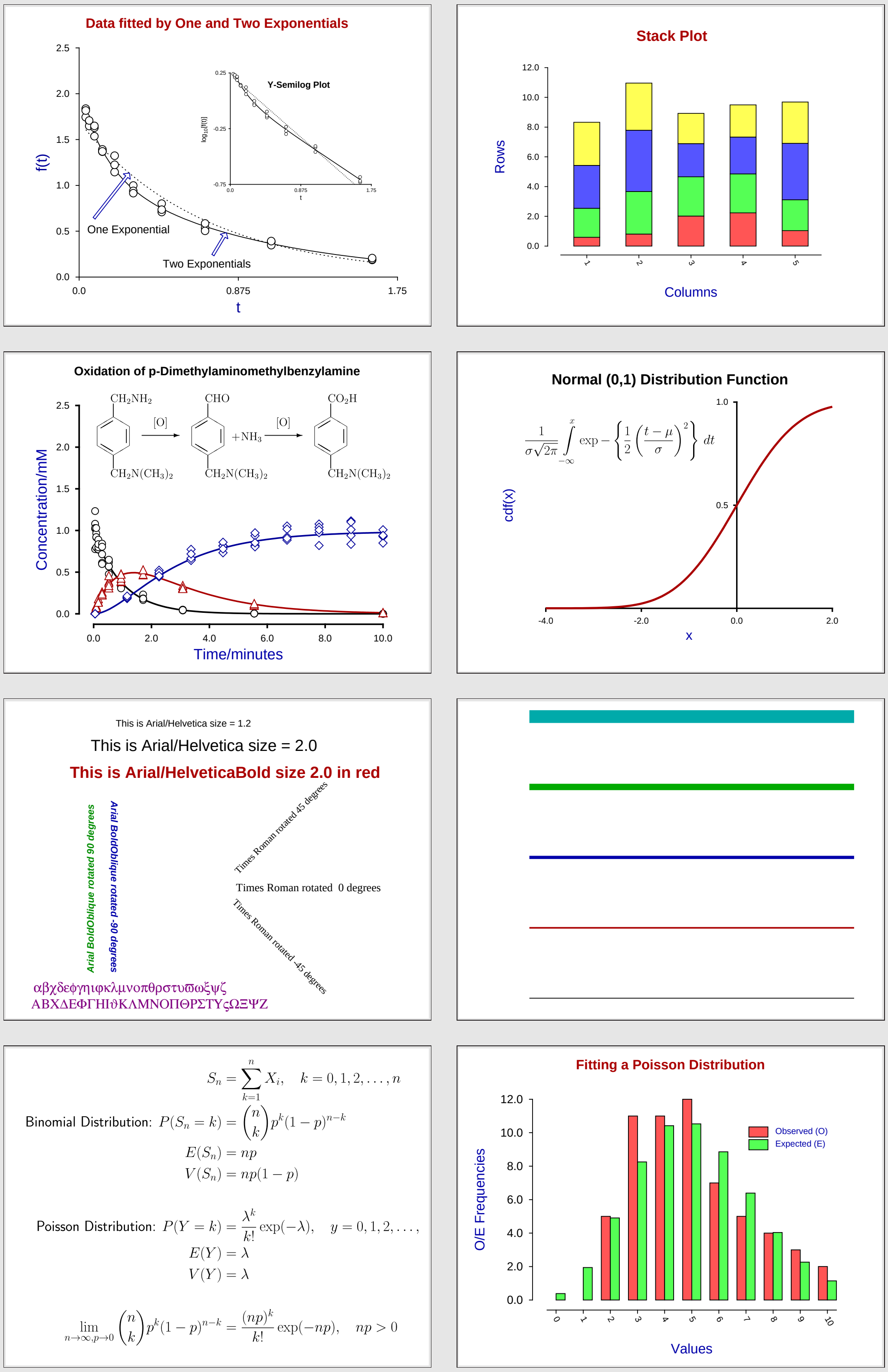
using

p-cyanobenzyl bromide, dimethylamine and Red-Al

Oxidation of p-Dimethylaminomethylbenzylamine



22.7.4 Collage 4: Tutorial examples



$$\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}\exp-\left\{\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right\}dt$$

$$S_n = \sum_{k=1}^{n} X_i, \quad k = 0, 1, 2, \dots, n$$

Binomial Distribution: $P(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k}$

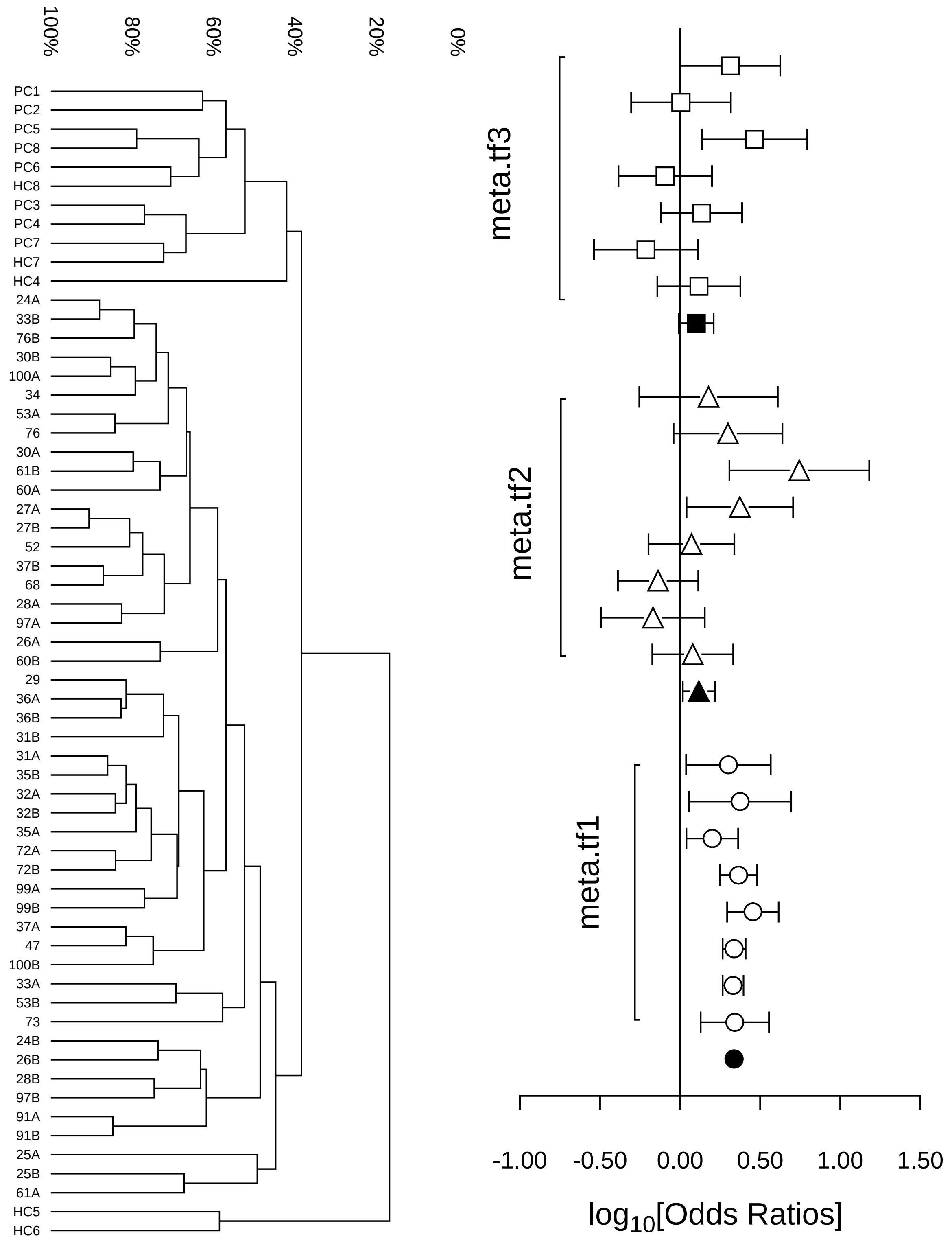
$$E(S_n) = np$$
$$V(S_n) = np(1-p)$$

Poisson Distribution: $P(Y = k) = \frac{\lambda^k}{k!}\exp(-\lambda), \quad y = 0, 1, 2, \dots,$

$$E(Y) = \lambda$$
$$V(Y) = \lambda$$

$$\lim_{n\to\infty, p\to 0} \binom{n}{k} p^k (1-p)^{n-k} = \frac{(np)^k}{k!}\exp(-np), \quad np > 0$$

22.7.5 Collage 5: Differential scaling to create ribbon graphs



22.8 SVG: Differential scaling examples

Usually scientific graphs are in landscape, square, or portrait aspect ratios as they are intended for inclusion in documents. However there many occasions where the data to be displayed are very extensive which would lead to overcrowding of curves or overlapping of labels and plotting symbols. Examples would be the display of dendrograms, forest plots, time series, or spectra where it would be useful to display data with extreme aspect ratios and attached scroll bars to scan the graph vertically or horizontally, as with chart–paper. Fortunately internet graphics using scalable vector graphics (SVG) allows this as all browsers support the SVG format, and SIMFIT provides facilities to stretch out overcrowded SVG graphs.

Unfortunately, where a graph is in bitmap, compressed bitmap, or vector format with landscape, square, or portrait aspect ratio, it is not possible to merely stretch the graph as this would lead to pixellation, and distortion of characters and plotting symbols, e.g., circles becoming ellipses, squares becoming rectangles, etc., so a special type of stretching procedure is required.

SIMFIT allows users to sculpture plots in advanced 2D format by editing symbols, line–types, labels, colors, titles and legends, but then to save in SVG format followed by interactively applying differential scaling until a satisfactory aspect ratio has been achieved before saving to a new SVG file which will automatically be displayed with scroll bars in browsers.

So differential scaling as defined in this way requires several steps in SIMFIT as follows.

1. Sculpture the graph in advanced 2D graphics
2. Choose the [SVG] option to view the current plot as SVG
3. Select values for X_scale and Y_scale and view the outcome
4. When satisfied save as a new differentially scaled SVG file

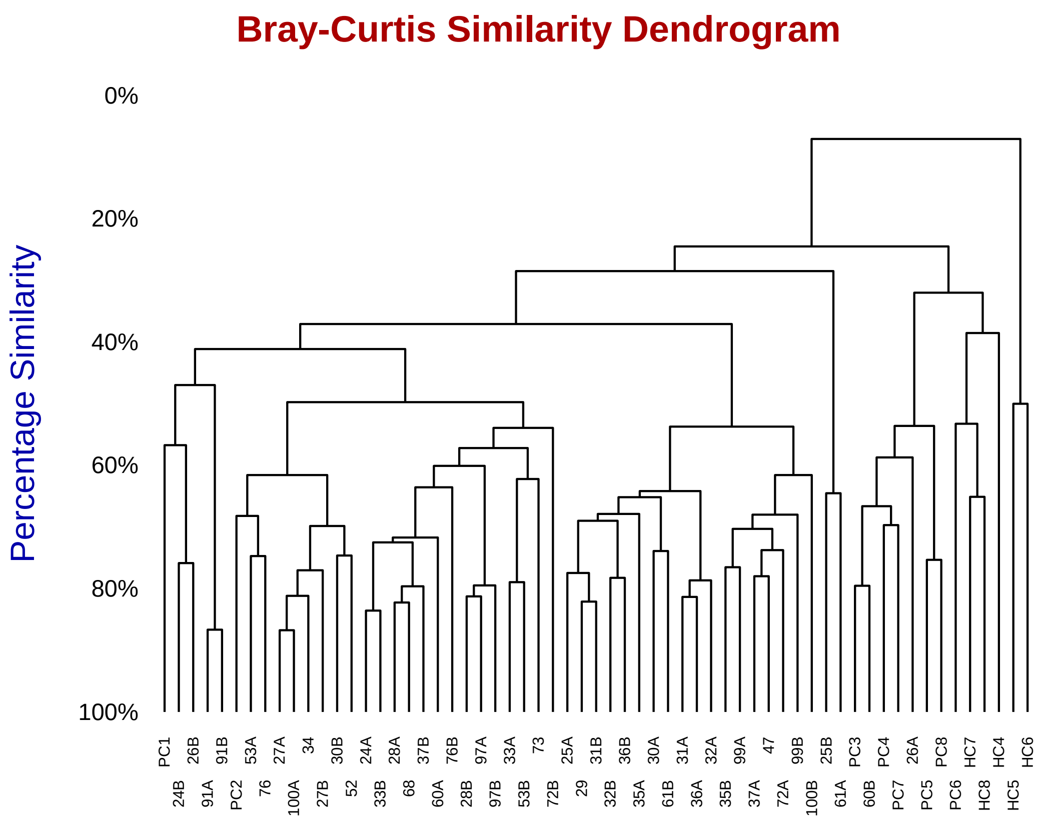
Note that the differentially scaled SVG files will have the following characteristics.

- The fonts and inter–character spacings will remain unchanged
- The line type and thickness will remain unchanged
- The plotting symbols will maintain their aspect ratios
- Only the white space between the lines, symbols and labels will change
- However there are likely to be unexpected effects if extreme scaling is used. For instance.
  - With horizontally stretched graphs a centralized title or X–legend may only become visible after horizontal scrolling.
  - With vertically stretched graphs a centralized Y–legend may only become visible after vertical scrolling.
  - In extreme cases like this it is best to delete the offending title or legend and add a new text string in an appropriate position, i.e., close to the top left of the plot.
  - Moving text, arrows, or information panels by eye with the red arrow will never be precise and this will become increasingly obvious at as the scaling increases. Positioning can be improved by using the graticule function, i.e., the mesh of intersecting lines created from the [Style] button.

Some of the unexpected effects and avoidance of such issues will be clear by detailing several examples that will be discussed next and, to understand this material, it should be combined with viewing the page `https://simfit.org.uk/svg.html`.

22.8.1 A normal dendrogram

This is a very simple example where differential scaling is scarcely necessary but, as it does illustrate several points, creation of the next plot requires some explanation.



The steps required to generate this default plot are as follows.

1. The SimFIT test data file `cluster.tf2` contains a multivariate data set that can be used to generate a dendrogram either by input into the SimFITprogram **simstat** then choosing multivariate statistics to create a distance matrix with a chosen metric followed by plotting a dendrogram with labels, or by input into **simplot** then opening the option to create statistical graphs.

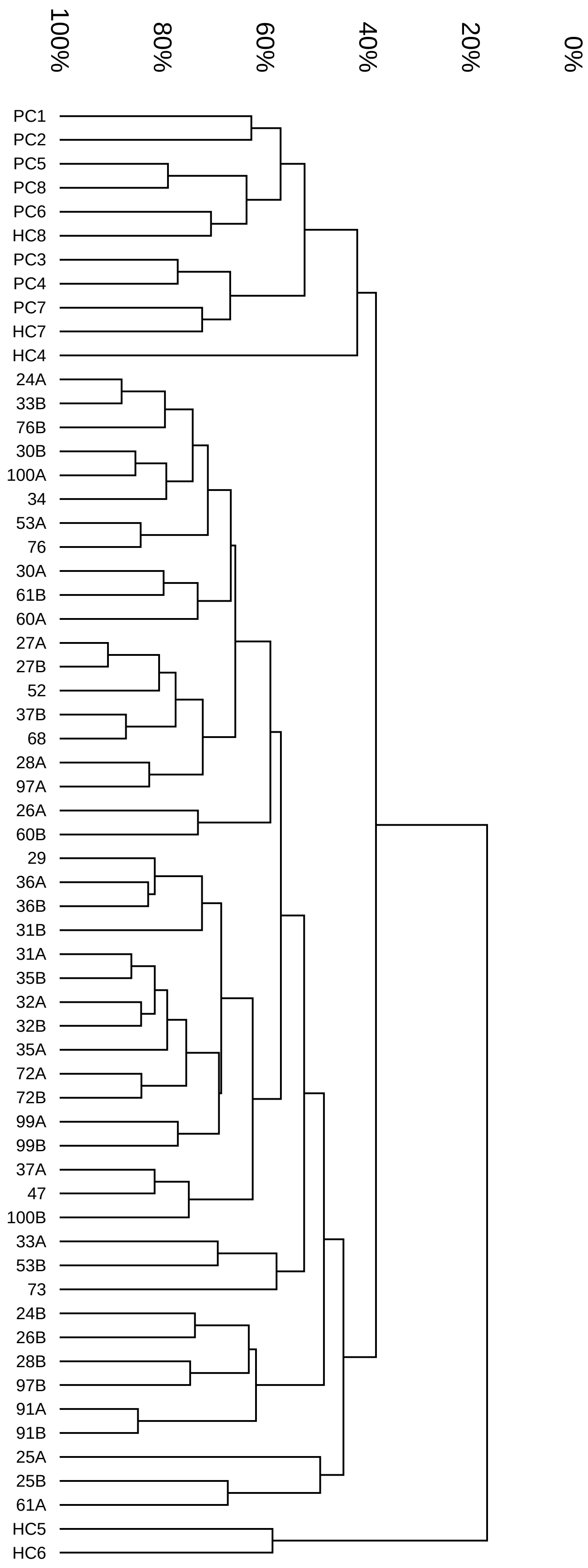
2. The type of dendrogram required can be selected as the Bray-Curtis similarity type, which is often used for such biological data. This would, of course, always be a user–selected decision that would depend on the data and either statistical arguments or simply the visual appearance preferred.

3. The title and legend would normally be edited at this stage as required.

4. It should be noted that, with this example, SimFIT has automatically chosen to create a double label system with rotated X axis labels in order to create a legible X–axis labeling system without the labels overlapping.

The next plot illustrates the effect of differential X–axis scaling, where the title and Y–legend have been suppressed in order to display the graph in as large a size as is possible in a document.

3.4



To generate this particular figure these steps were taken.

The dendrogram was created in SimFIT advanced 2D graphics

The title was suppressed.

The *Y*–legend was suppressed.

The double label threshold was increased using the `[Labels]`option followed by choosing to edit the *X*-axis labels
This allows the labels to be displayed on one line.

The font size for the *Y*–axis numbers and the *X*–axis labels was increased.

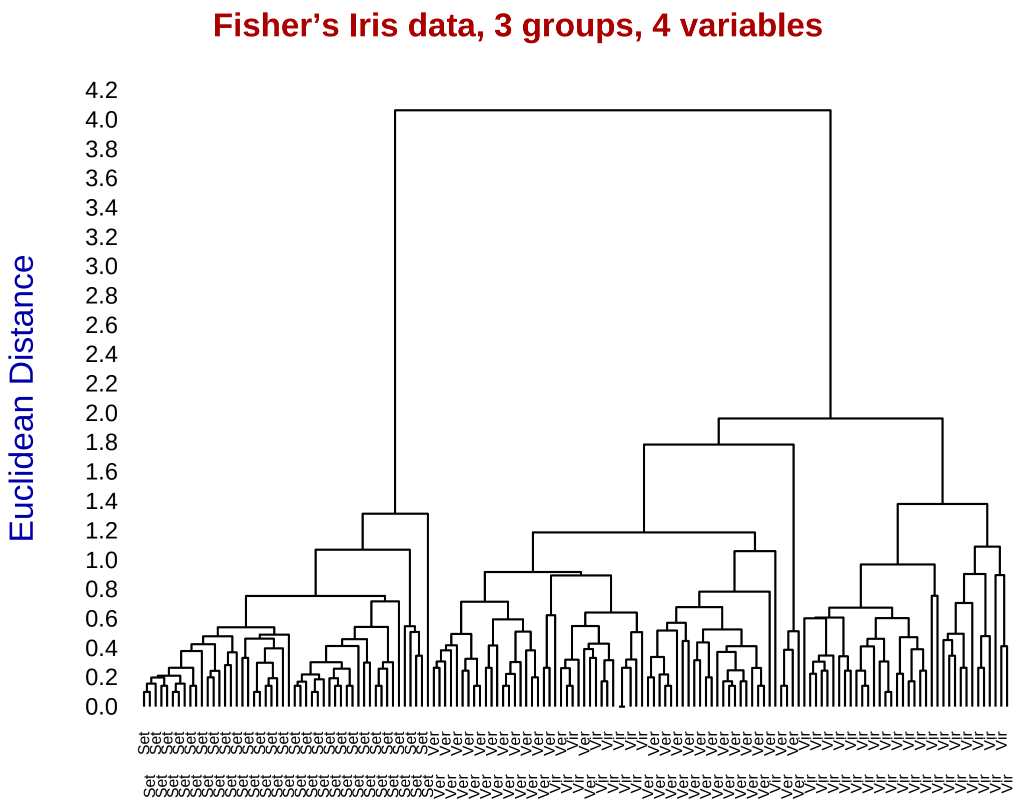
To improve legibility the *X*-axis scaling factor X_scale was then increased to 3. Of course the graph becomes too large to display in this document so the scrolling version with title and *Y*–legend has to be observed using the [SVG] option from the main page at
`https://simfit.org.uk`.

In order to view with the labels in standard orientation rather than rotated it would then be necessary to increase the *X*–scaling factor to a larger value than 3 so as to maintain legibility and prevent labels overlapping.

The re–scaled and edited graph was saved to a new file which is displayed to the left, after rotation.

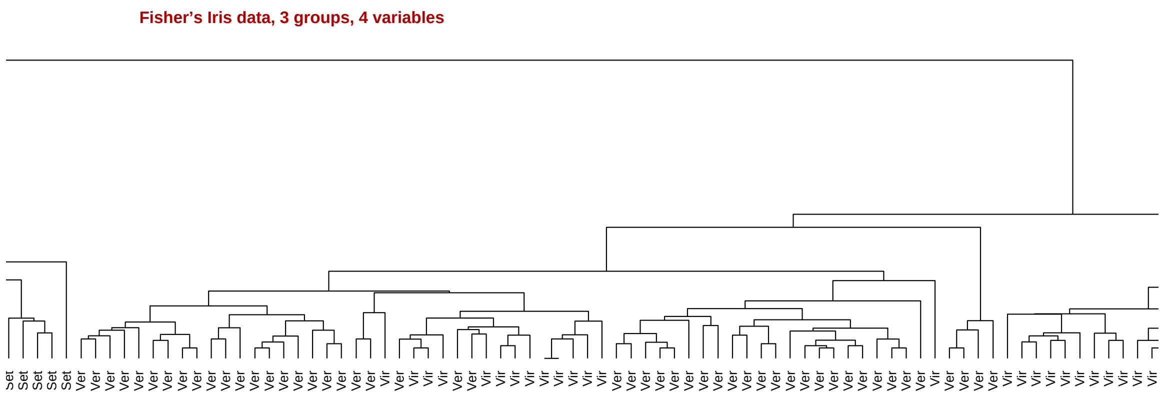
22.8.2 A crowded dendrogram

Proceeding as before but using the SIMFIT test file `iris.tf1` displays the famous Fisher iris multivariate data as the following dendrogram.



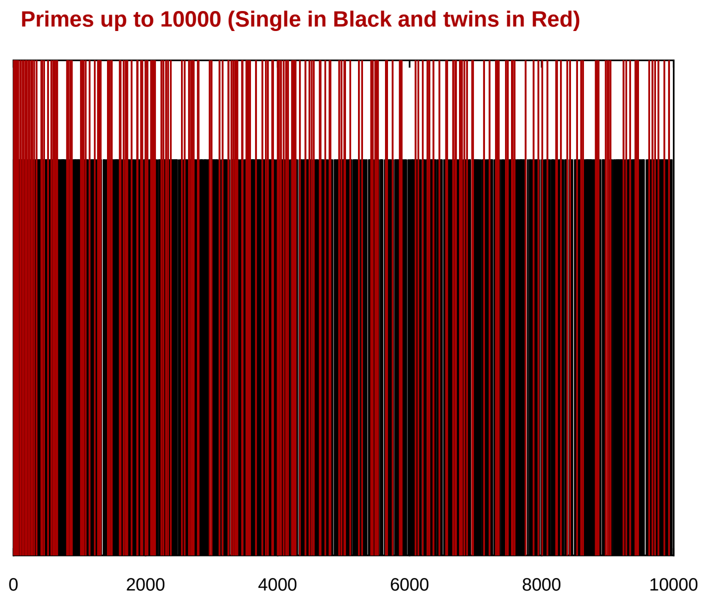
Here the labels are too crowded to read easily but placing all the labels on the same line instead of on double lines then stretching by using X_scale = 5 creates a much more legible dendrogram as will be seen from the SIMFIT website. Unfortunately this is then too wide to display in full in this document without serious reduction.

However, using the SIMFIT Postscript technique to clip a section out of such an expanded graph (which is described in the reference manuals) does allows the display of an arbitrary section clipped out of the center of the stretched graph as shown next (after some reduction).

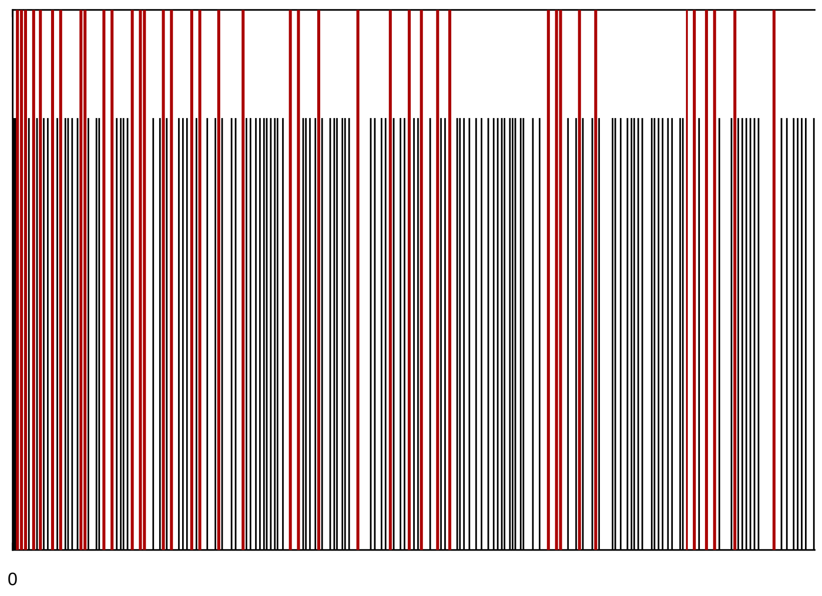


22.8.3 An extremely crowded plot

To create the chart–paper type of plot for primes displayed on the website the test files `primes.tf1` and `primes.tf2` were input into SIMFIT program **simplot** followed by editing the Y–axis to have range (0, 1.25), the data being plotted with no lines but with solid barchart–type symbols colored black for `primes.tf1` and red for `primes.tf2`, to create the next figure (after suppressing X and Y legends and Y labels and using integers as X–labels).



Using the SIMFIT Postscript facility described for the previous example to stretch by a factor of 10 and then clip out the start of this graph to an EPS file (shown below) indicates the effect of differential scaling to separate out the vertical bars from the solid mass of color in the plot (shown above) without differential scaling. This is seen more clearly in the scrolling example stretched by a factor of 20 on the website.



Appendix A

Distributions and special functions

Techniques for calling these functions from within user defined models are discussed starting on page 432.

A.1 Discrete distribution functions

A discrete random variable X can have one of n possible values $x_1, x_2, \ldots, x_n$ and has a mass function $f_X \geq 0$ and cumulative distribution function $0 \leq F_X \leq 1$ that define probability, expectation, and variance by

$$\begin{aligned}
P(X = x_j) &= f_X(x_j), \text{ for } j = 1, 2, \ldots, n \\
&= 0 \text{ otherwise} \\
P(X \leq x_j) &= \sum_{i=1}^{j} f_X(x_i) \\
1 &= \sum_{i=1}^{n} f_X(x_i) \\
E(g(X)) &= \sum_{i=1}^{n} g(x_i) f_X(x_i) \\
E(X) &= \sum_{i=1}^{n} x_i f(x_i) \\
V(X) &= \sum_{i=1}^{n} (x_i - E(X))^2 f_X(x_i) \\
&= E(X^2) - E(X)^2.
\end{aligned}$$

A.1.1 Bernoulli distribution

A Bernoulli trial has only two possible outcomes, $X = 1$ or $X = 0$ with probabilities p and $q = 1 - p$.

$$\begin{aligned}
P(X = k) &= p^k q^{1-k} \text{ for } k = 0 \text{ or } k = 1 \\
E(X) &= p \\
V(X) &= pq
\end{aligned}$$

A.1.2 Binomial distribution

This models the case of n independent Bernoulli trials with probability of success (i.e. $X_i = 1$) equal to p and failure (i.e. $X_i = 0$) equal to $q = 1 - p$. The random binomial variable S_n is defined as the sum of the n values of X_i without regard to order, i.e. the number of successes in n trials.

$$\begin{aligned} S_n &= \sum_{i=1}^{n} X_i \\ P(S_n = k) &= \binom{n}{k} p^k (1-p)^{n-k}, \text{ for } k = 0, 1, 2, \ldots, n \\ E(S_n) &= np \\ V(S_n) &= np(1-p) \end{aligned}$$

The run test, sign test, analysis of proportions, and many methods for analyzing experiments with only two possible outcomes are based on the binomial distribution.

A.1.3 Multinomial distribution

Extending the binomial distribution to k possible outcomes of frequency f_i in a sample of size n is described by

$$\begin{aligned} P(X_1 = f_1, X_2 = f_2, \ldots, X_k = f_k) &= \frac{n!}{f_1! f_2! \cdots f_k!} p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k} \\ \text{where } f_1 + f_2 + \cdots + f_k &= n \\ \text{and } p_1 + p_2 + \cdots + p_k &= 1. \end{aligned}$$

An example would be the trinomial distribution, which is used to analyse the outcome of incubating a clutch of eggs; they can hatch male, or female, or fail to hatch.

A.1.4 Geometric distribution

This is the distribution of the number of failures prior to the first success, where

$$\begin{aligned} P(X = k) &= pq^k \\ E(X) &= q/p \\ V(X) &= q/p^2. \end{aligned}$$

A.1.5 Negative binomial distribution

The probability of k failures prior to the rth success is the random variable S_r, where

$$\begin{aligned} P(S_r = k) &= \binom{r+k-1}{k} p^r q^k \\ E(S_r) &= rq/p \\ V(S_r) &= rq/p^2. \end{aligned}$$

A.1.6 Hypergeometric distribution

This models sampling without replacement, where n objects are selected from N objects, consisting of $M \le N$ of one kind and $N - M$ of another kind, and defines the random variable S_n as

$$S_n = X_1 + X_2 + \ldots X_n$$

where $X_i = 1$ for success with $P(X_i = 1) = M/N$, and $X_i = 0$ for failure.

$$\begin{aligned} P(S_n = k) &= \binom{M}{k}\binom{N-M}{n-k} \Big/ \binom{N}{n}, \quad \text{where } \binom{a}{b} = 0 \text{ when } b > a > 0 \\ E(S_n) &= nM/N \\ V(S_n) &= npq(N-n)/(N-1) \end{aligned}$$

Note that when $N \gg n$ this reduces to the binomial distribution with $p = M/N$.

A.1.7 Poisson distribution

This is the limiting form of the binomial distribution for large n and small p but finite $np = \lambda > 0$.

$$\begin{aligned} P(X = k) &= \frac{\lambda^k}{k!}\exp(-\lambda), \text{ for } x = 0, 1, 2, \ldots, \\ E(X) &= \lambda \\ V(X) &= \lambda \end{aligned}$$

The limiting result, for fixed $np > 0$, that

$$\lim_{n\to\infty} \binom{n}{k} p^k (1-p)^{n-k} = \frac{(np)^k}{k!}\exp(-np)$$

can be used to support the hypothesis that counting is a Poisson process, as in the distribution of bacteria in an sample, so that the error is of the order of the mean. The Poisson distribution also arises from Poisson processes, like radioactive decay, where the probability of k events which occur at a rate λ per unit time is

$$P(k \text{ events in } (0,t)) = \frac{(\lambda t)^k}{k!}\exp(-\lambda t).$$

The Poisson distribution has the additive property that, given n independent Poisson variables X_i with parameters λ_i, the sum $Y = \sum_{i=1}^n X_i$ has a Poisson distribution with parameter $\lambda_y = \sum_{i=1}^n \lambda_i$.

A.2 Continuous distributions

A continuous random variable X is defined over some range by a probability density function $f_X \geq 0$ and cumulative distribution function $0 \leq F_X \leq 1$ that define probability, expectation, and variance by

$$\begin{aligned} F_X(x) &= \int_{-\infty}^{x} f_X(t)\,dt \\ P(A \leq x \leq B) &= F_x(B) - F_X(A) \\ &= \int_A^B f_X(t)\,dt \\ 1 &= \int_{-\infty}^{\infty} f_X(t)\,dt \\ E(g(X)) &= \int_{-\infty}^{\infty} g(t) f_X(t)\,dt \\ E(X) &= \int_{-\infty}^{\infty} t f_X(t)\,dt \\ V(X) &= \int_{-\infty}^{\infty} (t - E(X))^2 f_X(t)\,dt. \end{aligned}$$

In the context of survival analysis, the random survival time $X \geq 0$, with density $f(x)$, cumulative distribution function $F(x)$, survivor function $S(x)$, hazard function $h(x)$, and integrated hazard function $H(x)$ are defined by

$$\begin{aligned} S(x) &= 1 - F(x) \\ h(x) &= f(x)/S(x) \\ H(x) &= \int_0^x h(u)\,du \\ f(x) &= h(x)\exp\{-H(x)\}. \end{aligned}$$

A.2.1 Uniform distribution

This assumes that every value is equally likely for $A \le X \le B$, so that

$$\begin{aligned} f_X(x) &= 1/(B-A) \\ E(X) &= (A+B)/2 \\ V(X) &= (A+B)^2/12. \end{aligned}$$

A.2.2 Normal (or Gaussian) distribution

This has mean μ and variance σ^2 and, for convenience, X is often standardized to Z, so that if $X \sim N(\mu, \sigma^2)$, then $Z = (x-\mu)/\sigma \sim N(0,1)$.

$$\begin{aligned} f_X(x) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \\ E(X) &= \mu \\ V(X) &= \sigma^2 \\ \Phi(z) &= F_X(z) \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(-t^2/2)\, dt. \end{aligned}$$

It is widely used in statistical modelling, e.g., the assumption of normally distributed dosage tolerance leads to a probit regression model for the relationship between the probability of death and dose. There are several important results concerning the normal distribution which are heavily used in hypothesis testing.

A.2.2.1 Example 1. Sums of normal variables

Given n independent random variables $X_i \sim N(\mu_i, \sigma_i^2)$, then the linear combination $Y = \sum_{i=1}^n a_i X_i$ is normally distributed with parameters $\mu_y = \sum_{i=1}^n a_i \mu_i$ and $\sigma_y^2 = \sum_{i=1}^n a_i^2 \sigma_i^2$.

A.2.2.2 Example 2. Convergence of a binomial to a normal distribution

If S_n is the sum of n Bernoulli variables that can be 1 with probability p, and 0 with probability $1-p$, then S_n is binomially distributed and, by the central limit theorem, it is asymptotically normal in the sense that

$$\lim_{n\to\infty} P\left(\frac{S_n - np}{\sqrt{np(1-p)}} \le z\right) = \Phi(z).$$

The argument that experimental error is the sum of many errors that are equally likely to be positive or negative can be used, along with the above result, to support the view that experimental error is often approximately normally distributed.

A.2.2.3 Example 3. Distribution of a normal sample mean and variance

If $X \sim N(\mu, \sigma^2)$ and from a sample of size n the sample mean

$$\bar{x} = \sum_{i=1}^{n} x_i/n$$

and the sample variance

$$S^2 = \sum_{i=1}^{n} (x_i - \bar{x})^2/n$$

are calculated, then

(a) $\bar{X} \sim N(\mu, \sigma^2/n)$;
(b) $nS^2/\sigma^2 \sim \chi^2(n-1)$, $E(S^2) = (n-1)\sigma^2/n$, $V(S^2) = 2(n-1)\sigma^4/n^2$; and
(c) $\bar{X}$ and S^2 are stochastically independent.

A.2.2.4 Example 4. The central limit theorem

If independent random variables X_i have mean μ and variance σ^2 from some distribution, then the sum $S_n = \sum_{i=1}^{n} X_i$, suitably normalized, is asymptotically normal, that is

$$\lim_{n\to\infty} P\left(\frac{S_n - n\mu}{\sigma\sqrt{n}} \le z\right) = \Phi(z), \text{ or}$$
$$P(X_1 + X_2 + \cdots + X_n \le y) \approx \Phi\left(\frac{y - n\mu}{\sigma\sqrt{n}}\right).$$

Under appropriate restrictions, even the need for identical distributions can be relaxed.

A.2.3 Lognormal distribution

This is frequently used to model unimodal distributions that are skewed to the right, e.g., plasma concentrations which cannot be negative, that is, where the logarithm is presumed to be normally distributed so that, for $X = \exp(Y)$ where $Y \sim N(\mu, \sigma^2)$, then

$$\begin{aligned} f_X(x) &= \frac{1}{\sigma x\sqrt{2\pi}} \exp\left(-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right) \\ E(X) &= \exp(\mu + \sigma^2/2) \\ V(X) &= (\exp(\sigma^2) - 1)\exp(2\mu + \sigma^2). \end{aligned}$$

A.2.4 Bivariate normal distribution

If variables X and Y are jointly distributed according to a bivariate normal distribution the density function is

$$\begin{aligned} f_{X,Y} &= \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2}Q\right) \\ \text{where } Q &= \frac{1}{1-\rho^2}\left(\frac{(x-\mu_X)^2}{\sigma_X^2} - 2\rho\frac{(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y} + \frac{(y-\mu_Y)^2}{\sigma_Y^2}\right) \end{aligned}$$

with $\sigma_X^2 > 0, \sigma_Y^2 > 0$, and $-1 < \rho < 1$. Here the marginal density for X is normal with mean μ_X and variance σ_X^2, the marginal density for Y is normal with mean μ_Y and variance σ_Y^2, and when X and Y are independent the correlation ρ is zero. At fixed probability levels, the quadratic form Q defines an ellipse in the X, Y plane which will have axes parallel to the X, Y axes if $\rho = 0$, but with rotated axes otherwise.

A.2.5 Multivariate normal distribution

If a m dimensional random vector X has a $N(\mu, \Sigma)$ distribution , the density is

$$f_X(x) = (2\pi)^{-m/2}|\Sigma|^{-1/2}\exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}.$$

Contours of equi-probability are defined by $f(x) = k$ for some $k > 0$ as a hyper-ellipsoid in m dimensional space, and the density has the properties that any subsets of X or linear transformations of X are also multivariate normal. Many techniques, e.g., MANOVA, assume this distribution.

A.2.6 *t* distribution

The t distribution arises naturally as the distribution of the ratio of a normalized normal variate Z divided by the square root of a chi-square variable χ^2 divided by its degrees of freedom ν.

$$
\begin{aligned}
t(\nu) &= \frac{Z}{\sqrt{\chi^2(\nu)/\nu}}, \text{ or setting } X = t(\nu)\\
f_X(x) &= \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}}\left(1+\frac{x^2}{\nu}\right)^{-(\nu+1)/2}\\
E(X) &= 0\\
V(X) &= \nu/(\nu-2) \text{ for } \nu > 2.
\end{aligned}
$$

The use of the t test for testing for equality of means with two normal samples X_1, and X_2 (page 136), with sizes n_1 and n_2 and the same variance, uses the fact that the sample means are normally distributed, while the sample variances are chi-square distributed, so that under H_0,

$$
\begin{aligned}
Z &= \frac{\bar{x}_1 - \bar{x}_2}{\sigma\sqrt{1/n_1 + 1/n_2}}\\
U &= \frac{n_1 s_1^2 + n_2 s_2^2}{\sigma^2(n_1 + n_2 - 2)}\\
T &= Z/\sqrt{U}\\
&\sim t(n_1 + n_2 - 2)\\
T^2 &\sim F(1, n_1 + n_2 - 2).
\end{aligned}
$$

For the case of unequal variances the Welch approximation is used, where the above test statistic T and degrees of freedom ν calculated using a pooled variance estimate, are replaced by

$$
\begin{aligned}
T &= \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}\\
\nu &= \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{(s_1^2/n_1)^2/(n_1-1) + (s_2^2/n_2)^2/(n_2-1)}.
\end{aligned}
$$

The paired t test (page 138) uses the differences $d_i = x_i - y_i$ between correlated variables X and Y and only assumes that the differences are normally distributed, so that the test statistic for the null hypothesis is

$$
\begin{aligned}
\bar{d} &= \sum_{i=1}^{n} d_i/n\\
s_d^2 &= \sum_{i=1}^{n} (d_i - \bar{d})^2/(n-1)\\
T &= \bar{d}/\sqrt{s_{\bar{d}}^2/n}.
\end{aligned}
$$

A.2.7 Cauchy distribution

This is the distribution of the ratio of two normal variables. For, instance, if $X_1 \sim N(0,1)$ and $X_2 \sim N(0,1)$ then the ratio $X = X_1/X_2$ has a Cauchy distribution, where $E(X)$ and $V(X)$ are not defined, with

$$
f_X = \frac{1}{\pi(1+x^2)}.
$$

This is a better model for experimental error than the normal distribution as the tails are larger than with a normal distribution. However, because of the large tails, the mean and variance are not defined, as with the t distribution with $\nu = 1$ which reduces to a Cauchy distribution.

A.2.8 Chi-square distribution

The χ^2 distribution with ν degrees of freedom results from adding together the squares of ν independent Z variables.

$$\begin{aligned}
\chi^2(\nu) &= \sum_{i=1}^{\nu} z_i^2 \text{ or, setting } X = \chi^2(\nu),\\
f_X(x) &= \frac{1}{2^{\nu/2}\Gamma(\nu/2)} x^{\nu/2-1}\exp(-x/2)\\
E(X) &= \nu\\
V(X) &= 2\nu.
\end{aligned}$$

It is the distribution of the sample variance from a normal distribution, and is widely used in goodness of fit testing since, if n frequencies E_i are expected and n frequencies O_i are observed, then

$$\lim_{n\to\infty}\sum_{i=1}^{n}\frac{(O_i - E_i)^2}{E_i} = \chi^2(\nu).$$

Here the degrees of freedom ν is just $n-1$ minus the number of extra parameters estimated from the data to define the expected frequencies. Cochran's theorem is another result of considerable importance in several areas of data analysis, e.g., the analysis of variance, and this considers the situation where $Z_1, Z_2, \ldots, Z_n$ are independent standard normal variables that can be written in the form

$$\sum_{i=1}^{n} Z_i^2 = Q_1 + Q_2 + \cdots + Q_k$$

where each Q_i is a sum of squares of linear combinations of the Z_i. If the rank of each Q_i is r_i and

$$n = r_1 + r_2 + \cdots + r_k$$

then the Q_i have independent chi-square distributions, each with r_i degrees of freedom.

A.2.9 *F* distribution

The F distribution arises when a chi-square variable with ν_1 degrees of freedom (divided by ν_1) is divided by another independent chi-square variable with ν_2 degrees of freedom (divided by ν_2).

$$\begin{aligned}
F(\nu_1,\nu_2) &= \frac{\chi^2(\nu_1)/\nu_1}{\chi^2(\nu_2)/\nu_2} \text{ or, setting } X = F(\nu_1,\nu_2),\\
f_X(x) &= \frac{\nu_1^{\nu_1/2}\nu_2^{\nu_2/2}\Gamma((\nu_1+\nu_2)/2)}{\Gamma(\nu_1/2)\Gamma(\nu_2/2)} x^{(\nu_1-2)/2}(\nu_1 x + \nu_2)^{-(\nu_1+\nu_2)/2}\\
E(X) &= \nu_2/(\nu_2 - 2) \text{ for } \nu_2 > 2.
\end{aligned}$$

The F distribution is used in the variance ratio tests and analysis of variance, where sums of squares are partitioned into independent chi-square variables whose normalized ratios, as described above, are tested for equality, etc. as variance ratios.

A.2.10 Exponential distribution

This is the distribution of time to the next failure in a Poisson process. It is also known as the Laplace or negative exponential distribution, and is defined for $X \ge 0$ and $\lambda > 0$.

$$\begin{aligned}
f_X(x) &= \lambda\exp(-\lambda x)\\
E(X) &= 1/\lambda\\
V(X) &= 1/\lambda^2.
\end{aligned}$$

Note that, if $0 \le X \le 1$ has a uniform distribution, then $Y = (1/\lambda)\log(X)$ has an exponential distribution. Also, when used as a model in survival analysis, this distribution does not allow for wear and tear, as the hazard function is just the constant λ, as follows:

$$\begin{aligned} S(x) &= \exp(-\lambda x) \\ h(x) &= \lambda \\ H(x) &= \lambda x. \end{aligned}$$

A.2.11 Beta distribution

This is useful for modelling densities that are constrained to the unit interval $0 \le x \le 1$, as a great many shapes can be generated by varying the parameters $r > 0$ and $s > 0$.

$$\begin{aligned} f_X(x) &= \frac{\Gamma(r+s)}{\Gamma(r)\Gamma(s)} x^{r-1}(1-x)^{s-1} \\ E(X) &= r/(r+s) \\ V(X) &= rs/((r+s+1)(r+s)^2) \end{aligned}$$

It is also used to model situations where a probability, e.g., the binomial parameter, is treated as a random variable, and it is also used in order statistics, e.g., the distribution of the kth largest of n uniform $(0, 1)$ random numbers is beta distributed with $r = k$ and $s = n - k + 1$.

A.2.12 Gamma distribution

This distribution with $x > 0$, $r > 0$, and $\lambda > 0$ arises in modelling waiting times from Poisson processes.

$$\begin{aligned} f_X(x) &= \frac{\lambda^r x^{r-1}}{\Gamma(r)} \exp(-\lambda x) \\ E(X) &= r/\lambda \\ V(X) &= r/\lambda^2 \end{aligned}$$

When r is a positive integer it is also known as the Erlang density, i.e. the time to the rth occurrence in a Poisson process with parameter λ.

A.2.13 Weibull distribution

This is used to model survival times where the survivor function $S(x)$ and hazard rate or failure rate function $h(x)$ are defined as follows.

$$\begin{aligned} f_X(x) &= AB(Ax)^{B-1}\exp(-Ax)^B \\ F_X(x) &= 1 - \exp(-Ax)^B \\ S(x) &= 1 - F_X(x) \\ h(x) &= f_X(x)/S(x) \\ &= AB(Ax)^{B-1}. \end{aligned}$$

It reduces to the exponential distribution when $B = 1$, but it is a much better model for survival times, due to the flexibility in curve shapes when B is allowed to vary, and the simple forms for the survivor and hazard functions. Various alternative parameterizations are used, for instance

$$\begin{aligned} f_X(x) &= \left(\frac{\alpha}{\beta}\right)\left(\frac{x-\gamma}{\beta}\right)^{\alpha-1} \exp - \left(\frac{x-\gamma}{\beta}\right)^{\alpha} \\ E(X) &= \gamma + \beta\Gamma\left(\frac{1}{\alpha}+1\right) \\ V(X) &= \beta^2\left\{\Gamma\left(\frac{2}{\alpha}+1\right) - \Gamma^2\left(\frac{1}{\alpha}+1\right)\right\}. \end{aligned}$$

A.2.14 Logistic distribution

This resembles the normal distribution and is widely used in statistical modelling, e.g., the assumption of logistic dosage tolerances leads to a linear logistic regression model.

$$\begin{aligned} f_X(x) &= \frac{\exp[(x-\mu)/\tau]}{\tau\{1-\exp[(x-\mu)/\tau]\}^2} \\ F_X(x) &= \frac{\exp[(x-\mu)/\tau]}{1+\exp[(x-\mu)/\tau]} \\ E(X) &= \mu \\ V(x) &= \pi^2\tau^2/3 \end{aligned}$$

A.2.15 Log logistic distribution

By analogy with the lognormal distribution, if $\log X$ has the logistic distribution, and $\mu = -\log\rho$, $\kappa = 1/\tau$, then the density, survivor function, and hazard functions are simpler than for the log normal distribution, namely

$$\begin{aligned} f(x) &= \frac{\kappa x^{\kappa-1}\rho^\kappa}{\{1+(\rho x)^\kappa\}^2} \\ S(x) &= \frac{1}{1+(\rho x)^\kappa} \\ h(x) &= \frac{\kappa x^{\kappa-1}\rho^\kappa}{1+(\rho x)^\kappa}. \end{aligned}$$

A.3 Non-central distributions

These distributions are similar to the corresponding central distributions but they require an additional non-centrality parameter, λ. One of the main uses is in calculations of statistical power as a function of sample size. For instance, calculating the power for a chi-square test requires the cumulative probability function for the non-central chi-square distribution.

A.3.1 Non-central beta distribution

The lower tail probability for parameters a and b is

$$P(X \le x) = \sum_{i=0}^{\infty} \frac{(\lambda/2)\ \exp(-\lambda/2)}{i!} P(\beta_{a,b} \le x),$$

where $0 \le x \le 1$, $a > 0$, $b > 0$, $\lambda \ge 0$, and $P(\beta_{a,b} \le x)$ is the lower tail probability for the central beta distribution.

A.3.2 Non-central chi-square distribution

The lower tail probability for ν degrees of freedom is

$$P(X \le x) = \sum_{i=0}^{\infty} \frac{(\lambda/2)^i\ \exp(-\lambda/2)}{i!} P(\chi^2_{\nu+2i} \le x),$$

where $x \ge 0$, $\lambda \ge 0$, and $P(\chi^2_k \le x)$ is the lower tail probability for the central chi-square distribution with k degrees of freedom.

A.3.3 Non-central F distribution

The lower tail probability $P(X \le x)$ for ν_1 and ν_2 degrees of freedom is

$$\int_0^x \sum_{i=0}^{\infty} \frac{(\lambda/2)^i (\nu_1 + 2i)^{(\nu_1+2i)/2} \nu_2^{\nu_2/2} \exp(-\lambda/2)}{i!\, B((\nu_1+2i)/2, \nu_2/2)} u^{(\nu_1+2i-2)/2} [\nu_2 + (\nu_1 + 2i)u]^{-(\nu_1+2i+\nu_2)/2}\, du$$

where $x \ge 0$, $\lambda \ge 0$, and $B(a, b)$ is the beta function for parameters a and b.

A.3.4 Non-central t distribution

The lower tail probability for ν degrees of freedom is

$$P(X \le x) = \frac{1}{\Gamma(\nu/2) 2^{(\nu-2)/2}} \int_0^{\infty} \Phi\left(\frac{ux}{\sqrt{\nu}} - \lambda\right) u^{\nu-1} \exp(-u^2/2)\, du,$$

where $\Phi(y)$ is the lower tail probability for the standard normal distribution and argument y.

A.4 Special functions

A.4.1 Binomial coefficient

This is required in the treatment of discrete distributions. It is just the number of selections without regard to order of k objects out of n.

$$\begin{aligned} \binom{n}{k} &= \frac{n!}{k!(n-k)!} \\ &= \frac{n(n-1)(n-2)\cdots(n-k+1)}{k(k-1)(k-2)\cdots 3.2.1} \\ &= \binom{n}{n-k} \end{aligned}$$

A.4.2 Gamma and incomplete gamma functions

The gamma function is widely used in the treatment of continuous random variables and is defined as follows.

$$\begin{aligned} \Gamma(\alpha) &= \int_0^{\infty} t^{\alpha-1} \exp(-t)\, dt \\ \Gamma(\alpha+1) &= \alpha\Gamma(\alpha). \\ \text{So that } \Gamma(k) &= (k-1)! \text{ for integer } k \ge 0 \\ \text{and } \Gamma(k+1/2) &= (2k-1)(2k-3)\cdots 5.3.1.\sqrt{\pi}/2^k. \end{aligned}$$

The incomplete gamma function $P(x)$ and incomplete gamma function complement $Q(x)$ given parameter α usually, as here, normalized by the complete gamma function, are also frequently required.

$$\begin{aligned} P(x, \alpha) &= \frac{1}{\Gamma(\alpha)} \int_0^x t^{\alpha-1} \exp(-t)\, dt \\ Q(x, \alpha) &= \frac{1}{\Gamma(\alpha)} \int_x^{\infty} t^{\alpha-1} \exp(-t)\, dt \end{aligned}$$

As the gamma distribution function with $G > 0$, $\alpha > 0$, and $\beta > 0$ is

$$P(x, \alpha, \beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} \int_0^x G^{\alpha-1} \exp(-G/\beta)\, dG,$$

the incomplete gamma function is also the cumulative distribution function for a gamma distribution with second parameter β equal to one.

A.4.3 Beta and incomplete beta functions

Using the gamma function, the beta function is then defined as

$$B(g,h) = \frac{\Gamma(g)\Gamma(h)}{\Gamma(g+h)}$$

and the incomplete beta function for $0 \le x \le 1$ as

$$R(x,g,h) = \frac{1}{B(g,h)} \int_0^x t^{g-1}(1-t)^{h-1}\,dt.$$

The incomplete beta function is also the cumulative distribution function for the beta distribution.

A.4.4 Exponential integrals

$$E_1(x) = \int_x^\infty \frac{\exp(-t)}{t}\,dt$$

$$E_i(x) = -\int_{-x}^\infty \frac{\exp(-t)}{t}\,dt$$

where $x > 0$, excluding the origin in $Ei(x)$.

A.4.5 Sine and cosine integrals and Euler's gamma

$$Si(x) = \int_0^x \frac{\sin t}{t}\,dt$$

$$Ci(x) = \gamma + \log x + \int_0^x \frac{\cos t - 1}{t}\,dt,\ x > 0$$

$$\gamma = \lim_{m\to\infty}\{1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}\cdots + \frac{1}{m} - \log m\}$$

$$= .5772156649\ldots$$

A.4.6 Fermi-Dirac integrals

$$f(x) = \frac{1}{\Gamma(1+\alpha)}\int_0^\infty \frac{t^\alpha}{1+\exp(t-x)}\,dt$$

A.4.7 Debye functions

$$f(x) = \frac{n}{x^n}\int_0^x \frac{t^n}{\exp(t)-1}\,dt,\ x > 0, n \ge 1$$

A.4.8 Clausen integral

$$f(x) = -\int_0^x \log\left(2\sin\frac{t}{2}\right)\,dt, 0 \le x \le \pi$$

A.4.9 Spence integral

$$f(x) = \int_0^x \frac{-\log|1-t|}{t}\,dt$$

A.4.10 Dawson integral

$$f(x) = \exp(-x^2) \int_0^x \exp(t^2)\, dt$$

A.4.11 Fresnel integrals

$$C(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right)\, dt$$
$$S(x) = \int_0^x \sin\left(\frac{\pi}{2}t^2\right)\, dt$$

A.4.12 Polygamma functions

The polygamma function is defined in terms of the gamma function $\Gamma(x)$, or the Psi function (i.e. digamma function) $\Psi(x) = \Gamma'(x)/\Gamma(x)$

$$\begin{aligned}\psi^{(n)}(x) &= \frac{d^{n+1}}{dx^{n+1}} \log \Gamma(x) \\ &= \frac{d^n}{dx^n}\psi(x) \\ &= (-1)^{n+1} \int_0^\infty \frac{t^n \exp(-xt)}{1-\exp(-t)}\, dt.\end{aligned}$$

So the case $n = 0$ is the digamma function, the case $n = 1$ is the trigamma function, and so on.

A.4.13 Struve functions

$$H_\nu(z) = \frac{2(\frac{1}{2}z)^\nu}{\sqrt{\pi}\Gamma(\nu+\frac{1}{2})} \int_0^{\frac{\pi}{2}} \sin(z\cos\theta)\sin^{2\nu}\theta\, d\theta$$
$$L_\nu(z) = \frac{2(\frac{1}{2}z)^\nu}{\sqrt{\pi}\Gamma(\nu+\frac{1}{2})} \int_0^{\frac{\pi}{2}} \sinh(z\cos\theta)\sin^{2\nu}\theta\, d\theta$$

A.4.14 Kummer confluent hypergeometric functions

$$M(a,b,z) = 1 + \frac{az}{b} + \frac{(a)_2 z^2}{(b)_2 2!} + \cdots + \frac{(a)_n z^n}{(b)_n n!} + \cdots$$

where $(a)_n = a(a+1)(a+2)\ldots(a+n-1),\ (a)_0 = 1$

$$U(a,b,z) = \frac{\pi}{\sin \pi b}\left\{\frac{M(a,b,z)}{\Gamma(1+a-b)\Gamma(b)} - z^{1-b}\frac{M(1+a-b,2-b,z)}{\Gamma(a)\Gamma(2-b)}\right\}$$

$$\begin{aligned}U(a,n+1,z) = \frac{(-1)^{n+1}}{n!\Gamma(a-n)} &[M(a,n+1,z)\log z \\ &+ \sum_{r=0}^{\infty}\frac{(a)_r z^r}{(n+1)_r r!}\{\psi(a+r) - \psi(1+r) - \psi(1+n+r)\}\,] \\ &+ \frac{(n-1)!}{\Gamma(a)} z^{-n} M(a-n, 1-n, z)_n\end{aligned}$$

A.4.15 Abramovitz functions

The Abramovitz functions of order $n = 0, 1, 2$ are defined for $x \geq 0$ as

$$f(x) = \int_0^\infty t^n \exp(-t^2 - x/t)\, dt$$

A.4.16 Legendre polynomials

The Legendre polynomials $P_n(x)$ and P_n^m are defined in terms of the hypergeometric function F or Rodrigue's formula for $-1 \le x \le 1$ by

$$\begin{aligned}
P_n(x) &= F\left(-n, n+1, 1, \frac{1}{2}(1-x)\right) \\
&= \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \\
P_n^m(x) &= (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x).
\end{aligned}$$

A.4.17 Bessel, Kelvin, and Airy functions

$$\begin{aligned}
J_\nu(z) &= (\tfrac{1}{2}z)^\nu \sum_{k=0}^{\infty}\frac{(-\frac{1}{4}z^2)^k}{k!\Gamma(\nu+k+1)} \\
Y_\nu(z) &= \frac{J_\nu(z)\cos(\nu\pi) - J_{-\nu}(z)}{\sin(\nu\pi)} \\
I_\nu(z) &= (\tfrac{1}{2}z)^\nu \sum_{k=0}^{\infty}\frac{(\frac{1}{4}z^2)^k}{k!\Gamma(\nu+k+1)} \\
K_\nu(z) &= \tfrac{1}{2}\pi\frac{I_{-\nu}(z) - I_\nu(z)}{\sin(\nu\pi)} \\
\mathrm{ber}_\nu x + i\,\mathrm{bei}_\nu x &= \exp(\tfrac{1}{2}\nu\pi i)I_\nu(x\ \exp(\tfrac{1}{4}\pi i)) \\
\mathrm{ker}_\nu x + i\,\mathrm{kei}_\nu x &= \exp(-\tfrac{1}{2}\nu\pi i)K_\nu(x\ \exp(\tfrac{1}{4}\pi i)) \\
Ai(z) &= \tfrac{1}{3}\sqrt{z}[I_{-1/3}(\xi) - I_{1/3}(\xi)],\ \text{where } \xi = \tfrac{2}{3}z^{3/2} \\
Ai'(z) &= -\tfrac{1}{3}z[I_{-2/3}(\xi) - I_{2/3}(\xi)] \\
Bi(z) &= \sqrt{z/3}[I_{-1/3}(\xi) + I_{1/3}(\xi)] \\
Bi'(z) &= (z/\sqrt{3})[I_{-2/3}(\xi) + I_{2/3}(\xi)]
\end{aligned}$$

A.4.18 Elliptic integrals

$$\begin{aligned}
R_C(x,y) &= \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{t+x}\,(t+y)} \\
R_F(x,y,z) &= \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)}} \\
R_D(x,y,z) &= \frac{3}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)^3}} \\
R_j(x,y,z,\rho) &= \frac{3}{2}\int_0^\infty \frac{dt}{(t+\rho)\sqrt{(t+x)(t+y)(t+z)}} \\
u &= \int_0^\phi \frac{d\theta}{\sqrt{1-m\sin^2\theta}} \\
S_N(u|m) &= \sin\phi \\
C_N(u|m) &= \cos\phi \\
D_N(u|m) &= \sqrt{1-m\sin^2\phi}
\end{aligned}$$

A.4.19 Single impulse functions

These discontinuous functions all generate a single impulse, but some of them require special techniques for plotting, which are described on page 357.

A.4.19.1 Heaviside unit function

The Heaviside unit function $h(x - a)$ is defined as

$$\begin{aligned} h(x - a) &= 0, \quad \text{for } x < a \\ &= 1, \quad \text{for } x \geq a, \end{aligned}$$

so it provides a useful way to construct models that switch definition at critical values of the independent variable, in addition to acting in the usual way as a ramp function.

A.4.19.2 Kronecker delta function

The Kronecker delta function δ_{ij} is defined as

$$\begin{aligned} \delta_{ij} &= 0, \quad \text{for } i \neq j \\ &= 1, \quad \text{for } i = j, \end{aligned}$$

which can be very useful when constructing models with simple logical switches.

A.4.19.3 Unit impulse function

The single square wave impulse function $f(x, a, b)$ of width $2b > 0$ with unit area is defined as

$$\begin{aligned} f(x, a, b) &= 0, \quad \text{for } x < a - b, \; x > a + b \\ &= 1/2b, \quad \text{for } a - b \leq x \leq a + b, \end{aligned}$$

so it can be used to model the Dirac delta function by using extremely small values for b.

A.4.19.4 Unit spike function

The triangular spike function $f(x, a, b)$ of width $2b > 0$ with unit area is defined as

$$\begin{aligned} f(x, a, b) &= 0, \quad \text{for } x < a - b, \; x > a + b \\ &= (x - a + b)/b^2, \quad \text{for } a - b \leq x \leq a \\ &= (a + b - x)/b^2, \quad \text{for } a \leq x \leq a + b. \end{aligned}$$

A.4.19.5 Gauss pdf

The probability density function $f(x, a, b)$ for the normal distribution with unit area is defined for $b > 0$ as

$$f(x, a, b) = \frac{1}{\sqrt{2\pi}\, b} \exp\left\{ -\frac{1}{2} \left(\frac{x - a}{b} \right)^2 \right\},$$

which is very useful for modelling bell shaped impulse functions.

A.4.20 Periodic impulse functions

These generate pulses at regular intervals, and some of them require special techniques for plotting, as described on page 358.

A.4.20.1 Square wave function

This has an amplitude of one and period of $2a > 0$, and can be described for $t \geq 0$,in terms of the Heaviside unit function $h(t)$ as

$$f_1(t) = h(t) - 2h(t-a) + 2h(t-2a) - \cdots,$$

so it oscillates between plus and minus one at each x-increment of length a, with pulses of area plus and minus a.

A.4.20.2 Rectified triangular wave

This generates a triangular pulse of unit amplitude with period $2a > 0$ and can be defined for $t \geq 0$ as

$$f_2 = \frac{1}{a}\int_0^t f_1(u)\,du$$

so it consists of a series of contiguous isosceles triangles of area a.

A.4.20.3 Morse dot wave function

This is essentially the upper half of the square wave. It has a period of $2a > 0$ and can be defined for $t \geq 0$ by

$$f_3(t) = \frac{1}{2}[h(t) + f_1(t)] = \sum_{i=0}^{\infty}(-1)^i h(t-ia),$$

so it alternates between sections of zero and squares of area a.

A.4.20.4 Sawtooth wave function

This consists of half triangles of unit amplitude and period $a > 0$ and can be defined for $t \geq 0$ by

$$f_4(t) = \frac{t}{a} - \sum_{i=1}^{\infty} h(t-ia),$$

so it generates a sequence of right angled triangles with area $a/2$

A.4.20.5 Rectified sine wave function

This is just the absolute value of the sine function with argument at, that is

$$f_5(t) = |\sin at|,$$

so it has unit amplitude and period π/a.

A.4.20.6 Rectified sine half-wave function

This is just the positive part of the sine wave

$$f_6(t) = \frac{1}{2}[\sin at + |\sin at|]$$

so it has period $2\pi/a$.

A.4.20.7 Unit impulse wave function

This is a sequence of Dirac delta function pulses with unit area given by

$$f_7(t) = \sum_{i=1}^{\infty}\delta(i - at).$$

Of course, the width of the pulse has to be specified as a parameter b, so the function can be used to generate a spaced sequence of rectangular impulse with arbitrary width but unit area. Small values of b simulate pulses of the Dirac delta function at intervals of length a.

Appendix B

User defined models

B.1 Supplying models as a dynamic link library

This is still the best method to supply your own models, but it requires programming skill. You can write in a language, such as Fortran or C, or even use assembler. Using this technique, you can specify your own numerical analysis library, or even reference other dynamic link libraries, as long as all the locations and addresses are consistent with the other SimFIT dynamic link libraries. Since the development of program **usermod** it is now only necessary to create new entries in the dynamic link library for convenience, or for very large sets of differential equations, or for models requiring special functions not supported by **w_models.dll**.

B.2 Supplying models using standard mathematical notation

For simple models it is very easy to define equations using standard mathematical notation. For instance, a line would require

```
begin{expression}
f(1) = p(1) + p(2)x
end{expression}
```

while a quadratic would need

```
begin{expression}
f(1) = p(1) + p(2)x + p(3)x^2
end{expression}
```

and a 2:2 rational function could be formulated as follows

```
begin{expression}
f(1) = [p(1)x + p(2)x^2]/[1.0 + p(3)x + p(4)x^2].
end{expression}
```

The rule is that equations inside `begin{expression}` and `end{expression}` tokens can use all the standard mathematical expressions then, at execution time, SimFIT simply transforms the code into reverse Polish to create a virtual stack. Note that there are many test files illustrating this technique and, after parsing, the resulting reverse Polish expression is written to a file called `f$parser.tmp` in the temporary folder, should it be required to consult it or archive it. Test files using expressions in this way have an underscore e character, for example `usermod1_e.tf?` defines the same model as `usermod1.tf?`

Inside expressions it is advisable to make liberal use of * for multiplying and brackets {.}, [.], and (.) to avoid ambiguities. For example the following advice should be noted.

- Instead of `xlogx` use `x*log(x)`.

- Instead of `1/2pi` use `1/(2pi)`.
- Instead of `e^-kt` use `exp(-kt)`.

Full details for this method can be found in the file `w_readme.f10` and note that program **usermod** now assumes the new user-defined model files will be created using this technique. Further examples are presented on page 453.

B.3 Supplying models as ASCII text files

The method that has been developed for the SimFIT package works extremely well, and can be used to create very complex model equations, even though it requires no programming skill. You can use all the usual mathematical functions, including trigonometric and hyperbolic functions, as well as the gamma function, the log-gamma function, the normal probability integral and the erfc function. The set of allowable functions will increase as **w_models.dll** is upgraded. The essence of the method is to supply an ASCII text file containing a set of instructions in reverse Polish, that is, postfix, or last in first out, which will be familiar to all programmers, since it is, after all, essentially the way that computers evaluate mathematical expressions. Using reverse Polish, any explicit model can be written as a series of sequential instructions without using any brackets. Just think of a stack to which arguments are added and functions which operate on the top item of the stack. Suppose the top item is the current value of x and the operator log is added, then this will replace x by $\log(x)$ as the current item on the top of the stack. What happens is that the model file is read in, checked, and parsed just once to create a virtual instruction stack. This means that model execution is very rapid, since the file is only read once, but also it allows users to optimize the stack operations by rolling, duplicating, storing, retrieving, and so on, which is very useful when developing code with repeated subexpressions, such as occur frequently with systems of differential equation and Jacobians.

So, to supply a model this way, you must create an ASCII text file with the appropriate model or differential equation. This is described in the `w_readme.?` files and can be best understood by browsing the test files supplied, i.e. `usermod1.tf?` for functions of one variable, `usermod2.tf?` for functions of two variables, `usermod3.tf?` for functions of three variables and `usermodd.tf?` for single differential equations. The special program **usermod** should be used to develop and test your models before trying them with **makdat** or **qnfit**. Note that **usermod** checks your model file for syntax errors, but it also allows you to evaluate the model, plot it, or even use it to find areas or zeros of n functions in n unknowns.

Note that new syntax is being developed for this method, as described in the `w_readme.*` files. For instance `put` and `get` commands considerably simplify the formulation of models with repeated sub-expressions.

Further details about the performance of this method for supplying mathematical models as ASCII text files can be found in an article by Bardsley, W.G. and Prasad, N. in Computers and Chemistry (1997) 21, 71–82.

B.3.1 Formatting conventions for user defined models

Please observe the use of the special symbol % in model files. The symbol % starting a line is an escape sequence to indicate a change in the meaning of the input stream, e.g., from text to parameters, from parameters to model instructions, from model to Jacobian, etc. Characters occurring to the right of the first non-blank character are interpreted as comments and text here is ignored when the model is parsed. The % symbol *must* be used to indicate:-

i) start of the file
ii) start of the model parameters
iii) start of the model equations
iv) end of model equations (start of Jacobian with diff. eqns.)

The file you supply must have *exactly* the format now described.

a) The file must start with a % symbol indicating where text starts The next lines must be the name/details you choose for the model. This would normally be at least 4 and not greater than 24 lines. This text is only to identify the model and is not used by SimFIT. The end of this section is marked by a % symbol. The next three lines define the type of model.

b) The first of these lines must indicate the number of equations in the model, e.g., 1 equation, 2 equations, 3 equations, etc.

c) The next must indicate the number of independent variables as in:- 1 variable, 2 variables, 3 variables, etc. or else it could be differential equation to indicate that the model is one or a set of ordinary differential equations with one independent variable.

d) The next line must define the number of parameters in the model.

e) With differential equations, the last parameters are reserved to set the values for the integration constants $y_0(i)$, which can be either estimated or fixed as required. For example, if there are n equations and m parameters are declared in the file, only $m - n$ can be actually used in the model, since $y_0(i) = p(m - n + i)$ for $i = 1, 2, ..., n$.

f) Lines are broken up into tokens by spaces.

g) Only the first token in each line matters after the model starts.

h) Comments begin with % and are added just to explain what's going on.

i) Usually the comments beginning with a % can be omitted.

j) Critical lines starting with % must be present as explained above.

k) The model operations then follow, one per line until the next line starting with a % character indicates the end of the model.

l) Numbers can be in any format, e.g., 2, 1.234, 1.234E-6, 1.234E6

m) The symbol f(i) indicates that model equation i is evaluated at this point.

n) Differential equations can define the Jacobian after defining the model. If there are n differential equations of the form

$$\frac{dy}{dx} = f(i)(x, y(1), y(2), ..., y(n))$$

then the symbol $y(i)$ is used to put $y(i)$ on the stack and there must be a n by n matrix defined in the following way. The element $J(a, b)$ is indicated by putting $j(n(b - 1) + a)$ on the stack. That is the columns are filled up first. For instance with 3 equations you would have a Jacobian $J(i, j) = df(i)/dy(j)$ defined by the sequence:

```
J(1,1) = j(1),  J(1,2) = j(4),  J(3,1) = j(7)
J(2,1) = j(2),  J(2,2) = j(5),  J(3,2) = j(8)
J(3,1) = j(3),  J(3,2) = j(6),  J(3,3) = j(9)
```

B.3.1.1 Table of user-defined model commands

```
Command        Effects produced

x              stack -> stack, x
y              stack -> stack, y
z              stack -> stack, z
add            stack, a, b -> stack, (a + b)
subtract       stack, a, b -> stack, (a - b)
multiply       stack, a, b -> stack, (a*b)
```

```
divide         stack, a, b -> stack, (a/b)
p(i)           stack -> stack, p(i)         ...  i can be 1, 2, 3, etc
f(i)           stack, a -> stack  ...evaluate model since now f(i) = a
power          stack, a, b -> stack, (a^b)
squareroot     stack, a -> stack, sqrt(a)
exponential    stack, a -> stack, exp(a)
tentothepower  stack, a -> stack, 10^a
ln (or log)    stack, a -> stack, ln(a)
log10          stack, a -> stack, log(a) (to base ten)
pi             stack -> stack, 3.1415927
sine           stack, a -> stack, sin(a)       ... radians not degrees
cosine         stack, a -> stack, cos(a)       ... radians not degrees
tangent        stack, a -> stack, tan(a)       ... radians not degrees
arcsine        stack, a -> stack, arcsin(a)    ... radians not degrees
arccosine      stack, a -> stack, arccos(a)    ... radians not degrees
arctangent     stack, a -> stack, arctan(a)    ... radians not degrees
sinh           stack, a -> stack, sinh(a)
cosh           stack, a -> stack, cosh(a)
tanh           stack, a -> stack, tanh(a)
exchange       stack, a, b -> stack, b, a
duplicate      stack, a -> stack, a, a
pop            stack, a, b -> stack, a
absolutevalue  stack, a -> stack, abs(a)
negative       stack, a -> stack , -a
minimum        stack, a, b -> stack, min(a,b)
maximum        stack, a, b -> stack, max(a,b)
gammafunction  stack, a -> stack, gamma(a)
lgamma         stack, a -> stack, ln(gamma(a))
normalcdf      stack, a -> stack, phi(a)  integral from -infinity to a
erfc           stack, a -> stack, erfc(a)
y(i)           stack    -> stack, y(i)                 Only diff. eqns.
j(i)           stack, a -> stack  J(i-(i/n), (i/n)+1) Only diff. eqns.
***            stack -> stack, ***          ...  *** can be any number
```

B.3.1.2 Table of synonyms for user-defined model commands

The following sets of commands are equivalent:-

```
sub, minus, subtract
mul, multiply
div, divide
sqrt, squarero, squareroot
exp, exponent, exponential
ten, tentothe, tentothepower
ln, log
sin, sine
cos, cosine
tan, tangent
asin, arcsin, arcsine
acos, arccos, arccosine
atan, arctan, arctangent
dup, duplicate
exch, exchange, swap
del, delete, pop
abs, absolute
```

```
neg, negative
min, minimum
max, maximum
phi, normal, normalcd, normalcdf
abserr, epsabs
relerr, epsrel
middle, mid
```

B.3.1.3 Error handling in user defined models

As the stack is evaluated, action is taken to avoid underflow, overflow and forbidden operations, like $1/x$ as x tends to zero or taking the log or square root of a negative number etc. This should never be necessary, as users should be able to design the fitting or simulation procedures in such a way that such singularities are not encountered, rather than relying on default actions which can lead to very misleading results.

B.3.1.4 Notation for functions of more than three variables

The model for nine equations in nine unknowns coded in `usermodn.tf4` is provided to show you how to use **usermod** to find a zero vector for a system of nonlinear equations. It illustrates how to code for n functions of m variables, and shows how to use $y(1), y(2), \ldots, y(m)$ instead of x, y, z, etc. The idea is that, when there are more than three variables, you should not use x, y or z in the model file, you should use $y(1), y(2)$ and $y(3)$, etc.

B.3.1.5 The commands `put(.)` and `get(.)`

These facilitate writing complicated models that re-use previously calculated expressions (very common with differential equations).
The commands are as follows

```
put(i)        : move the top stack element into storage location i
get(j)        : transfer storage element j onto the top of the stack
```

and the following code gives an example.

```
x
put(11)
get(11)
get(11)
multiply
put(23)
get(11)
get(23)
add           : now (x + x^2) has been added to the top of the stack
```

It should be observed that these two commands reference a global store. This is particularly important when a main model calls sub-models for evaluation, root finding or quadrature, as it provides a way to communicate information between the models.

B.3.1.6 The command `get3(.,.,.)`

Often a three way branch is required where the next step in a calculation depends on whether a critical expression is negative, zero or positive, e.g., the nature of the roots of a quadratic equation depend on the value of the discriminant. The way this is effected is to define the three possible actions and store the results using three `put` commands. Then use a `get3(i,j,k)` command to pop the top stack element and invoke `get(i)` if the element is negative, `get(j)` if it is zero (to machine precision), or `get(k)` otherwise. For example, the model `updown.mod`, illustrated on page 356, is a simple example of how to use a `get3(.,.,.)`

command to define a model which swaps over from one equation to another at a critical value of the independent variable. This command can be used after the command `order` has been used to place a $-1, 0$ or 1 on the stack, and the model `updownup.mod` illustrates how to use `order` with `value3` to create a model with two swap-over points.

```
x
negative
put(5)
1.0e-100
put(6)
x
put(7)
x
get3(5,6,7)     : now the top stack element is |x|, or 1.0e-100
```

B.3.1.7 The commands epsabs and epsrel

When the evaluation of a model requires iterative procedures, like root finding or quadrature, the absolute and relative error tolerances must be specified. Default values (1.0e-6 and 1.0e-3) are initialized and these should suffice for most purposes. However you may need to specify alternative values by using the `epasabs` or `epsrel` commands with difficult problems, or to control the accuracy used in a calculation. For instance, when fitting a model that involves the evaluation of a multiple integral as a sub-model, you would normally use fairly large values for the error tolerances until a good fit has been found, then decrease the tolerances for a final refinement. Values added to the stack to define the tolerances are popped.

```
1.0e-4
epsabs
1.0e-2
epsrel          : now epsabs = 0.0001 and epsrel = 0.01
```

B.3.1.8 The commands `blim(.)` and `tlim(.)`

When finding roots of equations it is necessary to specify starting limits and when integrating by quadrature it is necessary to supply lower and upper limits of integration. The command `blim(i)` sets the lower limit for variable i while the command `tlim(j)` sets the upper limit for variable j. Values added to the stack to define the limits are popped.

```
0
blim(1)
0
blim(2)
pi
tlim(1)
pi
2
multiply
tlim(2)         :limits are now 0 < x < 3.14159 and 0 < y < 6.28318
```

B.3.2 Plotting user defined models

Once a model has been checked by program **usermod** it can be plotted directly if it is a function of one variable, a function of two variables, a parametric curve in $r(\theta)$ format (page 362), $x(t), y(t)$ format (page 354), or a space curve in $x(t), y(t), z(t)$ format (page 355). This is also be a very convenient way to simulate families of curves described as separate functions of one variable, as will be readily appreciated by reading in the test file `usermodn.tf1`, which defines four trigonometric functions of one variable.

B.3.3 Finding zeros of user defined models

After a model function of one variable has been checked by program **usermod** it is possible to locate zeros of the equation

$$f(x) - k = 0$$

for fixed values of k. It is no use expecting this root finding to work unless you know the approximate location of a root and can supply two values A, B that bracket the root required, in the sense that

$$f(A)f(B) < 0.$$

For this reason, it is usual to simulate the model first and observe the plot until two sensible limits A, B are located. Try `usermod.tf1` which is just a straight line to get the idea. Note that in difficult cases, where IFAIL is not returned as zero, it will be necessary to adjust EPSABS and EPSREL, the absolute and relative error tolerances.

B.3.4 Finding zeros of n functions in n variables

When a model file defining n functions of n unknowns has been checked by program **usermod** it is possible to attempt to find a n-vector solution given n starting estimates, if $n > 1$. As the location of such vectors uses iterative techniques, it is only likely to succeed if sensible starting estimates are provided. As an example, try the model file `usermodn.tf4` which defines nine equations in nine unknowns. Note that to obtain IFAIL equal to zero, i.e. a satisfactory solution, you will have to experiment with starting estimates. Observe that these are supplied using the **usermod** y vector, not the parameter vector p. Try setting the nine elements of the y vector to zero, which is easily done from a menu.

B.3.5 Integrating 1 function of 1 variable

After using program **usermod** to check a function of one variable, definite integration over fixed limits can be done by two methods. Simpson's rule is used, because users may wish to embed a straightforward Simpson's rule calculation in a model, but also adaptive quadrature is used, in case the integral is ill conditioned, e.g., has spikes or poles. Again preliminary plotting is recommended for ill-conditioned functions. Try `usermod1.tf1` to see how it all works.

B.3.6 Integrating n functions of m variables

When a model defining n functions of m variables has been successfully parsed by program **usermod**, adaptive integration can be attempted if $m > 1$. For this to succeed, the user must set the m lower limits (blim) and the m upper limits (tlim) to sensible values, and it probably will be necessary to alter the error tolerances for success (i.e. zero IFAIL). Where users wish to embed the calculation of an adaptive quadrature procedure inside a main model, it is essential to investigate the quadrature independently, especially if the quadrature is part of a sub-model involving root finding. Try `usermod4.tf1` which is a single four dimensional integral (i.e. $n = 1$ and $m = 4$) that should be integrated between zero and one for all four variables. Observe, in this model, that $y(1), y(2), y(3), y(4)$ are the variables, because $m > 3$.

B.3.7 Calling sub-models from user-defined models

B.3.7.1 The command `putpar`

This command is used to communicate parameters $p(i)$ to a sub-model. It must be used to transfer the current parameter values into global storage locations if it is wished to use them in a subsidiary model. Unless the command `putpar` is used in a main model, the sub-models have no access to parameter values to enable the command $p(i)$ to add parameter i to the sub-model stack. The stack length is unchanged. Note that the storage locations for `putpar` are initialized to zero so, if you do not use `putpar` at the start of the main model, calls to $p(i)$ in subsequent sub-models will not lead to a crash, they will simply use $p(i) = 0$. The command `putpar` cannot be used in a subsidiary model, of course.

B.3.7.2 The command `value(.)`

This command is used to evaluate a subsidiary model. It uses the current values for independent variables to evaluate subsidiary model i. The stack length is increased by one, as the value returned by function evaluation is added to the top of the stack. The command `putpar` must be used before `value(i)` if it wished to use the main parameters in subsidiary model number i.

It is important to make sure that a subsidiary model is correct, by testing it as a main model, if possible, before using it as a subsidiary model. You must be careful that the independent variables passed to the sub-model for evaluation are the ones intended. Of course, value can call sub-models which themselves can call `root`, and/or `quad`.

B.3.7.3 The command `quad(.)`

This command is used to estimate an integral by adaptive quadrature. It uses the `epsabs`, `epsrel`, `blim` and `tlim` values to integrate model i and place the return value on the stack. The values assigned to the `blim` and `tlim` arrays are the limits for the integration. If the model i requires j independent variables then j `blim` and `tlim` values must be set before `quad(i)` is used. The length of the stack increases by one, the return value placed on the top of the stack. The command `putpar` must be used before `quad(i)` if it is wished to use the main parameters in the subsidiary model number i.

Adaptive quadrature cannot be expected to work correctly if the range of integration includes sharp spikes or long stretches of near-zero values, e.g., the extreme upper tail of a decaying exponential. The routines used (D01AJF and D01EAF) cannot really handle infinite ranges, but excellent results can be obtained using commonsense extreme limits, e.g., several relaxation times for a decaying exponential. With difficult problems it will be necessary to increase `epsrel` and `epsabs`.

B.3.7.4 The command `convolute(.,.)`

When two or sub-models have been defined, say $model(i) = f(x)$ and $model(j) = g(x)$, a special type of adaptive quadrature, which is actually a special case of the `quad(.)` command just explained, can be invoked to evaluate the convolution integral

$$\begin{aligned} f * g &= \int_0^x f(u)g(x-u)\,du \\ &= g * f \end{aligned}$$

using the command `convolute(i,j)`. To illustrate this type of model, consider the convolution of an exponential input function and gamma response function defined by the test file `convolve.mod` and listed as Example 8 later.

B.3.7.5 The command `root(.)`

This command is used to estimate a zero of a sub-model iteratively. It uses the `epsabs`, `epsrel`, `blim` and `tlim` values to find a root for model i and places the return value on the stack. The values assigned to `blim(1)` and `tlim(1)` are the limits for root location. The length of the stack increases by one, the root value placed on the top of the stack. The command `putpar` must be used before `root(i)` if it wished to use the main parameters in the subsidiary model i.

The limits `A = blim(1)` and `B = tlim(1)` are used as starting estimates to bracket the root. If $f(A)*f(B) > 0$ then the range (A, B) is expanded by up to ten orders of magnitude (without changing `blim(1)` or `tlim(1)`) until $f(A) * f(B) < 0$. If this or any other failure occurs, the root is returned as zero. Note that A and B will not change sign, so you can search for, say, just positive roots. If this is too restrictive, make sure `blim(1)*tlim(1) < 0`. C05AZF is used, and with difficult problems it will be necessary to increase `epsrel`.

B.3.7.6 The command `value3(.,.,.)`

This is a very powerful command which is capable of many applications of the form: if ... elseif ... else. If the top stack number is negative `value(i)` is called, if it is zero (to machine precision), `value(j)` is called, and if it is positive `value(k)` is called. It relies on the presence of correctly formatted sub-models i, j and k of course, but the values returned by sub-models i, j and k are arbitrary, as almost any code can be employed in models i, j and k. The top value of the stack is replaced by the value returned by the appropriate sub-model. This command is best used in conjunction with the command `order`, which places either $-1, 0$ or 1 on the stack.

B.3.7.7 The command `order`

Given a lower limit, an initial value, and an upper limit, this command puts -1 on the stack for values below the lower limit, puts 0 on the stack for values between the limits, and puts 1 on the stack for values in excess of the upper limit.

```
0
x
4
order
value3(1,2,3)
f(1)
```

This code is used in the model `updownup.mod` to generate a model that changes definition at the critical swap-over points $x = 0$ and $x = 4$.

To summarize, the effect of this command is to replace the top three stack elements, say a, w, b where $a < b$ by either -1 if $w \le a$, 0 if $a < w \le b$, or 1 if $w > b$, so reducing the stack length by two.

B.3.7.8 The command `middle`

Given a lower limit, an initial value, and an upper limit, this command reflects values below the lower limit back up to the lower limit and decreases values in excess of the upper limit back down to the upper limit, but leaves intermediate values unchanged. For example, the code

```
0
x
1
middle
```

will always place a value w on the stack for $0 \le w \le 1$, and $w = x$ only if $0 \le x \le 1$.

To summarize, the effect of this command is to replace the top three stack elements, say a, w, b where $a < b$ by either a if $w \le a$, w if $a < w \le b$ or b if $w > b$, so reducing the stack length by two.

B.3.7.9 The syntax for subsidiary models

The format for defining sub-models is very strict and must be used exactly as now described. Suppose you want to use n independent equations. Then n separate user files are developed and, when these are tested, they are placed in order directly after the end of the main model, each surrounded by a `begin{model(i)}` and `end{model(i)}` command. So, if you want to use a particular model as a sub-model, you first of all develop it using program usermod then, when it is satisfactory, just add it to the main model. However, note that sub-models are subject to several restrictions.

B.3.7.10 Rules for using sub-models

- Sub-model files must be placed in numerically increasing order at the end of the main model file. Model parsing is abandoned if a sub-model occurs out of sequence.
- There must be no spaces or non-model lines between the main model and the subsidiary models, or between any subsequent sub-models.
- Sub-models cannot be differential equations.
- Sub-models of the type being described must define just one equation.
- Sub-models are not tested for consistent put and get commands, since puts might be defined in the main model, etc.
- Sub-models cannot use `putpar`, since `putpar` only has meaning in a main model.
- Sub-models can use the commands `value(i)`, `root(j)` and `quad(k)`, but it is up to users to make sure that all calls are consistent.
- When the command `value(i)` is used, the arguments passed to the sub-model for evaluation are the independent variables at the level at which the command is used. For instance if the main model uses `value(i)` then `value(i)` will be evaluated at the x, y, z, etc. of the main model, but with `model(i)` being used for evaluation. Note that $y(k)$ must be used for functions with more than three independent variables, i.e. when x, y and z no longer suffice. It is clear that if a model uses `value(i)` then the number of independent variables in that model must be equal to or greater than the number of independent variables in sub-model(i).
- When the commands `root(i)` and `quad(j)` are used, the independent variables in the sub-model numbers i and j are always dummy variables.
- When developing models and subsidiary models independently you may get error messages about x not being used, or a `get` without a corresponding `put`. Often these can be suppressed by using a `pop` until the model is developed. For instance x followed by `pop` will silence the message about x not being used.

B.3.7.11 Nesting subsidiary models

The subsidiary models can be considered to be independent except when there is a clash that would lead to recursion. For instance, `value(1)` can call `model(1)` which uses `root(2)` to find a root of `model(2)`, which calls `quad(3)` to integrate `model(3)`. However, at no stage can there be simultaneous use of the same model as `value(k)`, and/or `quad(k)`, and/or `root(k)`. The same subsidiary model cannot be used by more than any one instance of `value`, `quad`, `root` at the same time. Just commonsense really, virtual stack k for model k can only be used for one function evaluation at a time. Obviously there can be at most one instance each of `value`, `root` and `quad` simultaneously.

B.3.7.12 IFAIL values for D01AJF, D01AEF and C05AZF

Since these iterative techniques may be used inside optimization or numerical integration procedures, the soft fail option IFAIL = 1 is used. If the SIMFIT version of these routines is used, a silent exit will occur and failure will not be communicated to users. So it is up to users to be very careful that all calls to quadrature and root finding are consistent and certain to succeed. Default function values of zero are returned on failure.

B.3.7.13 Test files illustrating how to call sub-models

The test files `usermodx.tf?` give numerous examples of how to use sub-models for function evaluation, root finding, and adaptive quadrature.

B.3.8 Calling special functions from user-defined models

The special functions commonly used in numerical analysis, statistics, mathematical simulation and data fitting, can be called by one-line commands as in this table.

B.3.8.1 Table of special function commands

Command	NAG	Description
arctanh(x)	S11AAF	Inverse hyperbolic tangent
arcsinh(x)	S11AAF	Inverse hyperbolic sine
arccosh(x)	S11AAF	Inverse hyperbolic cosine
ai(x)	S17AGF	Airy function Ai(x)
dai(x)	S17AJF	Derivative of Ai(x)
bi(x)	S17AHF	Airy function Bi(x)
dbi(x)	S17AKF	Derivative of Bi(x)
besj0(x)	S17AEF	Bessel function J0
besj1(x)	S17AFF	Bessel function J1
besy0(x)	S17ACF	Bessel function Y0
besy1(x)	S17ADF	Bessel function Y1
besi0(x)	S18ADF	Bessel function I0
besi1(x)	S18AFF	Bessel function I1
besk0(x)	S18ACF	Bessel function K0
besk1(x)	S18ADF	Bessel function K1
phi(x)	S15ABF	Normal cdf
phic(x)	S15ACF	Normal cdf complement
erf(x)	S15AEF	Error function
erfc(x)	S15ADF	Error function complement
dawson(x)	S15AFF	Dawson integral
ci(x)	S13ACF	Cosine integral Ci(x)
si(x)	S13ADF	Sine integral Si(x)
e1(x)	S13AAF	Exponential integral E1(x)
ei(x)	......	Exponential integral Ei(x)
rc(x,y)	S21BAF	Elliptic integral RC
rf(x,y,z)	S21BBF	Elliptic integral RF
rd(x,y,z)	S21BCF	Elliptic integral RD
rj(x,y,z,r)	S21BDF	Elliptic integral RJ
sn(x,m)	S21CAF	Jacobi elliptic function SN
cn(x,m)	S21CAF	Jacobi elliptic function CN
dn(x,m)	S21CAF	Jacobi elliptic function DN
ln(1+x)	S01BAF	ln(1 + x) for x near zero
mchoosen(m,n)	......	Binomial coefficient
gamma(x)	S13AAF	Gamma function
lngamma(x)	S14ABF	log Gamma function
psi(x)	S14ADF	Digamma function, (d/dx)log(Gamma(x))
dpsi(x)	S14ADF	Trigamma function, (d^2/dx^2)log(Gamma(x))
igamma(x,a)	S14BAF	Incomplete Gamma function
igammac(x,a)	S14BAF	Complement of Incomplete Gamma function
fresnelc(x)	S20ADF	Fresnel C function
fresnels(x)	S20ACF	Fresnel S function
bei(x)	S19ABF	Kelvin bei function
ber(x)	S19AAF	Kelvin ber function
kei(x)	S19ADF	Kelvin kei function
ker(x)	S19ACF	Kelvin ker function
cdft(x,m)	G01EBF	cdf for t distribution

```
cdfc(x,m)      G01ECF cdf for chi-square distribution
cdff(x,m,n)    G01EDF cdf for F distribution (m = num, n = denom)
cdfb(x,a,b)    G01EEF cdf for beta distribution
cdfg(x,a,b)    G01EFF cdf for gamma distribution
invn(x)        G01FAF inverse normal
invt(x,m)      G01FBF inverse t
invc(x,m)      G01FCF inverse chi-square
invb(x,a,b)    G01FEF inverse beta
invg(x,a,b)    G01FFF inverse gamma
spence(x)      ...... Spence integral: 0 to x of -(1/y)log|(1-y)|
clausen(x)     ...... Clausen integral: 0 to x of -log(2*sin(t/2))
struveh(x,m)   ...... Struve H function order m (m = 0, 1)
struvel(x,m)   ...... Struve L function order m (m = 0, 1)
kummerm(x,a,b)...... Confluent hypergeometric function M(a,b,x)
kummeru(x,a,b)...... U(a,b,x), b = 1 + n, the logarithmic solution
lpol(x,m,n)    ...... Legendre polynomial of the 1st kind, P_n^m(x),
                      -1 =< x =< 1, 0 =< m =< n
abram(x,m)     ...... Abramovitz function order m (m = 0, 1, 2), x > 0,
                      integral: 0 to infinity of t^m exp( - t^2 - x/t)
debye(x,m)     ...... Debye function of order m (m = 1, 2, 3, 4)
                      (m/x^m)[integral: 0 to x of t^m/(exp(t) - 1)]
fermi(x,a)     ...... Fermi-Dirac integral (1/Gamma(1 + a))[integral:
                      0 to infinity t^a/(1 + exp(t - x))]
heaviside(x,a)...... Heaviside unit function h(x - a)
delta(i,j)     ...... Kronecker delta function
impulse(x,a,b)...... Unit impulse function (small b for Dirac delta)
spike(x,a,b)   ...... Unit triangular spike function
gauss(x,a,b)   ...... Gauss pdf
sqwave(x,a)    ...... Square wave amplitude 1, period 2a
rtwave(x,a)    ...... Rectified triangular wave amplitude 1, period 2a
mdwave(x,a)    ...... Morse dot wave amplitude 1, period 2a
stwave(x,a)    ...... Sawtooth wave amplitude 1, period a
rswave(x,a)    ...... Rectified sine wave amplitude 1, period pi/a
shwave(x,a)    ...... Sine half-wave amplitude 1, period 2*pi/a
uiwave(x,a,b)  ...... Unit impulse wave area 1, period a, width b
```

Also, to allow users to document their models, all lines starting with a !, a / or a * character within models are ignored and treated as comment lines.

Any of the above commands included as a line in a SimFIT model or sub-model simply takes the top stack element as argument and replaces it by the function value. The NAG routines indicated can be consulted for details, as equivalent routines, agreeing very closely with the NAG specifications, are used. The soft fail (IFAIL = 1) options have been used so the simulation will not terminate on error condition, a default value will be returned. Obviously it is up to users to make sure that sensible arguments are supplied, for instance positive degrees of freedom, *F* or chi-square arguments, etc. To help prevent this problem, and to provide additional opportunities, the command `middle` (synonym mid) is available.

B.3.8.2 Using the command `middle` with special functions

Given a lower limit, an initial value, and an upper limit, this command reflects values below the lower limit back up to the lower limit and decreases values in excess of the upper limit back down to the upper limit, but leaves intermediate values unchanged. For example, the code

```
5
0.001
```

```
 x
 0.999
 middle
 invn(x,n)
```

will always return a zero IFAIL when calculating a percentage point for the t distribution with 5 degrees of freedom, because the argument will always be in the range $(0.001, 0.999)$ whatever the value of x.

B.3.8.3 Special functions with one argument

The top stack element will be popped and used as an argument, so the routines can be used in several ways. For instance the following code

```
 x
 phi(x)
 f(1)
```

would simulate model 1 as a normal cdf, while the code

```
 get(4)
 phi(x)
 f(3)
```

would return model three as the normal cdf for whatever was stored in storage location 4.

B.3.8.4 Special functions with two arguments

The top stack element is popped and used as x, while the second is popped and used as variable a, n, or y, as the case may be. For instance the code

```
 10
 x
 cdft(x,n)
```

would place the t distribution cdf with 10 degrees of freedom on the stack, while the code

```
 5
 0.975
 invt(x,n)
```

would place the critical value for a two-tailed t test with 5 degrees of freedom at a confidence level of 95% on the stack.
Another simple example would be

```
p(1)
x
heavi(x,a)
f(1)
```

which would return the function value 0 for $x < p(1)$ but 1 otherwise.

B.3.8.5 Special functions with three or more arguments

The procedure is a simple extension of that described for functions of two arguments. First the stack is prepared as $\ldots u, v, w, z, y, x$ but, after the function call, it would be $\ldots u, v, w, f(x, y, z)$. For example, the code

```
z
y
x
rf(x,y,z)
f(11)
```

would return model 11 as the elliptic function RF, since `f(11)` would have been defined as a function of at least three variables. However, the code

```
get(3)
get(2)
get(1)
rd(x,y,z)
1
add
f(7)
```

would define `f(7)` as one plus the elliptic function RD evaluated at whatever was stored in locations 3 (i.e. z), 2 (i.e. y) and 1 (i.e. x).

B.3.8.6 Test files illustrating how to call special functions

Three test files have been supplied to illustrate these commands:

- `usermods.tf1`: special functions with one argument
- `usermods.tf2`: special functions with two arguments
- `usermods.tf3`: special functions with three arguments

These should be used in program **usermod** by repeatedly editing, reading in the edited files, simulating, etc. to explore the options. Users can choose which of the options provided is to be used, by simply uncommenting the desired option and leaving all the others commented. Note that these are all set up for `f(1)` as a function of one variable and that, by commenting and removing comments so that only one command is active at any one time, the models can be plotted as continuous functions. Alternatively singly calculated values can be compared to tabulated values, which should be indistinguishable if your editing is correct.

B.3.9 Operations with scalars and vectors

B.3.9.1 The command `store(j)`

This command is similar to the `put(j)` command, but there is an important difference; the command `put(j)` is executed every time the model is evaluated, but the command `store(j)` is only executed when the model file is parsed for the first time. So `store(j)` is really equivalent to a data initialization statement at compile time. For instance, the code

```
3
store(14)
```

would initialize storage location 14 to the value 3. If no further `put(14)` is used, then storage location 14 would preserve the value 3 for all subsequent calculations in the main model or any sub-model, so that storage location 14 could be regarded as a global constant. Of course any `put(14)` in the main model or any sub-model would overwrite storage location 14. The main use for the store command is to define special values that are to be used repeatedly for model evaluation, e.g., coefficients for a Chebyshev expansion. For this reason there is another very important difference between `put(j)` and `store(j)`; `store(j)` must be preceded by a literal constant, e.g., 3.2e-6, and cannot be assigned as the end result of a calculation, because storage initialization is done before calculations.

To summarize: `store(j)` must be preceded by a numerical value, when it pops this top stack element after copying it into storage location j. So the stack length is decreased by one, to initialize storage location j, but only on the first pass through the model, i.e. when parsing.

B.3.9.2 The command `storef(file)`

Since it is tedious to define more than a few storage locations using the command `store(j)`, the command `storef(*.*)`, for some named file instead of *.*, provides a mechanism for initializing an arbitrary number of storage locations at first pass using contiguous locations. The file specified by the `storef(*.*)` command is read and, if it is a SimFIT vector file, all the successive components are copied into corresponding storage locations. An example of this is the test model file `cheby.mod` (and the related data file `cheby.dat`) which should be run using program **usermod** to see how a global vector is set up for a Chebshev approximation. Other uses could be when a model involves calculations with a set of fixed observations, such as a time series.

To summarize: the command `storef(mydatya)` will read the components of any n-dimensional SimFIT vector file, `mydata`, into n successive storage locations starting at position 1, but only when the model file is parsed at first pass. Subsequent use of `put(j)` or `store(j)` for j in the range $(1, n)$ will overwrite the previous effect of `storef(mydata)`.

B.3.9.3 The command `poly(x,m,n)`

This evaluates m terms of a polynomial by Horner's method of nested multiplication, with terms starting at `store(n)` and proceeding as far as `store(n + m - 1)`. The polynomial will be of degree $m - 1$ and it will be evaluated in ascending order. For example, the code

```
1
store(10)
0
store(11)
-1
store(12)
10
3
x
poly(x,m,n)
```

will place the value of $f(x) = 1 - x^2$ on the stack, where x is the local argument. Of course, the contents of the storage locations can also be set by `put(j)` commands which would then overwrite the previous `store(j)` command. For instance, the following code

```
5
put(12)
10
3
2
poly(x,m,n)
```

used after the previous code, would now place the value 21 on the stack, since $f(t) = 1 + 5t^2 = 21$, and the argument is now $t = 2$.

To summarize: `poly(x,m,n)` evaluates a polynomial of degree $m - 1$, using successive storage locations $n, n+1, n+2, \ldots, n+m-1$, i.e. the constant term is storage location n, and the coefficient of degree $m - 1$ is storage location $n + m - 1$. The argument is whatever value is on the top of the stack when `poly(x,m,n)` is invoked. This command takes three arguments x, m, n off the stack and replaces them by one value, so the stack is decreased by two elements. If there is an error in m or n, e.g., m or n negative, there is no error message, and the value $f(x) = 0$ is returned.

B.3.9.4 The command cheby(x,m,n)

The Chebyshev coefficients are first stored in locations n to $n + m - 1$, then the command `cheby(x,m,n)` will evaluate a Chebyshev expansion using the Broucke method with m terms. Note that the first term must be twice the constant term since, as usual, only half the constant is used in the expansion. This code, for instance, will return the Chebyshev approximation to $\exp(x)$.

```
2.532132
store(20)
1.130318
store(21)
0.271495
store(22)
0.044337
store(23)
0.005474
store(24)
20
5
x
cheby(x,m,n)
```

Note that, if the numerical values are placed on the stack sequentially, then they obviously must be peeled off in reverse order, as follows.

```
2.532132
1.130318
0.271495
0.044337
0.005474
store(24)
store(23)
store(22)
store(21)
store(20)
20
5
x
cheby(x,m,n)
```

To summarize: `cheby(x,m,n)` evaluates a Chebyshev approximation with m terms, using successive storage locations $n, n+1, n+2, \ldots, n+m+1$, i.e. twice the constant term is in storage location n, and the coefficient of $T(m-1)$ is in storage location $m + n - 1$. The argument is whatever value is on the top of the stack when cheby(x,m,n) is invoked. This command takes three arguments x, m, n off the stack and replaces them by one value, so the stack is decreased by two elements. If there is an error in x, m or n, e.g., x not in $(-1, 1)$, or m or n negative, there is no error message, and the value $f(x) = 0$ is returned. Use the test file `cheby.mod` with program **usermod** to appreciate this command.

B.3.9.5 The commands `l1norm(m,n)`, `l2norm(m,n)` and `linorm(m,n)`

The L_p norms are calculated for a vector with m terms, starting at storage location n, i.e. `l1norm` calculates the sum of the absolute values, `l2norm` calculates the Euclidean norm, while `linorm` calculates the infinity norm (that is, the largest absolute value in the vector).

It should be emphasized that `l2norm(m,n)` puts the Euclidean norm on the stack, that is the length of the vector (the square root of the sum of squares of the elements) and not the square of the distance. For example, the code

```
2
put(5)
-4
put(6)
3
put(7)
4
put(8)
1
put(9)
l1norm(3,5)
```

would place 9 on the stack, while the command `l2norm(5,5)` would put 6.78233 on the stack, and the command `linorm(5,5)` would return 4.

To summarize: these commands take two arguments off the stack and calculate either the sum of the absolute values, the square root of the sum of squares, or the largest absolute value in m successive storage locations starting at location n. The stack length is decreased by one since m and n are popped and replaced by the norm. There are no error messages and, if an error is encountered, a zero value is returned.

B.3.9.6 The commands `sum(m,n)` and `ssq(m,n)`

As there are occasions when it is useful to be able to add up the signed values or the squares of values in storage, these commands are provided. For instance, the code

```
1
2
3
4
put(103)
put(102)
put(101)
put(100)
100
4
sum(m,n)
f(1)
101
3
ssq(m,n)
f(2)
```

would assign 10 to function 1 and 29 to function 2.

To summarize: these commands take two arguments off the stack and then replace them with either the sum of m storage locations starting at position n, or the sum of squares of m storage locations starting at position n, so decreasing the stack length by 1.

B.3.9.7 The command `dotprod(l,m,n)`

This calculates the scalar product of two vectors of length l which are stored in successive locations starting at positions m and n.

To summarize: The stack length is decreased by 2, as three arguments are consumed, and the top stack element is then set equal to the dot product, unless an error is encountered when zero is returned.

B.3.9.8 Commands to use mathematical constants

The following commands are provided to facilitate model building.

Command	Value	Comment
pi	3.141592653589793e+00	pi
piby2	1.570796326794897e+00	pi divided by two
piby3	1.047197551196598e+00	pi divided by three
piby4	7.853981633974483e-01	pi divided by four
twopi	6.283185307179586e+00	pi multiplied by two
root2pi	2.506628274631000e+00	square root of two pi
deg2rad	1.745329251994330e-02	degrees to radians
rad2deg	5.729577951308232e+01	radians to degrees
root2	1.414213562373095e+00	square root of two
root3	1.732050807568877e+00	square root of three
eulerg	5.772156649015329e-01	Euler's gamma
lneulerg	-5.495393129816448e-01	log (Euler's gamma)

To summarize: these constants are merely added passively to the stack and do not affect any existing stack elements. To use the constants, the necessary further instructions would be required. So, for instance, to transform degrees into radial measure, the code

```
94.25
deg2rad
multiply
```

would replace the 94.25 degrees by the equivalent radians.

B.3.10 Integer functions

Sometimes integers are needed in models, for instance, as exponents, as summation indices, as logical flags, as limits in do loops, or as pointers in case constructs, etc. So there are special integer functions that take the top argument off the stack whatever number it is (say x) then replace it by an appropriate integer as follows.

Command	Description
int(x)	replace x by the integer part of x
nint(x)	replace x by the nearest integer to x
sign(x)	replace x by -1 if x < 0, by 0 if x = 0, or by 1 if x > 0

When using integers with SimFIT models it must be observed that only double precision floating point numbers are stored, and all calculations are done with such numbers, so that 0 actually means 0.0 to machine precision. So, for instance, when using these integer functions with real arguments to create logicals or indices for summation, etc. the numbers on the stack that are to be used as logicals or integers are actually transformed dynamically into integers when required at run time, using the equivalent of `nint(x)` to generate the appropriate integers. Because of this, you should note that code such as

```
...
11.3
19.7
1.2
int(x)
2.9
nint(x)
divide
```

would result in 1.0/3.0 being added to the stack (i.e. 0.3333 . . .) and not 1/3 (i.e 0) as it would for true integer division, leading to the stack

```
..., 11.3, 19.7, 0.3333333
```

B.3.11 Logical functions

Logical variables are stored in the global storage vector as either 1.0 (so that `nint(x) = 1 = true`) or as 0.0 (so that `nint(x) = 0 = false`). The logical functions either generate logical variables by testing the magnitude of the arbitrary stack value (say x) with respect to zero (to machine precision) or they accept only logical arguments (say m or n) and return an error message if the stack values are not pre-set to 0.0 or 1.0. Note that logical variables (i.e. Booleans) can be stored using `put(i)` and retrieved using `get(i)`, so that logical tests of any order of complexity can be constructed.

Command	Description
lt0(x)	replace x by 1 if x < 0, otherwise by 0
le0(x)	replace x by 1 if x =< 0, otherwise by 0
eq0(x)	replace x by 1 if x = 0, otherwise by 0
ge0(x)	replace x by 1 if x >= 0, otherwise by 0
gt0(x)	replace x by 1 if x > 0, otherwise by 0
not(m)	replace m by NOT(m), error if m not 0 or 1
and(m,n)	replace m and n by AND(m,n), error if m or n not 0 or 1
or(m,n)	replace m and n by OR(m,n), error if m or n not 0 or 1
xor(m,n)	replace m and n by XOR(m,n), error if m or n not 0 or 1

B.3.12 Conditional execution

Using these integer and logical functions in an appropriate sequence interspersed by `put(.)` and `get(.)` commands, any storage location (say j) can be set up to test whether any logical condition is true or false. So, the commands `if(.)` and `ifnot(.)` are provided to select model features depending on logical variables. The idea is to calculate the logical variables using the integer and logical functions, then load them into storage using `put(.)` commands. The `if(.)` and `ifnot(.)` commands inspect the designated storage locations and return 1 if the storage location has the value 1.0 (to machine precision), or 0 otherwise, even if the location is not 0.0 (to machine precision). The logical values returned are not added to the stack but, if a 1 is returned, the next line of the model code is executed whereas, if a 0 is returned, the next line is missed out.

```
Command    Description

if(j)      execute the next line only if storage(j) = 1.0
ifnot(j)   execute the next line only if storage(j) = 0.0
```

Note that very extensive logical tests and blocks of code for conditional executions, do loops, while and case constructs can be generated by using these logical functions sequentially but, because not all the lines of code will be active, the parsing routines will indicate the number of `if(.)` and `ifnot(.)` commands and the resulting potentially unused lines of code. This information is not important for correctly formatted models, but it can be used to check or debug code if required.

Consult the test file `if.mod` to see how to use logical functions.

B.3.13 Arbitrary functions with arbitrary arguments

The sub-models described so far for evaluation, integration, root finding, etc. are indexed at compile time and take dummy arguments, i.e. the ones supplied by the SimFIT calls to the model evaluation subroutines. However, sometimes it is useful to be able to evaluate a sub-model with arbitrary arguments added to the stack, or arguments that are functions of the main arguments. Again, it is useful to be able to evaluate an arbitrary function chosen dynamically from a set of sub-models indexed by an integer parameter calculated at run time, rather than read in at compile time when the model is first parsed. So, to extend the user-defined model syntax, the command `user1(x,m)` is provided. The way this works involves three steps:

1. an integer (m) is put on the stack to denote the required model,
2. calculations are performed to put the argument (x) on the stack, then
3. the user defined model is called and the result placed on the stack.

For instance the code

```
...
14.7
3
11.3
user1(x,m)
```

would result in

```
..., 14.7, 12.5
```

if the value of sub-model number 3 is 12.5 at an argument of 11.3.

Similar syntax is used for functions of two and three variables, i.e.

```
user1(x,m)
user2(x,y,m)
user3(x,y,z,m)
```

Clearly the integer m can be literal, calculated or retrieved from storage, but it must correspond to a sub-model that has been defined in the sequence of sub-models, and the calculated arbitrary arguments x, y, z must be sensible values. For instance the commands

```
2
x
user1(x,m)
```

and

```
value(2)
```

are equivalent. However the first form is more versatile, as the model number (m, 2 in this case) and argument (x, the dummy value in this case) can be altered dynamically as the result of stack calculations, while the second form will always invoke the case with $m = 2$ and $x =$ the subroutine argument to the model.

The model file `user1.mod` illustrates how to use the `user1(x,m)` command.

B.4 Examples using standard mathematical expressions

B.4.1 Test file usermod1_e.tf1: 1 function of 1 variable

```
begin{expression}
f(1) = p(1) +  p(2)x
end{expression}
```

B.4.2 Test file line3_e.mod: 3 functions of 1 variable

```
begin{expression}
f(1) = p(1) + p(2)x
f(2) = p(3) + p(4)x
f(3) = p(5) + p(6)x
end{expression}
```

B.4.3 Test file e04fyf_e.mod: 1 function of 3 variables

[ ] used for clarity and x, y, z for variables

```
begin{expression}
f(1) = p(1) + x/[p(2)y + p(3)z]
end{expression}
```

B.4.4 Test file d01fcf_e.mod: 1 function of 4 variables

[ ] used for clarity and y(i) for variables

```
begin{expression}
f(1) = 4y(1)y(3)^2[exp(2y(1)y(3))]/[1.0 + y(2) + y(4)]^2
end{expression}
```

B.4.5 Test file optimum_e.mod: 3 functions of 2 variables

Rosenbrock’s function and partial derivatives dummy variables A and B used to avoid repetition

```
begin{expression}
A = y - x^2
B = 1 - x
f(1) = 100*A^2 + B^2
f(2) = -400A*x - 2B
f(3) = 200A
end{expression}
```

B.4.6 Test file d01eaf_e.mod: 10 functions of 4 variables

Uses a * character for multiplication to be unambiguous

```
begin{expression}
A = y(1) + 2y(2) + 3y(3) + 4y(4)
B = log(A)
 f(1) = B*sin(1 + A)
 f(2) = B*sin(2 + A)
 f(3) = B*sin(3 + A)
 f(4) = B*sin(4 + A)
 f(5) = B*sin(5 + A)
 f(6) = B*sin(6 + A)
 f(7) = B*sin(7 + A)
 f(8) = B*sin(8 + A)
 f(9) = B*sin(9 + A)
f(10) = B*sin(10 + A)
end{expression}
```

B.4.7 Test file c05nbf_e.mod: 9 functions of 9 variables

```
begin{expression}
f(1) = (3 - 2y(1))y(1) + 1 - 2y(2)
f(2) = (3 - 2y(2))y(2) + 1 - y(1) - 2y(3)
f(3) = (3 - 2y(3))y(3) + 1 - y(2) - 2y(4)
f(4) = (3 - 2y(4))y(4) + 1 - y(3) - 2y(5)
f(5) = (3 - 2y(5))y(5) + 1 - y(4) - 2y(6)
f(6) = (3 - 2y(6))y(6) + 1 - y(5) - 2y(7)
f(7) = (3 - 2y(7))y(7) + 1 - y(6) - 2y(8)
f(8) = (3 - 2y(8))y(8) + 1 - y(7) - 2y(9)
f(9) = (3 - 2y(9))y(9) + 1 - y(8)
end{expression}
```

B.4.8 Test file deqmod2_e.tf2: 2 differential equations

Lotka-Volterra scheme with Jacobian

```
begin{expression}
f(1) = p(1)y(1) - p(2)y(1)y(2)
f(2) = -p(3)y(2) + p(4)y(1)y(2)
end{expression}
%
begin{expression}
j(1) = p(1) - p(2)y(2)
j(2) = p(4)y(2)
j(3) = -p(2)y(1)
j(4) = -p(3) + p(4)y(1)
end{expression}
```

B.5 Examples of user-defined models in reverse Polish notation

Examples will now be given in order to explain the format that must be adopted by users to define their own models for simulation, fitting and plotting. There are many more user-defined models distributed as test files with the SIMFIT package. The simpler method using standard mathematical expressions is described on page 453

B.5.1 Example 1: a straight line

This example illustrates how the test file `usermod1.tf1` codes for a simple straight line.

```
 %
Example: user supplied function of 1 variable ... a straight line
.............
p(1) + p(2)*x
.............
 %
 1 equation
 1 variable
 2 parameters
 %
 p(1)
 p(2)
 x
 multiply
 add
 f(1)
 %
```

Now exactly the same model but with comments added to explain what is going on. Note that in the model file, all text to the right of the instruction is treated as comment for the benefit of users and it is not referenced when the model is parsed.

```
 %                      start of text defining model indicated by %
Example: user supplied function of 1 variable ... a straight line
.............
p(1) + p(2)*x
.............
 %                      end of text, start of parameters indicated by %
 1 equation             number of equations to define the model
 1 variable             number of variables (or differential equation)
 2 parameters           number of parameters in the model
 %                      end of parameters, start of model indicated by %
 p(1)                   put p(1) on the stack: stack = p(1)
 p(2)                   put p(2) on the stack: stack = p(1), p(2)
 x                      put an x on the stack: stack = p(1), p(2), x
 multiply               multiply top elements: stack = p(1), p(2)*x
 add                    add the  top elements: stack = p(1) + p(2)*x
 f(1)                   evaluate the model f(1) = p(1) + p(2)*x
 %                      end of the model definitions indicated by %
```

B.5.2 Example 2: damped simple harmonic motion

This time test file `usermod1.tf9` illustrates trigonometric and exponential functions.

```
%
Example: user supplied function of 1 variable ... damped SHM
         Damped simple harmonic motion in the form
         f(x) = p(4)*exp[-p(3)*x]*cos[p(1)*x - p(2)]
         where p(i) >= 0
 %
 1 equation
 1 variable
 4 parameters
 %
 p(1)
 x
 multiply
 p(2)
 subtract
 cosine
 p(3)
 x
 multiply
 negative
 exponential
 multiply
 p(4)
 multiply
 f(1)
 %
```

B.5.3 Example 3: diffusion into a capillary

Test file `usermod1.tf8` codes for diffusion into a capillary and shows how to call special functions, in this case the error function complement with argument equal to distance divided by twice the square root of the product of the diffusion constant and time (i.e. p_2).

```
%
Example: user supplied function of 1 variable ... capillary diffusion

f(x) = p(1)*erfc[x/(2*sqrt(p(2))]

 %
 1 equation
 1 variable
 2 parameters
 %
 x
 p(2)
 squareroot
 2
 multiply
 divide
 erfc
 p(1)
 multiply
 f(1)
 %
```

B.5.4 Example 4: defining three models at the same time

The test file `line3.mod` illustrates the technique for defining several models for simultaneous fitting by program **qnfit**, in this case three straight lines unlinked for simplicity, although the models can be of arbitrary complexity and they can be linked by common parameters as illustrated later.

```
 %
f(1) = p(1) + p(2)x: (line 1)
f(2) = p(3) + p(4)x: (line 2)
f(3) = p(5) + p(6)x: (line 3)
Example: user supplied function of 1 variable ... 3 straight lines
 %
 3 equations
 1 variable
 6 parameters
 %
 p(1)
 p(2)
 x
 multiply
 add
 f(1)
 p(3)
 p(4)
 x
 multiply
 add
 f(2)
 p(5)
 p(6)
 x
 multiply
 add
 f(3)
 %
```

B.5.5 Example 5: Lotka-Volterra predator-prey differential equations

A special extended version of this format is needed with systems of differential equations, where the associated Jacobian can be supplied, as well as the differential equations if the equations are stiff and Gear's method is required. However, supplying the wrong Jacobian is a common source of error in differential equation solving, so you should always compare results with the option to calculate the Jacobian numerically, especially if slow convergence is suspected. A dummy Jacobian can be supplied if Gear's method is not to be used or if the Jacobian is to be estimated numerically. You can even prepare a differential equation file with no Jacobian at all.

So, to develop a model file for a system of differential equations, you first of all write the model ending with two lines, each containing only a %. When this runs properly you can start to add code for the Jacobian by adding new lines between the two % lines. This will be clear from inspecting the large number of model files provided and the `readme.*` files. If at any stage the code with Jacobian runs more slowly than the code without the Jacobian, then the Jacobian must be coded incorrectly.

The next example is the text for test file `deqmod2.tf2` which codes for the Lotka-Volterra predator-prey equations. This time all the comments are left in and a Jacobian is coded. This can be left out entirely by following the model by a percentage sign on two consecutive lines. SIMFIT can use the Adam's method but can still use Gears method by estimating the Jacobian by finite differences. Note that the Jacobian is initialized to the identity, so when supplying a Jacobian only the elements not equal to identity elements need be set.

```
%
Example of a user supplied pair of differential equations
file: deqmod2.tf2 (typical parameter file deqpar2.tf2)
model: Lotka-Volterra predator-prey equations

differential equations: f(1) = dy(1)/dx
                             = p(1)*y(1) - p(2)*y(1)*y(2)
                        f(2) = dy(2)/dx
                             = -p(3)*y(2) + p(4)*y(1)*y(2)

             jacobian: j(1) = df(1)/dy(1)
                            = p(1) - p(2)*y(2)
                       j(2) = df(2)/dy(1)
                            = p(4)*y(2)
                       j(3) = df(1)/dy(2)
                            = -p(2)*y(1)
                       j(4) = df(2)/dy(2)
                            = -p(3) + p(4)*y(1)

   initial condition: y0(1) = p(5), y0(2) = p(6)

Note: the last parameters must be y0(i) in differential equations
%
2 equations            no. equations
differential equation no. variables (or differential equation)
6 parameters           no. of parameters in this model
%
y(1)                   stack = y(1)
y(2)                   stack = y(1), y(2)
multiply               stack = y(1)*y(2)
duplicate              stack = y(1)*y(2), y(1)*y(2)
p(2)                   stack = y(1)*y(2), y(1)*y(2), p(2)
multiply               stack = y(1)*y(2), p(2)*y(1)*y(2)
negative               stack = y(1)*y(2), -p(2)*y(1)*y(2)
p(1)                   stack = y(1)*y(2), -p(2)*y(1)*y(2), p(1)
y(1)                   stack = y(1)*y(2), -p(2)*y(1)*y(2), p(1), y(1)
multiply               stack = y(1)*y(2), -p(2)*y(1)*y(2), p(1)*y(1)
add                    stack = y(1)*y(2), p(1)*y(1) - p(2)*y(1)*y(2)
f(1)                   evaluate dy(1)/dx
p(4)                   stack = y(1)*y(2), p(4)
multiply               stack = p(4)*y(1)*y(2)
p(3)                   stack = p(4)*y(1)*y(2), p(3)
y(2)                   stack = p(4)*y(1)*y(2), p(3), y(2)
multiply               stack = p(4)*y(1)*y(2), p(3)*y(2)
subtract               stack = -p(3)*y(2) + p(4)*y(1)*y(2)
f(2)                   evaluate dy(2)/dx
%                      end of model, start of Jacobian
p(1)                   stack = p(1)
p(2)                   stack = p(1), p(2)
y(2)                   stack = p(1), p(2), y(2)
```

```
multiply             stack = p(1), p(2)*y(2)
subtract             stack = p(1) - p(2)*y(2)
j(1)                 evaluate J(1,1)
p(4)
y(2)
multiply
j(2)                 evaluate J(2,1)
p(2)
y(1)
multiply
negative
j(3)                 evaluate J(1,2)
p(4)
y(1)
multiply
p(3)
subtract
j(4)                 evaluate J(2,2)
%
```

B.5.6 Example 6: supplying initial conditions

The test file `deqpar2.tf2` illustrates how initial conditions, starting estimates and limits are supplied.

```
Title line...(1) Parameter file for deqmod2.tf2 .. this line is ignored
0           (2) IRELAB: mixed(0), decimal places(1), sig. digits(2)
6           (3) M = number of parameters (include p(M-N+1)=y0(1), etc.)
1           (4) METHOD: Gear(1), Runge_Kutta(2), Adams(3)
1           (5) MPED: Jacobian estimated(0), calculated(1)
2           (6) N = number of equations
41          (7) NPTS = number of time points
0.0,1.0,3.0                      (7+1) pl(1),p(1),ph(1) parameter 1
0.0,1.0,3.0                      (7+2) pl(2),p(2),ph(2) parameter 2
0.0,1.0,3.0                      (7+3) pl(3),p(3),ph(3) parameter 3
0.0,1.0,3.0                      (7+4) pl(4),p(4),ph(4) parameter 4
0.0,1.0,3.0                      (7+5) pl(5),p(5),ph(5) y0(1)
0.0,0.5,3.0                      (7+M) pl(6),p(6),ph(6) y0(2)
1.0e-4      (7+M+1) TOL: tolerance
10.0        (7+M+2) XEND: end of integration
0.0         (7+M+3) XSTART: start of integration
```

An initial conditions file supplies all the values required for a simulation or curve fitting problem with differential equations using programs **deqsol** or **qnfit**. Note that the values must be supplied in exactly the above order. The first line (title) and trailing lines after (7+M+3) are ignored. Field width for most values is 12 columns, but is 36 for parameters. Comments can be added after the last significant column if required. Parameters are in the order of $pl(i) \le p(i) \le ph(i)$ where $pl(i)$ are the bottom limits, $p(i)$ are the starting parameters and $ph(i)$ are the upper limits for curve fitting. To fix a parameter during curve fitting just set $pl(i) = p(i) = ph(i)$. Note that $pl(i)$ and $ph(i)$ are used in curve fitting but not simulation. Parameters 1 to $M - N$ are the parameters in the equations, but parameters $M - N + 1$ to M are the initial conditions, namely $y_0(1)$ to $y_0(N)$.

B.5.7 Example 7: transforming differential equations

If you just want information on a sub-set of the components, $y(i)$, you can select any required components (interactively) in **deqsol**. If you only want to fit a sub-set of components, this is done by adding escape sequences to the input data library file as shown by the % characters in the example files `deqsol.tf2` and `deqsol.tf3`. A more complicated process is required if you are interested only in some linear combination of the $y(i)$, and do not want to (or can not) re-write the differential equations into an appropriate form, even using conservation equations to eliminate variables. To solve this problem you can input a matrix A, then simply choose $y(\text{new}) = A * y(\text{old})$, where after integration $y(\text{new})$ replaces $y(\text{old})$.

Format for A-type files

The procedure is that when transformation is selected, deqsol sets A equal to the identity matrix then it reads in your file with the sub-matrix to overwrite A. The A-file simply contains a column of i-values,a column of j-values and a column of corresponding $A(i, j)$ values. To prepare such a file you can use makmat or a text editor. Consult the test files (`deqmat.tf?`) for examples.

Examples
An A matrix to interchange $y(1)$ and $y(2)$.

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

An A matrix to replace $y(2)$ by $y(1) + y(2)$.

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

An A matrix to replace $y(2)$ by $0.5y(1) + 2.0y(2) - y(3)$ then swap $y(1)$ and $y(3)$.

$$\begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.5 & 2.0 & -1.0 \\ 1.0 & 0.0 & 0.0 \end{pmatrix}$$

Note the following facts.

1. You only need to supply elements $A(i, j)$ which differ from those of the corresponding identity matrix.
2. The program solves for the actual $y(i)$ then makes new vectors $z(i)$ where $y(i)$ are to be transformed. The $z(i)$ are then copied onto the new $y(i)$.
3. *This is very important.* To solve the $y(i)$ the program has to start with the actual initial conditions $y_0(i)$. So, even if the $y(i)$ are transformed by an A which is not the identity, the y_0 are never transformed. When simulating you must remember that $y_0(i)$ you set are true $y_0(i)$ and when curve-fitting, parameters estimated are the actual $y_0(i)$, *not* transformed ones.

B.5.8 Example 8: consecutive irreversible chemical reactions

This example, from test file `consec3.mod`, requires two sub-models to be defined, because the standard formula for the second species becomes singular when the two rate constants are equal and a gamma function type solution is required.

```
%
Irreversible chemical kinetics A --> B --> C
with A(0) > 0, B(0) = 0, C(0) = 0
p(1) = k(1), p(2) = k(2), p(3) = A(0)
  A(x) = A(0)exp(-k(1)x)
If p(1) does not equal p(2) then sub-model 1 evaluates
  B = p(1)p(3)[exp(-k(1)x) - exp(-k(2)x)]/[(p(2) - p(1)]
otherwise if k(1) equals k(2) then sub-model 2 evaluates
  B = xA(x)
```

```
 C = A(0) - B(x) - C(x)
%
3 equations
1 variable
3 parameters
%
putpar             communicate parameters to sub-models
p(1)
x
multiply
negative
exp
p(3)
multiply
put(1)              storage(1) = A(x)
p(2)
p(1)
subtract
value3(1,2,1)       B(x) depends on p(2) - p(1)
put(2)              storage(2) = B(x)
p(3)
get(1)
subtract
get(2)
subtract
f(3)
get(1)
f(1)
get(2)
f(2)
%
begin{model(1)}
%
sub-model 1
B in A --> B --> C where p(1) is not equal to p(2)
%
1 equation
1 variable
3 parameters
%
p(1)
x
multiply
negative
exp
p(2)
x
multiply
negative
exp
subtract
p(1)
multiply
p(3)
```

```
 multiply
 p(2)
 p(1)
 subtract
 divide
 f(1)
 %
 end{model(1)}
 begin{model(2)}
 %
 sub-model 2
 B in A --> B --> C where p(1) is equal to p(2)
 %
 1 equation
 1 variable
 3 parameters
 %
 get(1)
 x
 multiply
 f(1)
 %
 end{model(2)}
```

B.5.9 Example 9: evaluating a convolution integral

This model is in test file convolv3.mod and requires the two independent (or parameter-linked) models involved in the convolution to be defined as attached sub-models.

```
 %
An example using two sub-models for a convolution integral f*g
=================================================================
This demonstrates how to define 2 equations as sub-models, using
the command putpar to communicate parameters to the sub-models,
and the command convolute(1,2) to integrate sub-models 1 and 2
(by adaptive quadrature) from blim(1) = 0 to t = tlim(1) = x.
Precision of D01AJF quadrature is controlled by epsabs and epsrel
and blim(1) and tlim(1) must be used for the convolution limits
which, in this case are 0 to x, where x > 0 by assumption.
.................................................................
convolution integral: from 0 to x of f(u)*g(x - u) du, where
f1(t) = f(t) = exp(-p(1)*t)
f2(t) = g(t) = [p(2)^2]*t*exp(-p(2)*t)
f3(t) = f*g  = f1*f2
.................................................................
Note that usually extra parameters must be supplied if it wished
to normalise so that the integral of f or g or f*g is specified
(e.g.,equals 1) over the total range of possible integration.
This must often be done, e.g., if g(.) is a density function.
The gamma distribution normalising  factor p(2)**2 is stored in
this example to avoid unnecessary re-calculation.
 %
 3 equations
 1 variable
 2 parameters
 %
 putpar
 p(2)
 p(2)
 multiply
 put(1)
 1
 x
 user1(x,m)
 f(1)
 2
 x
 user1(x,m)
 f(2)
 0.0001
 epsabs
 0.001
 epsrel
 0
 blim(1)
 x
 tlim(1)
 convolute(1,2)
 f(3)
 %
begin{model(1)}
 %
Example: exponential decay, exp(-p(1)*x)
 %
 1 equation
 1 variable
 1 parameter
 %
 p(1)
 x
 multiply
 negative
 exponential
 f(1)
 %
end{model(1)}
begin{model(2)}
 %
Example: gamma density of order 2
 %
 1 equation
 1 variable
 2 parameters
 %
 p(2)
 x
 multiply
 negative
 exponential
 x
 multiply
 get(1)
 multiply
 f(1)
 %
end{model(2)}
```

The command `putpar` communicates the parameters from the main model to the sub-models, the quadrature precision is controlled by `epsabs` and `epsrel` and, irrespective of which models are used in the convolution, the limits of integration are always input using the `blim(1)` and `tlim(1)` commands just before using the `convolute(.,.)` command. Often the response function has to be normalized, usually to integrate to 1 over the overall range, and the prior squaring of $p(1)$ to use as a normalizing factor for the gamma density in this case is done to save multiplication each time model(2) is called by the quadrature routine. Such models are often used for deconvolution by curve fitting in situations where the sub-models are known, but unknown parameters have to be estimated from output data with associated error, and this technique should not be confuse with graphical deconvolution described on page 38.

Appendix C

Library of models

C.1 Mathematical models [Library: Version 2.0]

The SIMFIT libraries are used to generate exact data using program **makdat**, or to fit data using an advanced curve fitting program, e.g. **qnfit**. Version 2.0 of the SIMFIT library only contains a limited selection of models, but there are other versions available, with extra features and model equations. The models are protected to prevent overflow during function evaluation, and they return zero when called with meaningless arguments, e.g. the beta pdf with negative parameters or independent variable outside $[0, 1]$. After a model has been selected, it is initialized, and the parameters are described in more meaningful form, e.g. as rate constants or initial concentrations, etc. Also, any restrictions on the parameter values or range of independent variable are listed, and equations for singular cases such as $1/(p_2 - p_1)$ when $p_2 = p_1$ are given.

C.2 Functions of one variable

C.2.1 Differential equations

These can be integrated using the BDF method as an explicitly calculated Jacobian is supplied from the library.

1. Irreversible Michaelis-Menten substrate depletion.

$$\frac{dy}{dx} = \frac{-p_2 y}{p_1 + y}; y = S, S(0) = p_3, P(0) = 0, x = \text{time}.$$

2. Irreversible Michaelis-Menten product accumulation.

$$\frac{dy}{dx} = \frac{p_2(p_3 - y)}{p_1 + (p_3 - y)}; y = P, S(0) = p_3, P(0) = 0, x = \text{time}.$$

3. Generalized substrate depletion with a non-specific diffusion term.

$$\frac{dy}{dx} = \frac{-p_2 y}{p_1 + y} - p_3 y - p_4; y = S, S(0) = p_5, P(0) = 0, x = \text{time}.$$

4. General product accumulation with a non-specific diffusion term.

$$\frac{dy}{dx} = \frac{p_2(p_5 - y)}{p_1 + (p_5 - y)} + p_3(p_5 - y) + p_4; y = P, S(0) = p_5, P(0) = 0, x = \text{time}.$$

5. Membrane transport allowing for osmotic volume changes.

$$\frac{dy}{dx} = \frac{p_3(y - p_4)}{y^2 + p_1 y + p_2}; y = S, y(\infty) = p_4, y(0) = p_5, x = \text{time}.$$

This model has many applications in transport studies, e.g. S is the mass (not concentration) of glucose during efflux from loaded erythrocytes where solute moves to maintain osmotic equilibrium, so that volumes change.

6. Von Bertalanffy allometric growth.

$$\frac{dy}{dx} = p_1 y^{p_2} - p_3 y^{p_4};\, y = \text{ size}, y(0) = p_5, x = \text{time}.$$

This is a difficult model to fit and for most purposes **gcfit** can be used for the various cases where formal integration is possible. This model does not allow turning points.

7. Von Bertalanffy allometric growth and decay.

$$\frac{dy}{dx} = \exp(-p_5 x) p_1 y^{p_2} - p_3 y^{p_4};\, y = \text{size}, y(0) = p_6, x = \text{time}.$$

This model is extremely difficult to fit and, in order to get meaningful parameter estimates, it is usually necessary to fix the exponents p_2 and p_4. However, it does have the advantage of being able to model growth curves with a turning point, as occurs with isolated bacterial populations and fixed nutrient supply.

C.2.2 Systems of differential equations

The library has a selection of systems of 1, 2, 3, 4 and 5 differential equations which can be used for simulation and fitting by program **deqsol**. Also ASCII coordinate files called `deqmod?.tf?` and `deqpar?.tf?` are provided for the same models, to illustrate how to supply your own differential equations. Program **deqsol** can use the Adams methods and allows the use of Gear's method with an explicit Jacobian or else an internally approximated one.

C.2.3 Special models

Polynomial of degree n: $p_{n+1} + p_1 x + p_2 x^2 + \cdots + p_n x^n$

Order $n : n$ rational function: $\dfrac{p_{2n+1} + p_1 x + p_2 x^2 + \cdots + p_n x^n}{1 + p_{n+1} x + p_{n+2} x^2 + \cdots + p_{2n} x^n}$

Multi Michaelis-Menten functions: $\dfrac{p_1 x}{p_{n+1} + x} + \dfrac{p_2 x}{p_{n+2} + x} + \cdots + \dfrac{p_n x}{p_{2n} + x} + p_{2n+1}$

Multi M-M in isotope displacement mode, with y =[Hot]:

$$\frac{p_1 y}{p_{n+1} + y + x} + \frac{p_2 y}{p_{n+2} + y + x} + \cdots + \frac{p_n y}{p_{2n} + y + x} + p_{2n+1}$$

Multi M-M in isotope displacement mode, with [Hot] subsumed:

$$\frac{p_1}{p_{n+1} + x} + \frac{p_2}{p_{n+2} + x} + \cdots + \frac{p_n}{p_{2n} + x} + p_{2n+1}$$

High/Low affinity sites: $\dfrac{p_1 p_{n+1} x}{1 + p_{n+1} x} + \dfrac{p_2 p_{n+2} x}{1 + p_{n+2} x} + \cdots + \dfrac{p_n p_{2n} x}{1 + p_{2n} x} + p_{2n+1}$

H/L affinity sites in isotope displacement mode, with y = [Hot]:

$$\frac{p_1 p_{n+1} y}{1 + p_{n+1}(x + y)} + \frac{p_2 p_{n+2} y}{1 + p_{n+2}(x + y)} + \cdots + \frac{p_n p_{2n} y}{1 + p_{2n}(x + y)} + p_{2n+1}$$

H/L affinity sites in isotope displacement mode, with [Hot] subsumed:

$$\frac{p_1 p_{n+1}}{1 + p_{n+1}x} + \frac{p_2 p_{n+2}}{1 + p_{n+2}x} + \cdots + \frac{p_n p_{2n}}{1 + p_{2n}x} + p_{2n+1}$$

Binding constants saturation function: $\frac{p_{n+1}}{n}\left\{\frac{p_1 x + 2p_2 x^2 + \cdots + np_n x^n}{1 + p_1 x + p_2 x^2 + \cdots + p_n x^n}\right\} + p_{n+2}$

Binding constants in isotope displacement mode, with y = [Hot]:

$$\frac{p_{n+1}y}{n}\left\{\frac{p_1 + 2p_2(x+y) + \cdots + np_n(x+y)^{n-1}}{1 + p_1(x+y) + p_2(x+y)^2 + \cdots + p_n(x+y)^n}\right\} + p_{n+2}$$

Adair constants saturation function: $\frac{p_{n+1}}{n}\left\{\frac{p_1 x + 2p_1 p_2 x^2 + \cdots + np_1 p_2 \ldots p_n x^n}{1 + p_1 x + p_1 p_2 x^2 + \cdots + p_1 p_2 \ldots p_n x^n}\right\} + p_{n+2}$

Adair constants in isotope displacement mode , with y = [Hot]:

$$\frac{p_{n+1}y}{n}\left\{\frac{p_1 + 2p_1 p_2(x+y) + \cdots + np_1 p_2 \ldots p_n(x+y)^{n-1}}{1 + p_1(x+y) + p_1 p_2(x+y)^2 + \cdots + p_1 p_2 \ldots p_n(x+y)^n}\right\} + p_{n+2}$$

Sum of n exponentials: $p_1 \exp(-p_{n+1}x) + p_2 \exp(-p_{n+2}x) + \cdots + p_n \exp(-p_{2n}x) + p_{2n+1}$

Sum of n functions of the form $1 - \exp(-kx)$:

$$p_1\{1 - \exp(-p_{n+1}x)\} + p_2\{1 - \exp(-p_{n+2}x)\} + \cdots + p_n\{1 - \exp(-p_{2n}x)\} + p_{2n+1}$$

Sum of n sine functions: $\sum_{i=1}^{n} p_i \sin(p_{n+i}x + p_{2n+i}) + p_{3n+1}$

Sum of n cosine functions: $\sum_{i=1}^{n} p_i \cos(p_{n+i}x + p_{2n+i}) + p_{3n+1}$

Sum of n Gauss (Normal) pdf functions: $\frac{p_1}{p_{2n+1}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left\{\frac{x - p_{n+1}}{p_{2n+1}}\right\}^2\right) +$

$$\frac{p_2}{p_{2n+2}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left\{\frac{x - p_{n+2}}{p_{2n+2}}\right\}^2\right) + \cdots + \frac{p_n}{p_{3n}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left\{\frac{x - p_{2n}}{p_{3n}}\right\}^2\right) + p_{3n+1}$$

Sum of n Gauss (Normal) cdf functions: $\frac{p_1}{p_{2n+1}\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{1}{2}\left\{\frac{u - p_{n+1}}{p_{2n+1}}\right\}^2\right) du +$

$$\frac{p_2}{p_{2n+2}\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{1}{2}\left\{\frac{u - p_{n+2}}{p_{2n+2}}\right\}^2\right) du + \cdots +$$

$$\frac{p_n}{p_{3n}\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{1}{2}\left\{\frac{u - p_{2n}}{p_{3n}}\right\}^2\right) du + p_{3n+1}$$

C.2.4 Biological models

Exponential growth/decay in three parameterizations:

Parameterization 1: $p_1 \exp(-p_2 x) + p_3$

Parameterization 2: $\exp(p_1 - p_2 x) + p_3$

Parameterization 3: $\{p_1 - p_3\} \exp(-p_2 x) + p_3$

Monomolecular growth: $p_1\{1 - \exp(-p_2 x)\} + p_3$

Logistic growth in four parameterizations:

Parameterization1: $\dfrac{p_1}{1 + p_2 \exp(-p_3 x)} + p_4$

Parameterization 2: $\dfrac{p_1}{1 + \exp(p_2 - p_3 x)} + p_4$

Parameterization 3: $\dfrac{p_1}{1 + \exp(-[p_3(x - p_2)])} + p_4$

Parameterization 4: $\dfrac{p_1}{1 + \exp(-[p_2 + p_3 x])} + p_4$

Gompertz growth: $p_1 \exp\{-p_2 \exp(-p_3 x)\} + p_4$

Richards growth: $\left\{p_1^{(1-p_4)} - p_2 \exp(-p_3 x)\right\}^{\left(\frac{1}{1-p_4}\right)} + p_5$

Preece and Baines: $p_4 - \dfrac{2(p_4 - p_5)}{\exp[p_1(x - p_3)] + \exp[p_2(x - p_3)]}$

Weibull survival in four parameterizations:

Parameterization 1: $p_1 \exp\left(-p_2[x^{p_3}]\right) + p_4$

Parameterization 2: $p_1 \exp\left(-[x^{p_3}]/p_2\right) + p_4$

Parameterization 3: $p_1 \exp\left(-[p_2 x]^{p_3}\right) + p_4$

Parameterization 4: $p_1 \exp\left(-\exp(p_2)[x^{p_3}]\right) + p_4$

Gompertz survival: $p_1 \exp\left\{-\left(\dfrac{p_2}{p_3}\right)[\exp(p_3 x) - 1]\right\} + p_4$

C.2.5 Biochemical models

Monod-Wyman-Changeux allosterism: $K = p_1, c = p_2 < 1, L = p_3, V = p_4$

$$p_1 p_4 x \left\{\frac{(1 + p_1 x)^{n-1} + p_2 p_3 (1 + p_1 p_2 x)^{n-1}}{(1 + p_1 x)^n + p_3 (1 + p_1 p_2 x)^n}\right\}$$

Lag phase to steady state: $p_1 x + p_2\{1 - \exp(-p_3 x)\} + p_4$

One-site binding: x = [Total ligand], $K_d = p_1$, [Total sites] $= p_2$,

$$\frac{A - \sqrt{A^2 - 4p_2 x}}{2p_2}; \text{ where } A = x + p_1 + p_2$$

Irreversible Michaelis Menten progress curve: $K_m = p_1, V_{max} = p_2, [S(x = 0)] = p_3$

$$p_1 \ln\left|\frac{p_3}{p_3 - f(x)}\right| + f(x) - p_2 x = 0, \text{ where } f(0) = 0$$

Or, in a more familiar parameterization:

$$K_m \ln\left|\frac{S(0)}{S(0) - P(t)}\right| + P(t) - V_{max} t = 0$$

Irreversible Michaelis Menten deplettion curve: $K_m = p_1, V_{max} = p_2, [S(x = 0)] = p_3$

$p_1 \ln \left| \dfrac{f(x)}{p_3)} \right| + f(x) - p_3 + p_2 x = 0,$ where $f(0) = 0$

Or, in a more familiar parameterization:

$K_m \ln \left| \dfrac{S(t)}{S(0)} \right| + S(t) - S(0) + V_{max} t = 0$

Michaelis-Menten plus diffusion in three modes with x = [S] or [Cold], and y = [Hot]:

Type 1: $\dfrac{p_1 x}{p_2 + x} + p_3 x$, [No hot], $p_1 = V_{max}, p_2 = K_m, p_3 = D$

Type 2: $\dfrac{p_1 y}{p_2 + y + x} + p_3 y$, [Hot input], $p_1 = V_{max}, p_2 = K_m, p_3 = D$

Type 3: $\dfrac{p_1}{p_2 + x} + p_3$, [Hot subsumed], $p_1 = V_{max} y, p_2 = K_m + y, p_3 = Dy$

Generalized inhibition: $\dfrac{p_1}{p_2 + x}$

C.2.6 Chemical models

Arrhenius rate constant law: $p_1 \exp(-p_2/x)$

Transition state rate constant law: $p_1 x^{p_3} \exp(-p_2/x)$

B in A $\rightarrow$ B $\rightarrow$ C: $\left\{ \dfrac{p_1 p_3}{p_2 - p_1} \right\} \exp(-p_1 x) + \left\{ p_4 - \dfrac{p_1 p_3}{p_2 - p_1} \right\} \exp(-p_2 x)$

C in A $\rightarrow$ B $\rightarrow$ C: $p_3 + p_4 + p_5 - \left\{ \dfrac{p_2 p_3}{p_2 - p_1} \right\} \exp(-p_1 x) - \left\{ p_4 - \dfrac{p_1 p_3}{p_2 - p_1} \right\} \exp(-p_2 x)$

B in A $\rightleftharpoons$ B reversibly: $\dfrac{p_1(p_3 + p_4) + (p_2 p_4 - p_1 p_3) \exp\{-(p_1 + p_2)x\}}{p_1 + p_2}$

Michaelis pH functions with scaling and displacement factors:

$p_3[1 + p_1/x + p_1 p_2/x^2] + p_4$

$p_3[1 + x/p_1 + p_2/x] + p_4$

$p_3[1 + x/p_2 + x^2/(p_1 p_2)] + p_4$

Freundlich isotherm: $p_1 x^{1/p_2} + p_3$

C.2.7 Physical models

Diffusion into a capillary: $p_1 \operatorname{erfc} \left\{ \dfrac{x}{2\sqrt{p_2}} \right\}$, where $p_2 = Dt$

Full Mualen equation: $\left\{ \dfrac{1}{1 + (p_1 x)^{p_2}} \right\}^{p_3}$

Short Mualen equation: $\left\{ \dfrac{1}{1 + (p_1 x)^{p_2}} \right\}^{(1-1/n)}$

Brittle-Ductile-Transition equation: $p_1 + p_2 \tanh \left(\dfrac{x - p_4}{p_3} \right)$

Pabst-Einstein equation and log transforms: $p_3 \left(1 - \dfrac{x}{p_1} \right)^{-p_1 p_2}$

C.2.8 Statistical models

Normal pdf: $\dfrac{p_3}{p_2\sqrt{2\pi}}\exp\left(-\dfrac{1}{2}\left\{\dfrac{x-p_1}{p_2}\right\}^2\right)$

Beta pdf: $\dfrac{p_3\Gamma(p_1+p_2)}{\Gamma(p_1)\Gamma(p_2)}x^{p_1-1}(1-x)^{p_2-1}$

Exponential pdf: $p_1 p_2\exp(-p_1 x)$

Cauchy pdf: $\dfrac{p_3}{\pi p_2\left\{1+[(x-p_1)/p_2)]^2\right\}}$

Logistic pdf: $\dfrac{p_3\exp[(x-p_1)/p_2]}{p_2\{1+\exp[(x-p_1)/p_2]\}^2}$

Lognormal pdf: $\dfrac{p_3}{p_2 x\sqrt{2\pi}}\exp\left(-\dfrac{1}{2}\left\{\dfrac{\ln x-p_1}{p_2}\right\}^2\right)$

Gamma pdf: $\dfrac{p_3 p_1^{p_2} x^{p_2-1}\exp(-p_1 x)}{\Gamma(p_2)}$

Rayleigh pdf: $\left\{\dfrac{p_2 x}{p_1^2}\right\}\exp\left(-\dfrac{1}{2}\left\{\dfrac{x}{p_1}\right\}^2\right)$

Maxwell pdf: $\left\{\dfrac{2p_2x^2}{p_1^3\sqrt{2\pi}}\right\}\exp\left(-\dfrac{1}{2}\left\{\dfrac{x}{p_1}\right\}^2\right)$

Weibull pdf: $\left\{\dfrac{p_1 p_3 x^{p_1-1}}{p_2}\right\}\exp\left(\dfrac{-x^{p_1}}{p_2}\right)$

Normal cdf, i.e. integral from $-\infty$ to x of the normal pdf defined above

Beta cdf, i.e. integral from 0 to x of the beta pdf defined above

Exponential cdf: $p_2\{1-\exp(-p_1 x)\}$

Cauchy cdf: $p_3\left\{\dfrac{1}{2}+\dfrac{1}{\pi}\arctan\left(\dfrac{x-p_1}{p_2}\right)\right\}$

Logistic cdf: $\dfrac{p_3\exp\{(x-p_1)/p_2\}}{1+\exp\{(x-p_1)/p_2\}}$

Lognormal cdf, i.e. integral from 0 to x of Lognormal pdf

Weibull cdf: $p_3\left\{1-\exp\left(\dfrac{-x^{p_1}}{p_2}\right)\right\}$

Logit in exponential format: $\dfrac{1}{1+\exp[-\{p_1+p_2x\}]}$

Probit in Normal cdf format: $\Phi(p_1+p_2x)$

Sum of 2 normal pdfs: $\dfrac{p_3}{p_2\sqrt{2\pi}}\exp\left(-\dfrac{1}{2}\left\{\dfrac{x-p_1}{p_2}\right\}^2\right)+\dfrac{1-p_3}{p_5\sqrt{2\pi}}\exp\left(-\dfrac{1}{2}\left\{\dfrac{x-p_4}{p_5}\right\}^2\right)$

Sum of 2 normal cdfs, i.e. integral from $-\infty$ to x of sum of 2 normal pdfs defined above

C.2.9 Empirical models

Hill with n fixed: $\dfrac{p_1 x^n}{p_2^n + x^n} + p_3$

Hill with n varied: $\dfrac{p_1 x^{p_3}}{p_2^{p_3} + x^{p_3}} + p_4$

Power law: $p_1 x^{p_2} + p_3$

$\log_{10}$ law: $p_1 \log_{10} x + p_2$

Up/Down exponential: $p_3\{\exp(-p_1 x) - \exp(-p_2 x)\} + p_4$

Up/Down logistic: $\dfrac{p_1}{1 + \exp(p_2 - p_3 x) + \exp(p_4 + p_5 x)} + p_6$

Double exponential plus quadratic: $p_1 \exp(-p_2 x) + p_3 \exp(-p_4 x) + p_5 x^2 + p_6 x + p_7$

Double logistic: $\dfrac{p_1}{1 + \exp(p_2 - p_3 x)} + \dfrac{p_4}{1 + \exp(p_5 - p_6 x)} + p_7$

Linear plus reciprocal: $p_1 x + p_2/x + p_3$

Gaussian plus exponential: $\dfrac{p_3}{p_2\sqrt{2\pi}} \exp\left(-\dfrac{1}{2}\left\{\dfrac{x - p_1}{p_2}\right\}^2\right) + p_5 \exp(-p_4 x) + p_6$

Gaussian times exponential: $\dfrac{p_3}{p_2\sqrt{2\pi}} \exp\left(-\dfrac{1}{2}\left\{\dfrac{x - p_1}{p_2}\right\}^2\right) \exp(-p_4 x) + p_5$

C.2.10 Mathematical models

Upper or lower semicircle: $p_2 \pm \sqrt{p_3^2 - (x - p_1)^2}$

Upper or lower semiellipse: $p_2 \pm p_4\sqrt{1 - \left(\dfrac{x - p_1}{p_3}\right)^2}$

Sine/Cosine: $p_1 \sin(p_3 x) + p_2 \cos(p_3 x) + p_4$

Damped SHM: $\exp(-p_4 x)[p_1 \sin(p_3 x) + p_2 \cos(p_3 x)] + p_5$

Arctangent: $p_1 \arctan(p_2 x) + p_3$

Gamma type: $p_1 x^{p_2} \exp(-p_3 x) + p_4$

Sinh/Cosh: $p_1 \sinh(p_3 x) + p_2 \cosh(p_3 x) + p_4$

Tanh: $p_1 \tanh(p_2 x) + p_3$

C.3 Functions of two variables

C.3.1 Polynomials

Degree 1: $p_1 x + p_2 y + p_3$

Degree 2: $p_1 x + p_2 y + p_3 x^2 + p_4 xy + p_5 y^2 + p_6$

Degree 3: $p_1 x + p_2 y + p_3 x^2 + p_4 xy + p_5 y^2 + p_6 x^3 + p_7 x^2 y + p_8 x y^2 + p_9 y^3 + p_{10}$

C.3.2 Rational functions:

2 : 2 with $f(0,0) = 0$: $\dfrac{p_1xy}{1 + p_2x + p_3y + p_4x^2 + p_5xy + p_6y^2}$

3 : 3 with $f(0,0) = 0$: $\dfrac{p_1xy + p_2x^2y + p_3xy^2}{1 + p_4x + p_5y + p_6x^2 + p_7xy + p_8y^2 + p_9x^3 + p_{10}x^2y + p_{11}xy^2 + p_{12}y^3}$

1 : 1 rational function: $\dfrac{p_5 + p_1x + p_2y}{1 + p_3x + p_4y}$

2 : 2 rational function: $\dfrac{p_{11} + p_1x + p_2y + p_3x^2 + p_4xy + p_5y^2}{1 + p_6x + p_7y + p_8x^2 + p_9xy + p_{10}y^2}$

C.3.3 Enzyme kinetics

Reversible Michaelis Menten (product inhibition): $\dfrac{p_1x/p_2 - p_3y/p_4}{1 + x/p_2 + y/p_4}$

Competitive inhibition: $\dfrac{p_1x}{p_2(1 + y/p_3) + x}$

Uncompetitive inhibition: $\dfrac{p_1x}{p_2 + x(1 + y/p_3)}$

Noncompetitive inhibition: $\dfrac{p_1x}{(1 + y/p_3)(p_2 + x)}$

Mixed inhibition: $\dfrac{p_1x}{p_2(1 + y/p_3) + x(1 + y/p_4)}$

Ping pong bi bi: $\dfrac{p_1xy}{p_3x + p_2y + xy}$

Ordered bi bi: $\dfrac{p_1xy}{p_3p_4 + p_3x + p_2y + xy}$

Time dependent inhibition: $p_1 \exp - \left\{ \dfrac{p_2x}{1 + p_3/y} \right\}$

Inhibition by competing substrate: $\dfrac{p_1x/p_2}{1 + x/p_2 + y/p_3}$

Michaelis-Menten pH dependence: $f(y)x/[g(y) + x]$

$f(y) = p_1/[1 + y/p_5 + p_6y], \quad g(y) = p_2(1 + y/p_3 + p_4y)/[1 + y/p_5 + p_6y]$

C.3.4 Biological

Logistic growth: $\dfrac{p_1}{1 + \exp(-[p_2 + p_3x + p_4y + p_5xy])} + p_6$

C.3.5 Physical

Diffusion into a capillary: $p_1 \operatorname{erfc} \left\{ \dfrac{x}{2\sqrt{p_2y}} \right\}$

C.3.6 Statistical

Bivariate normal pdf: $p_1 = \mu_x, p_2 = \sigma_x, p_3 = \mu_y, p_4 = \sigma_y, p_5 = \rho$

$$\frac{p_6}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left\{\frac{-1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right) + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]\right\} + p_7$$

Logit in exponential format: $\dfrac{1}{1+\exp[-\{p_1 + p_2x + p_3y\}]}$

Probit in Normal cdf format: $\Phi(p_1 + p_2x + p_3y)$

C.4 Functions of three variables

C.4.1 Polynomials

Linear: $p_1x + p_2y + p_3z + p_4$

C.4.2 Enzyme kinetics

MWC activator/inhibitor: $\dfrac{nV\alpha(1+\alpha)^{n-1}}{(1+\alpha)^n + L[(1+\beta)/(1+\gamma)]^n}$

$\alpha = p_1[x], \beta = p_2[y], \gamma = p_3[z], V = p_4, L = p_5$

C.4.3 Biological

Logistic growth: $\dfrac{p_1}{1+\exp(-[p_2 + p_3x + p_4y + p_5z + p_6xy + p_7xz + p_8yz + p_9xyz])} + p_{10}$

C.4.4 Statistics

Logit in exponential format: $\dfrac{1}{1+\exp[-\{p_1 + p_2x + p_3y + p_4z\}]}$

Probit in Normal cdf format: $\Phi(p_1 + p_2x + p_3y + p_4z)$

Appendix D

Auxiliary programs

D.1 Recommended software

SimFIT can be used as a self-contained free-standing package. However, it is assumed that users will want to integrate SimFIT with other software, and the driver `w_simfit.exe` has been constructed with the Windows calculator, the Windows Notepad text editor, the GSview PostScript interpreter, and the Adobe Acrobat pdf reader as defaults. Users can, of course, easily replace these by their own choices, by using the Configuration option on the main menu. The clipboard can be used to integrate SimFIT with any other Windows programs, and it is assumed that some users will want to interface SimFIT with LaTeX while others would use the Microsoft Office suite.

D.1.1 Ghostscript

This valuable package is able to read PostScript files and output files in other formats, e.g., PDF, PNG, JPG, etc. There is no need to install this package any more as the run–time DLL is distributed with the SimFIT package.

D.1.2 GSview

This valuable program is able to read PostScript files then display or print them. There are also other useful features and so it should be installed. If it is not installed and you request to view or print PostScript files, SimFIT will use GhostScript to transform them into PDF format and pass them on to your PDF reader.

D.1.3 The interface between SimFIT, LaTeX , and Dvips

The .eps files generated by SimFIT have correct BoundingBox dimensions, so that LaTeX can use packages such as Dvips, Wrapfig, PSfrag and so on, as will be clear, for instance, from the LaTeX code for this manual.

D.1.4 SimFIT, Microsoft Office, and OpenOffice

SimFITcan import data tables with or without row and column labels in any of these formats.

a) Tables in SimFITformat
b) Tables copied to the clipboard from any program
c) Tables written to file using any program
d) Spreadsheet files saved in tab, space, comma, or semicolon delimited format
e) Spreadsheet files saved in XML or HTML format
f) Spreadsheet tables saved in Unicode format

Appendix E

MS_Office, OpenOffice, and LibreOffice

E.1 The easy way to import data into SimFIT

Often the easiest way to use data from from a spreadsheet program, word processor, or text editor in SimFIT is to use the clipboard which will now be illustrated by two simple examples. However, for repeated analysis or where extra features are required, it is better to create a SimFIT style data file, and this will be explained subsequently.

E.1.1 Creating a pie chart

This example illustrates how to plot a relatively small sample of positive numbers in pie chart format.

First of all open SimFIT to obtain the menu displayed in figure E.1 which will be referred to as the main SimFIT menu.

Now click on the [Editor] button which will open your text editor, and type in a data vector consisting of a single column of nonnegative numbers as in

1
2
3
4
5.

Then select, that is highlight, the column and copy to the clipboard as follows.

1
2
3
4
5

This data vector has been stored for further use so you can now click on the [A/Z] button to get an alphabetical list of SimFIT programs and select program **simplot**. Choose to create a pie chart and you will then reach the SimFIT file open control shown in figure E.2 where the [Paste] button will not be greyed out, showing that SimFIT has recognized the data vector that has been copied to the clipboard.

Note that, until you copy something else to the clipboard, this data set will remain ready for re-use by any SimFIT procedure, and on pressing the [Paste] button SimFIT gives you the option to create a temporary SimFIT data file then proceed with further analysis.

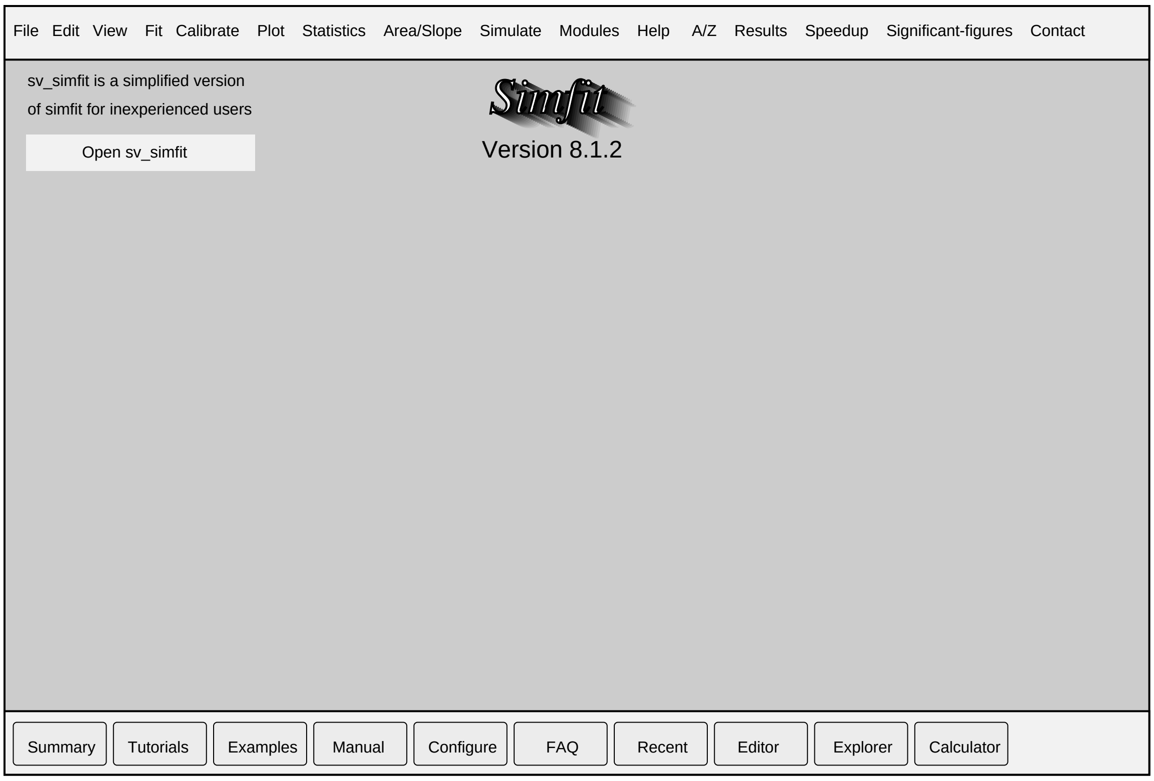

File Edit View Fit Calibrate Plot Statistics Area/Slope Simulate Modules Help A/Z Results Speedup Significant-figures Contact
sv_simfit is a simplified version
of simfit for inexperienced users
Open sv_simfit
Simfit
Version 8.1.2
Summary
Tutorials
Examples
Manual
Configure
FAQ
Recent
Editor
Explorer
Calculator

Figure E.1: The main SimFIT menu

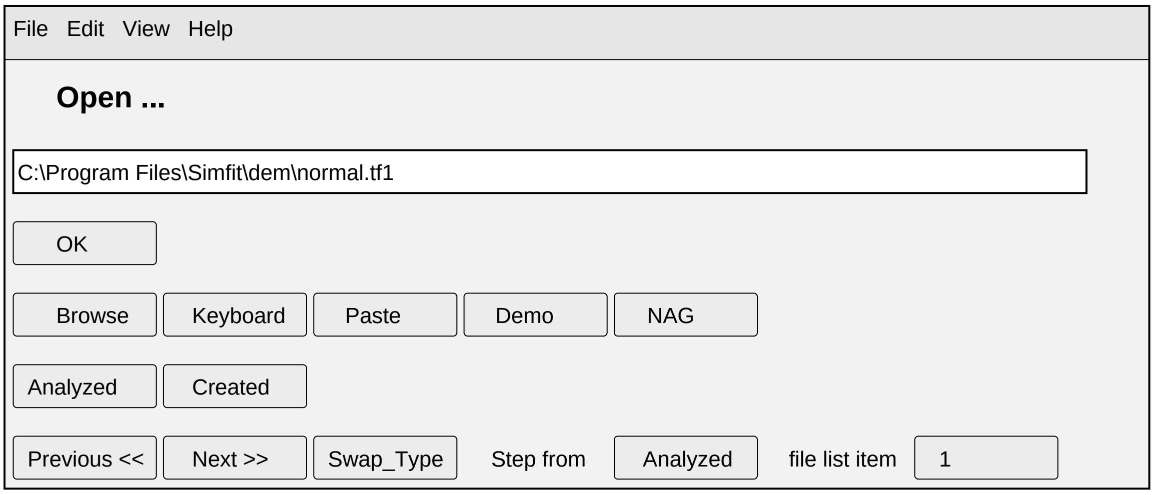
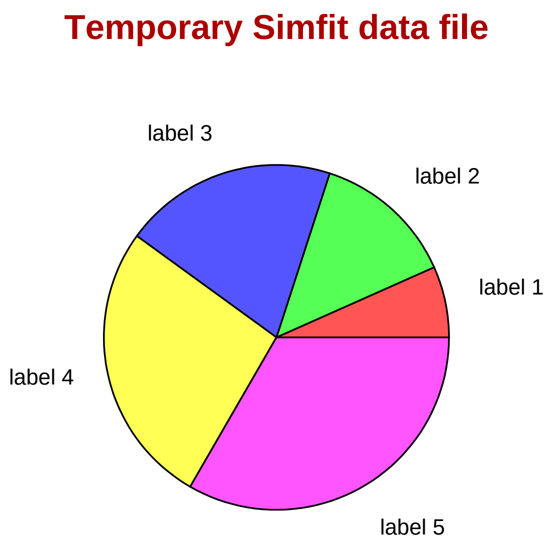


Figure E.2: The SimFIT file open control

First a default pie chart will be displayed which can then be edited if required, as in figure E.3 where the SimFIT segment fill styles are illustrated.



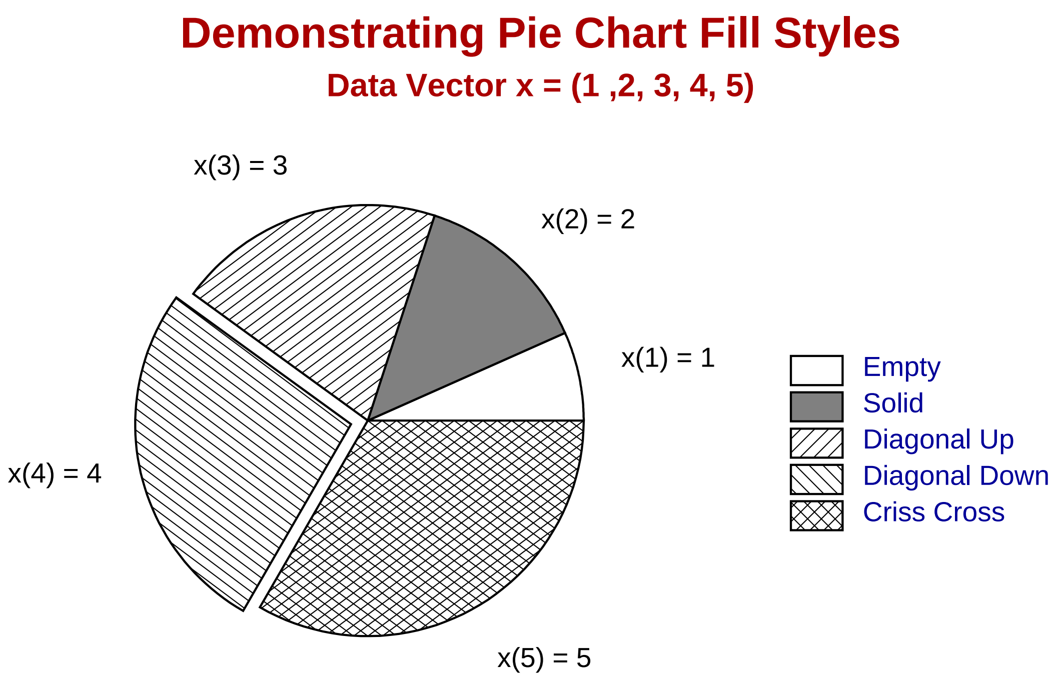


Figure E.3: Creating a SIMFIT pie chart

E.1.2 Creating a bar chart

This example explains how to input a data matrix with row and column labels into SimFIT in order to plot a simple bar chart, and also how to save the data matrix in SimFIT data file format for re-use.

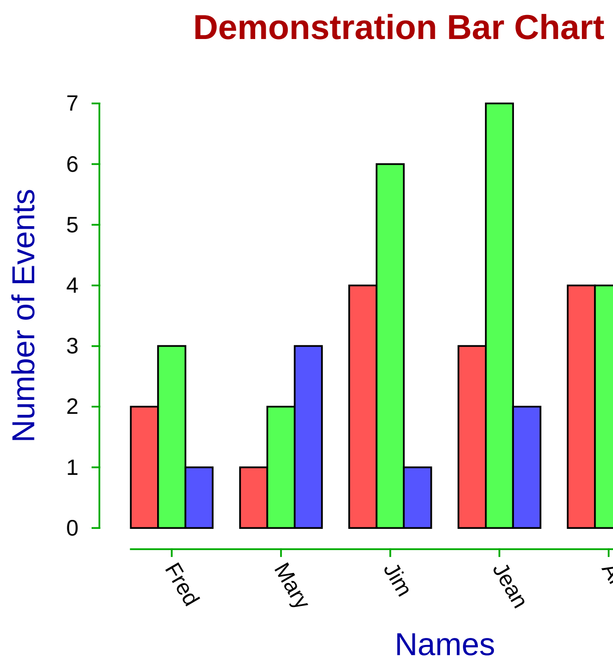
Type the following table into the Editor as for a pie chart, or highlight and copy to the clipboard from your spreadsheet. Note that each row and column has a label, also there is a dummy label in position 1,1 and labels must not contain spaces. For instance, you can use Time_of_Day as a label but not Time of Day.

name/variable	one	two	three
Fred	2	3	1
Mary	1	2	3
Jim	4	6	1
Jean	3	7	2
Anne	4	4	2
Kevin	5	1	3

After choosing the barchart option in program **simplot**, the clipboard data will be written to a temporary file called `clipboard_n.tmp` in your `My Documents\simfit\res` SimFIT results folder, accessible subsequently after pressing the main menu [Recent] button so you can save it for future re-use. The temporary SimFIT file will look like this

```
Temporary Simfit data file
  6  3
2 3 1
1 2 3
4 6 1
3 7 2
4 4 2
5 1 3
  12
begin{labels}
Fred
Mary
Jim
Jean
Anne
Kevin
one
two
three
end{labels}
ccyymmdd=20130703, hhmmss=085959
```

indicating the title, number of rows and columns, the data matrix, then the number of extra lines followed by row then column labels, and finally the date. Figure E.4 shows the resulting bar chart after preliminary then more extensive editing.



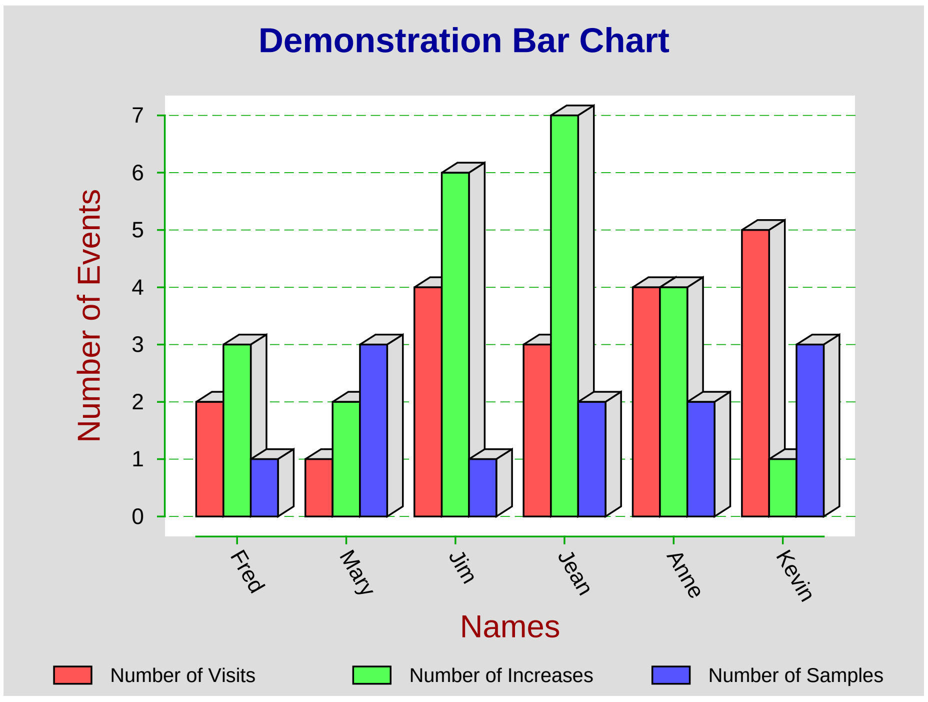


Figure E.4: Creating a SIMFIT bar chart

E.2 Definitions

Here we summarize the definitions to be used in this section.

E.2.1 Data tables

A data table is a n by m rectangular set of cells which contain only numerical values, as in the following example of a 2 by 3 data matrix.

```
1.1   1.2   1.3
2.1   2.2   2.3
```

Here the the columns are space-delimited, i.e., separated by one or more spaces, but we can also have comma-delimited tables, with zero or more spaces in addition to the commas, as shown next.

```
1.1,   1.2,   1.3
2.1,   2.2,   2.3
```

In spreadsheet and word-processing programs, data tables are stored in tab-delimited format, which results in tables being displayed with cells outlined by lines, as follows.

1.1	1.2	1.3
2.1	2.2	2.3

E.2.2 Labeled data tables

A labeled data table employs row 1 for column labels and column 1 for row labels with a dummy label in cell(1,1) as illustrated below.

R/C	Col-1	Col-2	Col-3
Row-1	1.1	1.2	1.3
Row-2	2.1	2.2	2.3

Here labels are shown as character strings, and the main table contains only numerical data. Note that there can be no spaces in labels of space-delimited tables, and underscores, hyphens, or similar must be used to avoid ambiguities. Spaces within labels also cause problems with importing SɪᴍFɪT results log files into Word and Excel.

E.2.3 Missing values

Missing values arise because data values are misplaced, not recorded, or discarded and this leads to a deficient table. The next example illustrates how a missing value can be indicated by an empty cell or a character variable in a data cell.

1.1		1.3
2.1	2.2	X

Any non-numerical characters can be used to indicate missing values, and pre-processing is required to replace empty cells by estimates derived in some way from the rest of the data set, before SɪᴍFɪT can process the data.

E.2.4 SɪᴍFɪT data files

SɪᴍFɪT data files can have an optional header with title on line 1, and number of rows and columns on line 2, then a data section table of numerical values, with no missing values, as follows.

```
Title for data
  2     3
1.1   1.2   1.3
2.1   2.2   2.3
```

E.2.5 SimFIT data files with labels

For some purposes row and column labels may be appended to SimFIT data files using a special symbolism as follows.

```
Title for data
  2,      3
1.1,    1.2,    1.3
2.1,    2.2,    2.3
7
begin{labels}
Row-1
Row-2
Col-1
Col-2
Col-3
end{labels}
```

The number 7 after the data table indicates that there are 7 extra lines appended, but this extra counter is optional. Note that the label in cell(1,1) is not recorded, and the labels are in sequential order with rows followed by columns.

Note that SimFIT data files are created automatically by all of the various methods described in the following sections, so that the user need only create a data table or labeled data table before using a particular method.

E.2.6 Clipboard data

The only type of clipboard data that can be used by SimFIT is where a data table or labeled data table has been selected and copied to the clipboard. If a space-delimited labeled table has been copied to the clipboard from a text editor like Notepad, there must be no spaces within labels, so use underscores.

E.2.7 Files exported from spreadsheet programs

The only types of files exported from spreadsheet programs that can be used by SimFIT are data tables or labeled data tables exported in text, XML, or HTML format. Text files can be delimited by spaces, tabs, commas, or semicolons, but files exported in space-delimited text format must have no spaces within labels, so use underscores.

E.3 Spreadsheet tables

Data for analysis by SimFIT would usually be held in a spreadsheet program, like Microsoft Office Excel or LibreOffice Calc, as in this example of an unselected table of multivariate statistical data from K.R. Gabriel in *Biometrika* 1971, **58**, 453–67.

Percent	Christian	Armenian	Jewish	Moslem	American	Shaafat	A-Tur	Silwan	Sur-Bahar
Toilet	98.2	97.2	97.3	96.9	97.6	94.4	90.2	94	70.5
Kitchen	78.8	81	65.6	73.3	91.4	88.7	82.2	84.2	55.1
Bath	14.4	17.6	6	9.6	56.2	69.5	31.8	19.5	10.7
Electricity	86.2	82.1	54.5	74.7	87.2	80.4	68.6	65.5	26.1
Water	32.9	30.3	21.1	26.9	80.1	74.3	46.3	36.2	9.8
Radio	73	70.4	53	60.5	81.2	78	67.9	64.8	57.1
TV set	4.6	6	1.5	3.4	12.7	23	5.6	2.7	1.3
Refrigerator	29.2	26.3	4.3	10.5	52.8	49.7	21.7	9.5	1.2

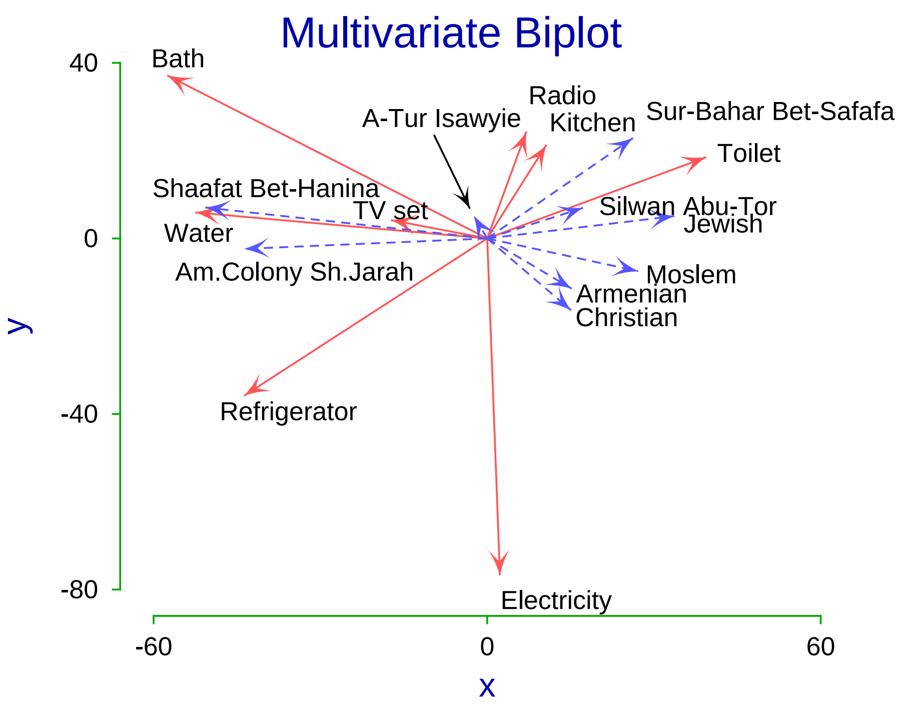
Such tables must be rectangular, with optional row and column labels, and all other cells filled with numerical data (missing data will be discussed later). Often it would only be necessary to select cells containing numerical values from such a table by highlighting as follows.

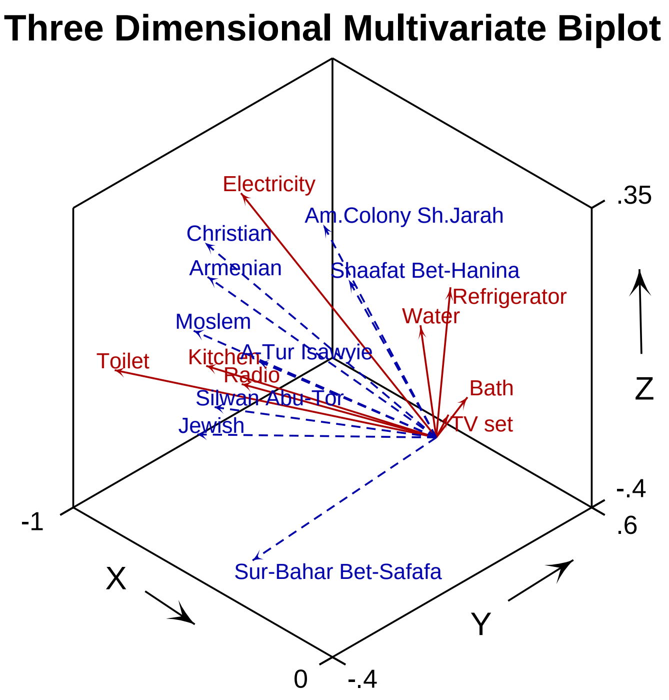
Percent	Christian	Armenian	Jewish	Moslem	American	Shaafat	A-Tur	Silwan	Sur-Bahar
Toilet	98.2	97.2	97.3	96.9	97.6	94.4	90.2	94	70.5
Kitchen	78.8	81	65.6	73.3	91.4	88.7	82.2	84.2	55.1
Bath	14.4	17.6	6	9.6	56.2	69.5	31.8	19.5	10.7
Electricity	86.2	82.1	54.5	74.7	87.2	80.4	68.6	65.5	26.1
Water	32.9	30.3	21.1	26.9	80.1	74.3	46.3	36.2	9.8
Radio	73	70.4	53	60.5	81.2	78	67.9	64.8	57.1
TV set	4.6	6	1.5	3.4	12.7	23	5.6	2.7	1.3
Refrigerator	29.2	26.3	4.3	10.5	52.8	49.7	21.7	9.5	1.2

However, sometimes row and column labels could also be needed, when a labeled table with cells containing either labels or numerical values would be selected, as follows.

Percent	Christian	Armenian	Jewish	Moslem	American	Shaafat	A-Tur	Silwan	Sur-Bahar
Toilet	98.2	97.2	97.3	96.9	97.6	94.4	90.2	94	70.5
Kitchen	78.8	81	65.6	73.3	91.4	88.7	82.2	84.2	55.1
Bath	14.4	17.6	6	9.6	56.2	69.5	31.8	19.5	10.7
Electricity	86.2	82.1	54.5	74.7	87.2	80.4	68.6	65.5	26.1
Water	32.9	30.3	21.1	26.9	80.1	74.3	46.3	36.2	9.8
Radio	73	70.4	53	60.5	81.2	78	67.9	64.8	57.1
TV set	4.6	6	1.5	3.4	12.7	23	5.6	2.7	1.3
Refrigerator	29.2	26.3	4.3	10.5	52.8	49.7	21.7	9.5	1.2

As an example, consider the next figures, showing these data from the SimFIT test file `houses.tf1` when displayed as biplots. The dummy label in cell(1,1) is not used.





This document explains how to transfer data from such spreadsheet tables into SIMFIT.

E.4 Using the clipboard to transfer data into SIMFIT

Two formats can be used to transfer a tabular data set into SIMFIT for analysis.

- The cells contain only numerical data
- The cells also have associated row and column labels

If labels are present then the column labels (usually variables) must occupy the top row of cells, the row labels (usually cases) must occupy the first column of cells, and there must be a dummy row/column label in cell(1,1). Purely numerical tables are used for curve fitting and statistical analysis, where the identity of the observations is immaterial. Row and column labels can be added just to help identify the data retrospectively, but they are also used in multivariate statistics to identify plotting symbols or label dendrograms. If there are no empty cells, such tables can be imported directly into SIMFIT by a [Paste] button, or be transformed into SIMFIT data files using program **maksim**.

Cells can be delimited by spaces, commas, semicolons, or tabs, and there must be no empty cells. If semicolons or tabs are used as column separators, then using continental notation like 1,5 for 1.5 is allowed. Further, if a space-delimited table has been copied to the clipboard from a text editor like Notepad, there must be no spaces within labels.

E.4.1 Pasting data from the clipboard directly into SIMFIT

In Microsoft Excel, LibreOffice Calc, or almost any text editor (such as Notepad) or word processing program, the rectangular table required is selected then copied to the clipboard. The appropriate SIMFIT program is opened and, when a data set is requested, the [Paste] button on the file opening control is pressed. If the table is rectangular, with no missing values, one of the following options will be available.

- Data: use as a SIMFIT data file (no labels)
- Data: use as a SIMFIT data file (with labels)

If the option to use as a SIMFIT data file is chosen from this SIMFIT clipboard control, the data will be written to a temporary file then analyzed. The names of these temporary files are displayed so they can be saved retrospectively. The options will only be available if the data are consistent with the SIMFIT format, otherwise error messages will be displayed.

E.4.2 Converting data from the clipboard into a SIMFIT file

This involves the use of the SIMFITprogram **maksim**. The clipboard table is input into **maksim** exactly as just described for any SIMFIT program, but then there are many options.

- Rows or columns selected from a check list can be suppressed or restored
- Sub-tables can be selected by properties
- Sub-tables can be Saved As ... SIMFIT data files.

The advantage of this technique is that, from a single table copied to the clipboard, numerous selected sub-tables can be archived, so that retrospective analysis of data sub-sets can be performed.

E.5 Using spreadsheet output files to transfer data into SIMFIT

SIMFIT will accept application files (created by any of the data preparation programs) containing rectangular data tables with or without labels, but with no missing values, and the exported files will be transformed interactively into temporary SIMFIT data files. Otherwise, the exported files can be transformed into SIMFIT data files using program **maksim**, in which case the options just described for clipboard data will also be available. Care is needed when using continental notation like 1,5 for 1.5 in text files with spaces or commas as delimiters, as mentioned below. Spreadsheets with missing values must be pre-processed, e.g., using the macro **simfit6.xls** which is in the SIMFIT documents installation folder.

E.5.1 Space-delimited text files (.txt)

These files must not have spaces to represent empty cells, or spaces within labels. For example, use `TV_set` instead of `TV set`. Using 1,5 for 1.5 is not acceptable with this file type, since commas are interpreted as column separators.

E.5.2 Comma-delimited text files (.csv with standard Excel setup)

If it is wished to use commas to separate thousands, i.e., triples of figures before the decimal point, then the cell must be quoted. That is, writing 1,234.56 as "1,234.56" to represent 1234.56. This has to be done explicitly in text editors like Notepad, but Excel and Calc will add the quotes. Note that, when the SIMFIT file is output, it will have 1234.56 not 1,234.56. Any comma pairs ,, will be taken to represent blank cells and be expanded to ,X,. Using 1,5 for 1.5 is not acceptable with this file type, since commas are reserved for use as column separators.

E.5.3 Semicolon-delimited text files (.csv with continental Excel setup)

Commas for thousands in numeric cells will still have to be quoted and semicolon pairs ;; will be taken to represent blank cells and be expanded to ;X;. Using 1,5 for 1.5 is acceptable with this file type, since semicolons are reserved for use as column separators.

E.5.4 Tab-delimited files (.txt)

Commas for thousands in numeric cells will still have to be quoted, and adjacent tabs will be interpreted as empty cells and be expanded to (tab)X(tab). Using 1,5 for 1.5 is acceptable with this file type, since tabs are reserved for use as column separators.

E.5.5 Unicode (.txt)

If SIMFIT program **maksim** fails to read Unicode files, then switch Notepad into ASCII text mode, read in the Unicode file, then export again as an ASCII text file, which will then be acceptable

E.5.6 Web documents (.xml, .html, .htm, .mht, .mhtml)

Using 1,5 for 1,5 is acceptable with these file types, since special techniques are used for column separators, but such files must have the file extension .xml, .html, .htm, .mht, or .mhtml.

Also note that XML files must follow this case-sensitive convention.

- The tag `<Table` indicates the start of a table and `</Table>` indicates the end.
- The tag `<Row` indicates the start of a row and `</Row>` indicates the end.
- The tag `<Cell` indicates the start of a new cell and `</Cell>` indicates the end.
- The following tokens are used:
  ss:Index="2" to indicate an empty cell in column 1,i.e., the next cell is cell 2.
  ss:Type="String" for a label, and
  ss:Type="Number" for a numeric cell.

HTML files must use standard case-insensitive tags. Files exported from Microsoft Office Excel and Libre-Office Calc use these conventions.

E.6 Exporting SIMFIT data files from Excel

E.6.1 Using simfit4.xls with Excel to create SIMFIT data files

The **simfit4.xls** macro that was distributed with SIMFIT version 4 and can be found in the SIMFIT documents installation folder is a very simple program that remains useful, especially for exporting curve fitting files from Excel, for these reasons:

- it can gather together non-adjacent columns;
- it can check for nondecreasing values, e.g., an independent variable in column 1; and
- it can check for nonnegative values, e.g., positive weights in column 3.

However, it cannot deal with labels, missing values, or perform data transformations, so the more versatile **simfit6.xls** macro distributed with SIMFIT version 6 is now the preferred tool for creating export files in SIMFITdata file format from Excel. The **simfit6.xls** macro also validates the data, checking for basic errors that would cause SIMFIT to reject the data table.

E.6.2 Using simfit6.xls with Excel to create transformed SIMFIT data files

The **simfit6.xls** macro replicates some of the functionality of the **simfit4.xls** macro, and previous versions of the **simfit6.xls** macro, but offers additional features for filling empty cells in the data table, for performing transformations of the data table, and for transposing the data table when the SIMFIT export file is created.

In contrast with **simfit4.xls**, and previous versions of **simfit6.xls**, which did not modify the data table in the worksheet, **simfit6.xls** now applies its editing and transformations to the actual worksheet, and leaves this modified worksheet in place after the macro has terminated.

E.6.2.1 Invoking the simfit6.xls macro

The **simfit6.xls** workbook can be located in the SIMFIT documents folder, and in order to use the macro you may need to change the Excel macro security setting. Navigate
`Tools>Macro>Security` from the Menu Bar and set the security level to Medium.

The **simfit6.xls** macro has been written and tested with the Microsoft Office XP Excel Version 2002, but should work satisfactorily with other versions from Excel 97 onwards.

E.6.2.2 Step 1: Open the simfit6.xls workbook

The **simfit6.xls** workbook acts purely as a carrier for the simfit6 macro and contains no worksheets accessible to the user. The workbook must, however, be opened in the normal manner, in order to make the macro available. With the macro security level set to Medium, as described above, you will be warned that workbook contains macros. Click the Enable Macros button to complete the opening procedure. After the workbook has been opened, there is no visible sign of its presence, because it operates in the background.

E.6.2.3 Step 2: Select the data table within the user's workbook

Open the user's workbook containing the SIMFIT table and activate the worksheet which holds the table. The SIMFIT table (either labeled or unlabeled) can be located anywhere within the spreadsheet, i.e. it does not have to occupy Row 1 and Column 1 of the spreadsheet. Next select the SIMFIT table with the mouse in order to inform the macro of the position and dimensions of the table. Selection is carried out by left-clicking the mouse on the top left cell of the table and dragging (moving the mouse with the left key held down) to the bottom right cell of the SIMFIT table. [The above selection is called a single-area selection. The simfit6 macro does not support multiple-area selections.]

E.6.2.4 Step 3: Invoke the simfit6.xls macro

The macro is invoked either by navigating `Tools>Macro>Macros` from the Menu Bar, or by `Alt+F8` from the keyboard (meaning hold the `Alt` key down whilst pressing the `F8` key). Click on simfit6 in the list of macros, then click on the Run button, and respond to the various dialog boxes which the macro uses for communicating with the user. If simfit6 does not appear in the list of macros, check that Step 1 has been carried out successfully.

E.6.2.5 Using the simfit6.xls macro

Since simfit6 applies its transformations directly to the table which is selected in the active worksheet, it is necessary to take precautions to preserve the original contents of the worksheet in order to safeguard the user's data. The simfit6 macro does this by offering the following two options to the user as soon as the macro commences:

- First, if the macro detects that the user's workbook has not been saved on disk, it offers the user the option of saving the workbook.
- The second option is to make a copy of the active worksheet within the user's workbook and to have the macro apply its transformations to this copy, leaving the original worksheet unaltered.

Next the macro ascertains from the user whether or not the selection contains a labeled table, after which it notifies the user of the data table's dimensions and obtains confirmation that a SimFIT export file is to be created. Users are reminded that a SimFIT labeled table must contain **both** row **and** column labels.

A simfit6 data table must not contain any non-numeric cells or any cells containing formulas. Any non-numeric cells found in the data table are reported to the user, but their presence forces the macro to terminate, as SimFIT does not accept non-numeric data items.

If any cell in the data table is found to contain a formula, this is likewise reported to the user and will force the macro to terminate. The user must replace all formulas in the data table by their numeric values before re-running the macro. [Tip: This is simply done by selecting the data table and navigating `Edit>Copy, Edit>Paste Special>Values`.]

E.6.2.6 Filling empty cells found in the data table

If any empty cells are found in the data table, the user's processing options are displayed in the `Data Table contains empty cells` dialog box. The options are as follows.

1. Report references of empty cells
2. Replace empty cells in rows by the row average
3. Replace empty cells in rows by by row-wise interpolation
4. Replace empty cells in columns by the column average
5. Replace empty cells in columns by by column-wise interpolation
6. Delete all rows containing an empty cell
7. Delete all columns containing an empty cell
8. Delete both rows and columns containing an empty cell

Row-wise interpolation acts as follows: an empty cell range in the middle of the row will be filled by the average of the neighboring cells adjacent to the range; and an empty cell range which extends to the beginning or end of the row will be filled by the neighboring adjacent cell to the range. Column-wise interpolation applies the same algorithm to columns.

Note that options 2 to 8 are applied to **all** rows (or columns) which contain empty cells, i.e. the user is not able to select a different option for each row or column. Note also that whereas options 2 to 8 will fill all empty cells in a data table, option 1 merely reports the empty cells and leaves the data table unchanged. In the latter case the macro will terminate after the reporting has been completed, as empty cells in the data table are not acceptable to SIMFIT.

E.6.2.7 Validation Checks Completed

If the data table passes the checks described above for non-numeric cells, formula cells and empty cells, the macro displays the `Validation Checks Completed` window.

- Perform transformations
- Create Simfit file

Clicking the `Perform transformations` option allows one or multiple transformations to be carried out on the data table as described below, after which the option to create the SIMFIT file will be offered.

If no transformations are required, click the `Create Simfit file` option to bypass the transformation process and commence creation of the SIMFIT file.

E.6.2.8 Performing transformations of the data table

Transformations can be selected via the `Select Required Transformation` dialog box, as follows.

1. Normalize columns to length 1
2. Normalize columns to standard deviation 1
3. Centralize columns to mean 0
4. Centralize columns to mean 0 and normalize columns to standard deviation 1
5. Normalize rows to length 1
6. Normalize rows to standard deviation 1
7. Centralize rows to mean 0
8. Centralize rows to mean 0 and normalize rows to standard deviation 1

Multiple transformations can be performed in sequence: each transformation changes the data table, and a subsequent transformation will be applied to the modified data table. Option 4 is an example of a sequence of Option 3 followed by Option 2, but is offered as a separate option because this combination is frequently used. It is the user's responsibility to ensure that any chosen sequence of transformations is viable from a statistical analysis standpoint.

The macro will report any normalization transformation for which the divisor evaluates to zero, as this would result in a division by zero error. However this occurrence is not treated as a catastrophic failure, as the transformation is declined and the data table is left unaltered.

E.6.2.9 Transposing the SIMFIT table

This facility is provided via the dialog box
`Option to create Simfit File from the transposed Data Table`.
Note that this option operates only on the exported file and does not alter the table which is contained within the worksheet. The possibilities are as follows.

1. Create Simfit file from the Data Table

2. Create Simfit file from the transposed Data Table

After this choice has been made the macro proceeds to create the export file with the SIMFIT-defined structure. At the end of this process, the macro will terminate after reporting successful creation of the SIMFIT file.

E.6.2.10 Inspecting and saving the modified worksheet

Because **simfit6.xls** makes successive changes to the actual worksheet, it can be quite useful to be able to inspect and/or save the modified worksheet. This can only be done when the macro has terminated, either on normal completion or by taking a termination option at an intermediate stage of processing, such as after filling empty cells, or after any transformation in a sequence, or before creating the final SIMFIT file.

If the user wishes to save the modified workbook it is advisable to do this using the `Save As` method with a different filename, otherwise the original file will be overwritten by the modified version. In order to resume processing, the user must either:

- Select the table in the modified worksheet and re-run the macro; or
- Re-load the original worksheet and re-run the macro, repeating the previous processing steps.

E.6.2.11 The History Log

Every time the **simfit6** macro is run, a new worksheet is inserted into the user's workbook to hold the History Log. Any History Log worksheet which is already present in the workbook (created by a previous macro run) is deleted. [Tip: if you wish to retain History Logs from previous macro runs, rename their worksheets (History Log 1, History Log 2, etc) before running the macro.]

The History Log records:

- The sequence of processes which have been applied by the macro. This is important as an audit trail of how the final data table was derived from the original version, especially if the process sequence has to be repeated at some future time.
- The processing time spent on the main iterative processes (to assist in estimating overall processing times for very large data tables).
- Any errors which were detected during processing.

In order to ensure that the History Log travels with the SIMFIT file, its contents are appended to the SIMFIT file as additional lines between the delimiters
`begin{transformation history}` and `end{transformation history}`.

E.6.2.12 Processing very large data tables

The processing steps for very large data tables can run into several minutes. During these periods, the mouse pointer will be displayed as an hourglass icon, indicating that the user must wait for the current process to finish.

Timing tests on a 60000 rows by 50 columns matrix indicated that the longest processes were the copying of the active worksheet (t_1), the actual transformations (t_2), and the creation of the SIMFIT file (t_3), where the times were in the ratio

$$t_1 : t_2 : t_3 = 69 : 62 : 14.$$

The length of time for the copying of the active worksheet is the reason why this process has been made optional, but the user who chooses to omit this step must take alternative precautions to safeguard the original data.

E.7 Importing SIMFIT results tables into documents and spreadsheets

E.7.1 SIMFIT results files

Each time a data set is analyzed the results are written to a file called `f$result.txt`, and the current files are renamed so that the existing `f$result.txt` becomes `f$result.001` while `f$result.001` becomes `f$result.002` and so on. These can be viewed using the [Results] option from the main SIMFIT menu.

These SIMFIT results files are formatted so that the numbers displayed only contain the number of significant figures that are meaningful in context. For instance, probabilities will usually only have four digits after the decimal point, which indicates that it does not make any sense to consider any subsequent digits for purpose of statistical testing, and in any case probability estimates will not be accurate for more than about four digits. Furthermore, as experimental data are rarely more accurate than about three or four significant figures anyway, it may be wishful thinking to ever consider more than say six. In addition, the tables are formatted using a fixed font with scientific notation to line up column entities irrespective of absolute size, and many users do not want this in a thesis or published document. Naturally, these arguments do not apply to integers.

E.7.2 Preliminary analysis

As an example consider the following case with the title Table 1.

```
Table 1
1-Way Analysis of Variance:  1 (Grand Mean  4.316E+01)

Transformation:-  x (untransformed data)
Source              SSQ       NDOF      MSQ          F          p
Between Groups    2.19344E+03     4   5.48316E+02  5.61546E+01  0.0000
Residual          2.44130E+02    25   9.76520E+00  ...          ...
Total             2.43757E+03    29   ...          ...          ...
```

Actually, most users would want to import such a table formatted as tabbed-text, html, xml, or LaTeX into documents such as a report, thesis, or publication looking something like Table 2.

Table 2
1-Way Analysis of Variance: 1 (Grand Mean 43.16)
Transformation:- x (untransformed data)

Source	SSQ	NDOF	MSQ	F	p
Between Groups	2193.44	4	548.316	56.1546	0.0000
Residual	244.130	25	9.76520		
Total	2437.57	29			

Or even, for those with artistic leanings, possibly something like Table 3.

Table 3
1-Way Analysis of Variance: Grand Mean 43.16
Transformation: x (untransformed data)

Source	SSQ	NDOF	MSQ	F	p
Between Groups	2193.44	4	548.316	56.1546	0.0000
Residual	244.130	25	9.76520		
Total	2437.57	29			

E.7.3 The procedure

This article explains the procedures required to export tables from from SimFIT results files into forms suitable for inclusion into word processors, spreadsheet programs, website scripts, or even professional document preparation systems such as LaTeX. Also, decimal points can be replaced by commas as in continental notation if required.
It is important to realize that the [Results] option from the SimFIT main menu gives access to all the currently saved results files.

- **Choosing a results file.**

  Early versions of SimFIT allowed users to name results files individually to avoid anything being lost. However, now that up to 100 results files are saved and users have the option [Results] from the main SimFIT menus from which to view, print, save, edit, or export tables, this is no longer usually necessary. Clearly, if results are always required for retrospective use, regular back-up or saving will be necessary.

- **Extracting a table.**

  From the [Extract tables] option view the file to make sure it is the one required then copy to the clipboard only the table required along with any associated header and trailer sections ... but nothing else.

- **Preliminary editing.**

  Sometimes editing of the file is required to make sure that every row of the table has exactly the same number of columns. So note that, for extracting a table there can be no empty cells, and each cell must contain precisely one word. Any column titles must be edited so that they consist of one word, for instance changing `Time of Day` to `Time_of_Day`, or filling empty cells by three dots. Added underscores and sets of three contiguous dots can be removed when the final table is written to file. A pre-processing option is provided for editing before attempting to create a table.

  Note that often tables have cells with added comments relating to goodness of fit or results of statistical testing, and these no not need to be underscored. There are also special tables with only two columns containing several words in some cells, and Example 1 later will make this clear.

- **Viewing the hashtag table.**

  The algorithm attempts to identify cells in a table by inserting a hashtag between every column. If the algorithm succeeds there would be no need to view this hashtag table. However this option should be switched on until the process of the algorithm is understood, or if it fails and you need to see why.

  The hashtag table is clarified in Example 1.

- **Headers and trailers.**

  Frequently tables have header and trailer sections that are descriptive and not part of the table itself. As these can have strings of word and numbers that would confuse the algorithm checking that every row must have the same number of columns, they must be identified. This is done by using buttons on a window that allows the header and trailer lines to be highlighted. If this is not done the table creating algorithm will fail.

  Selecting headers and trailers is clarified in Example 2.

- **Saving the table.**

  For programs that produce Windows quality hardcopy the table should be saved as html or xml as these will import directly into word processors or spreadsheets. Tabbed-text is also available but is much less versatile than html or xml, and LaTeX is available for those up to it.

- **Fine tuning.**

  Inevitably there will often be the need for dealing with details. For instance, users will sometimes want to replace `alpha` by α or `chi-sqd` by χ^2 and this can be done for html, xml, and LaTeX output but not for tabbed-text. On the other hand html, xml, and LaTeX have reserved letters and these must be dealt with retrospectively.

For instance, consider the transformation of the following expressions which can be done in html, xml, and LaTeX but not in tabbed text.

Original	Transformed
>=	$\geq$
=<	$\leq$
alpha	α
beta	β
delta	δ
gamma	γ
lambda	λ
infinity	∞
P(chi-sqd >= TS) = 0.2037	$P(\chi^2 \geq TS) = 0.2037$

Where ambiguity could arise in LaTeX using underscores or similar special characters they will be replaced by question marks. So LaTeX users should search for ? characters to replace for the final table.

- **Padding with zeros.**

  As the [Extract tables] option will never remove significant figures a problem arises if users wish to replace numbers in scientific notation by floating point representation. In order to extend the range over which this can be done, padding zeros can be introduced as illustrated in this next table.

Scientific notation	Padding	Floating point representation
1.234E-01	0	.1234
1.234E+00	0	1.234
1.234E+01	0	12.34
1.234E+02	0	123.4
1.234E-02	2	0.01234
1.234E-01	2	0.12340
1.234E+00	2	1.23400
1.234E+01	2	12.3400
1.234E+02	2	123.400
1.234E+03	2	1234.00
1.234E+04	2	12340.0
1.234E-04	4	0.0001234
...	...	...
1.234E+06	4	1234000.0

Evidently increasing the number of padding zeros increases the range over which transformation from scientific to floating point representation can be achieved, and the default is two which allows a wide range, but four ensures that a mixture of transformed and untransformed numbers will occupy the same width in the columns of a table.

However, there is here a problem because adding padding zeros could suggest all trailing zeros are meaningful. For instance, the number 1.2341213179 stored in the computer could be written as 1.234E+00 in SimFIT output because the analysis in question only justifies accuracy or meaning for up to four significant figures. However, 1.23400 could be mistaken for indicating the internal representation with twelve significant figures. So users may wish to suppress trailing zeros in such cases, noting that this could result in numbers with different widths in a column. In any case, the transformation from scientific notation to floating point can be switched off.

Sometimes, for instance with numerical analysis where more significant figures are justified than with data analysis, eight significant figures are output, and some procedures can optionally allow more. In addition special DLLs dedicated for particular routines can be supplied for this purpose.

E.7.4 Example 1

This example shows the transformation of a special type of SIMFIT table containing just two columns separated by equals signs (i.e., =) and containing cells with multiple words. As long as the equals signs are perfectly lined up and there is no header or trailer section, this type of table can always be transformed. Here is the table with no header or trailer sections as extracted using the [Results] then [Extract tables] options from the main SIMFIT menu using the results file `f$result.txt` following the exhaustive analysis of a vector process used to analyze data contained in the default test file `normal.tf1`.

```
Sample size                     =      50
Minimum value                   = -2.20820E+00
Maximum value                   =  1.61750E+00
Coefficient of skew             = -1.66905E-02
Coefficient of kurtosis         = -7.68395E-01
Lower Hinge  (25th percentile)  = -8.55015E-01
Median value (50th percentile)  = -9.73615E-02
Upper Hinge  (75th percentile)  =  7.85965E-01
Sample mean                     = -2.57897E-02
Sample standard deviation       =  1.00553E+00
Coefficient of variation (CV%)  =  > 100%
Standard error of the mean      =  1.42203E-01
Upper 2.5% t-value              =  2.00958E+00
Lower 95% con lim for mean      = -3.11558E-01
Upper 95% con lim for mean      =  2.59978E-01
Sample variance                 =  1.01109E+00
Lower 95% con lim for variance  =  7.05519E-01
Upper 95% con lim for variance  =  1.57006E+00
Shapiro-Wilks W statistic       =  9.62693E-01
Significance level for W        =  0.1153
Conclusion                      = Tentatively accept normality
```

This is the corresponding intermediate hashtag table.

```
Sample size                     #      50
Minimum value                   # -2.20820E+00
Maximum value                   #  1.61750E+00
Coefficient of skew             # -1.66905E-02
Coefficient of kurtosis         # -7.68395E-01
Lower Hinge  (25th percentile)  # -8.55015E-01
Median value (50th percentile)  # -9.73615E-02
Upper Hinge  (75th percentile)  #  7.85965E-01
Sample mean                     # -2.57897E-02
Sample standard deviation       #  1.00553E+00
Coefficient of variation (CV%)  #  > 100%
Standard error of the mean      #  1.42203E-01
Upper 2.5% t-value              #  2.00958E+00
Lower 95% con lim for mean      # -3.11558E-01
Upper 95% con lim for mean      #  2.59978E-01
Sample variance                 #  1.01109E+00
Lower 95% con lim for variance  #  7.05519E-01
Upper 95% con lim for variance  #  1.57006E+00
Shapiro-Wilks W statistic       #  9.62693E-01
Significance level for W        #  0.1153
Conclusion                      # Tentatively accept normality
```

The hashtag table is very useful for detecting the source of errors. The table making algorithm attempts to locate the position separating columns and writes a hashtag there. If every row has the same number of columns then every row will have the same number of hashtags and the algorithm has succeeded. Observing this hashtag table when the algorithm has failed will allow you identify then correct the error.

Here is the selected table as it would be written to the output file.

Sample size	50
Minimum value	-2.2082000
Maximum value	1.6175000
Coefficient of skew	-0.0166905
Coefficient of kurtosis	-0.7683950
Lower Hinge (25th percentile)	-0.8550150
Median value (50th percentile)	-0.0973615
Upper Hinge (75th percentile)	0.7859650
Sample mean	-0.0257897
Sample standard deviation	1.0055300
Coefficient of variation (CV%)	> 100%
Standard error of the mean	0.1422030
Upper 2.5% t-value	2.0095800
Lower 95% con lim for mean	-0.3115580
Upper 95% con lim for mean	0.2599780
Sample variance	1.0110900
Lower 95% con lim for variance	0.7055190
Upper 95% con lim for variance	1.5700600
Shapiro-Wilks W statistic	0.9626930
Significance level for W	0.1153
Conclusion	Tentatively accept normality

Here it is with a few minor cosmetic changes.

Exhaustive analysis of a vector	
Sample size	50
Minimum value	-2.2082000
Maximum value	1.6175000
Coefficient of skew	-0.0166905
Coefficient of kurtosis	-0.7683950
Lower Hinge (25th percentile)	-0.8550150
Median value (50th percentile)	-0.0973615
Upper Hinge (75th percentile)	0.7859650
Sample mean	-0.0257897
Sample standard deviation	1.0055300
Coefficient of variation (CV%)	> 100%
Standard error of the mean	0.1422030
Upper 2.5% t-value	2.0095800
Lower 95% confidence limit for mean	-0.3115580
Upper 95% confidence limit for mean	0.2599780
Sample variance	1.0110900
Lower 95% confidence limit for variance	0.7055190
Upper 95% confidence limit for variance	1.5700600
Shapiro-Wilks W statistic	0.9626930
Significance level for W	0.1153
Conclusion: *Tentatively accept normality*	

E.7.5 Example 2

From fitting a two-exponential model to data in the test file `exfit.tf4` using program **exfit** the following results can be extracted from the results file.

```
For best-fit 2-exponential function

Parameter    Value      Std.Error  Lower95%cl  Upper95%cl    p
   A(1)    8.5255E-01  6.7731E-02  7.1332E-01  9.9177E-01  0.0000
   A(2)    1.1765E+00  7.4779E-02  1.0227E+00  1.3302E+00  0.0000
   k(1)    6.7935E+00  8.5386E-01  5.0383E+00  8.5486E+00  0.0000
   k(2)    1.1121E+00  5.1128E-02  1.0070E+00  1.2172E+00  0.0000
   AUC     1.1834E+00  1.4714E-02  1.1531E+00  1.2136E+00  0.0000
AUC is the area under the curve from t = 0 to t = infinity
Initial time point (A)   =  3.5983E-02
Final time point   (B)   =  1.6110E+00
Area over range (A,B)    =  9.3832E-01
Average over range (A,B) =  5.9575E-01
```

Now the file has an additional head and trailer section so, if the full table is selected, it will have to be highlighted as follows in the header and trailer selection control as shown next, where the header is colored magenta and the trailer colored cyan.

```
For best-fit 2-exponential function

Parameter        Value     Std.Error  Lower95%cl  Upper95%cl    p
   A(1)       8.5255E-01  6.7731E-02  7.1332E-01  9.9177E-01  0.0000
   A(2)       1.1765E+00  7.4779E-02  1.0227E+00  1.3302E+00  0.0000
   k(1)       6.7935E+00  8.5386E-01  5.0383E+00  8.5486E+00  0.0000
   k(2)       1.1121E+00  5.1128E-02  1.0070E+00  1.2172E+00  0.0000
    AUC       1.1834E+00  1.4714E-02  1.1531E+00  1.2136E+00  0.0000
AUC is the area under the curve from t = 0 to t = infinity
Initial time point (A)    = 3.5983E-02
Final time point (B)       = 1.6110E+00
Area over range (A,B)      = 9.3832E-01
Average over range (A,B) = 5.9575E-01
```

Using two padding zeros this leads to the following table.

For best-fit 2-exponential function

Parameter	Value	Std.Error	Lower95%cl	Upper95%cl	p
A(1)	0.852550	0.067731	0.713320	0.991770	0.0000
A(2)	1.176500	0.074779	1.022700	1.330200	0.0000
k(1)	6.793500	0.853860	5.038300	8.548600	0.0000
k(2)	1.112100	0.051128	1.007000	1.217200	0.0000
AUC	1.183400	0.014714	1.153100	1.213600	0.0000

AUC is the area under the curve from $t = 0$ to $t = \infty$
Initial time point (A) = 0.035983
Final time point (B) = 1.611
Area over range (A,B) = 0.93832
Average over range (A,B) = 0.59575

However, note that, with this example, three points emerge.

1. Numbers outside the main table will also be transformed into floating point numbers.
2. Equals signs lined up the trailer will not lead directly to secondary tabulation.
3. Some special words, like infinity, will be recognized.

So, because a certain amount of fine tuning will be required, the possibilities for handcrafting are endless. Here, for example, the header is enlarged by adding a formula, while the trailer is added in the form of a footnote to the main table.

For the best-fit 2-exponential function

$$f(t) = A_1 \exp(-k_1 t) + A_2 \exp(-k_2 t)$$

Parameter	Value	Std.Error	Lower95%cl	Upper95%cl	p
A_1	0.852550	0.067731	0.713320	0.991770	0.0000
A_2	1.176500	0.074779	1.022700	1.330200	0.0000
k_1	6.793500	0.853860	5.038300	8.548600	0.0000
k_2	1.112100	0.051128	1.007000	1.217200	0.0000
AUC	1.183400	0.014714	1.153100	1.213600	0.0000

Area under the curve $AUC = \int_0^\infty \hat{f}(t)\, dt$

Initial time point (A) = 0.035983

Final time point (B) = 1.611

Area over range (A,B) = 0.93832

Average over range (A,B) = 0.59575

E.7.6 Example 3

A special situation exists with symmetric matrices where just a lower or upper triangle is displayed, and also some other related situations. For instance, following on from the previous example we have the parameter correlation matrix expressed in the following form.

```
Parameter correlation matrix
 1.0000
-0.8758  1.0000
-0.5964  0.8996  1.0000
-0.8480  0.9485  0.8200  1.0000
```

Clearly, the algorithm to count the number of columns per row in order to insert hashtags will fail because all the rows have different numbers of columns, unless editing is performed like this.

```
Parameter correlation matrix
 1.0000  ...     ...     ...
-0.8758  1.0000  ...     ...
-0.5964  0.8996  1.0000  ...
-0.8480  0.9485  0.8200  1.0000
```

Now transformation would be possible leading to a table such as the following.

Parameter correlation matrix			
1			
-0.8758	1		
-0.5964	0.8996	1	
-0.8480	0.9485	0.8200	1

Another example to consider is from correlation analysis which leads to r values in the strict upper triangle and significance levels in the strict lower triangle as shown next followed by the extracted table.

```
Pearson correlation results
Upper triangle = r, Lower = corresponding two-tail p values
 .....  0.5295  0.2874  0.0662  0.1941  0.6255 -0.5876  0.3010
0.0766   .....  0.3285 -0.0219  0.7930  0.5338 -0.4230  0.3006
0.3650  0.2971   .....  -0.2833  0.2165  0.0264  0.2314 -0.0304
0.8381  1.0000  1.0000   .....  0.2787  0.2837 -0.5238 -0.1166
0.5455  0.0021  0.4992  0.3804   .....  0.2029 -0.1949  0.2144
0.0296  0.0738  0.9351  1.0000  0.5271   ..... -0.4532  0.1360
1.0000  1.0000  0.4694  1.0000  1.0000  1.0000   ..... -0.1696
0.3418  0.3424  1.0000  1.0000  0.5035  0.6735  1.0000   .....
Test for absence of any significant correlations
H0: correlation matrix is the identity matrix
Determinant          =  2.476E-03
Test statistic (TS) =  4.501E+01
Degrees of freedom   =      28
P(chi-sq >= TS)      =  0.0220 Reject H0 at 5% sig.level
```

Pearson correlation results
Upper triangle = r, Lower = corresponding two-tail p values

.....	0.5295	0.2874	0.0662	0.1941	0.6255	-0.5876	0.3010
0.0766	.....	0.3285	-0.0219	0.7930	0.5338	-0.4230	0.3006
0.3650	0.2971	.....	-0.2833	0.2165	0.0264	0.2314	-0.0304
0.8381	1.0000	1.0000	.....	0.2787	-0.2837	-0.5238	-0.1166
0.5455	0.0021	0.4992	0.3804	.....	0.2029	-0.1949	0.2144
0.0296	0.0738	0.9351	1.0000	0.5271	.....	-0.4532	0.1360
1.0000	1.0000	0.4694	1.0000	1.0000	1.0000	.....	-0.1696
0.3418	0.3424	1.0000	1.0000	0.5035	0.6735	1.0000	.....

$$\begin{aligned}
\text{Test} &: \text{ for absence of any significant correlations}\\
H_0 &: \text{ correlation matrix is the identity matrix}\\
\text{Determinant} &= 0.002476\\
\text{Test statistic}(TS) &= 45.01\\
\text{Degrees of freedom} &= 28\\
P(\chi^2 \geq TS) &= 0.0220 \text{ Reject } H_0 \text{ at } 5\% \text{ significance level}
\end{aligned}$$

Here the five dots (.....) denote that the diagonal elements have no meaning and this is just a convenient way to conserve space by having one matrix instead of two. Note that the option to blank out three dots (...) used as temporary column separators does not blank out groups with less three or more than three contiguous dots.

A summary of the options available and procedure to be used comes next.

E.7.7 Summary

- The first step is to select just one table from the results file.
- This can be followed by an optional pre-processing step to edit the table so that every row has exactly the same number of columns.
- Empty cells must be denoted by a three dot ellipsis (...) and cells containing multiple words must have them joined by underscores or similar.
- Three dot symbols to denote empty cells are deleted from the output file.
- The input table can have optional header and trailer sections if required, but these must be highlighted by the control to select headers and trailers.
- There is an option to transform scientific numbers into floating point format by specifying the number of padding zeros required. This option can be switched off.
- Numbers in the header and trailer will also be transformed in this way.
- If it is required, decimal points in floating point numbers can be replaced by commas.
- If transformation fails then the option for pre-processing should be switched on, and also the hashtag table should be requested. By viewing the hashtag table most errors can easily be diagnosed, then rectified by a re-run using pre-process editing.
- If tabbed-text output is selected the resulting file will have to be input into a spreadsheet program for formatting before importing into a word processing program.
- Both html and xml output can allow a certain number of further changes, like changing alpha into α, or adding cell borders.
- LaTeX output will have question marks (?) inserted to replace forbidden character such as underscores which must be edited retrospectively depending the intention, e.g., linking words, or denoting subscripts.

Three further things should be emphasized.

1. Some tables have specialized features such as lined up equals signs that allow multiple words in a column and, as long as every row in the table has an equals sign in exactly the same position, this feature will be recognized.
2. Some SIMFIT results files output tables to the display without three dot separators (...) to create a more pleasing effect, but add them to the results files to assist the processing described in this document.
3. There are several widely used tables that can have empty cells and multi-word titles that the parsing routine will recognize and format automatically.

Finally, should you require further worked examples, you can browse the SIMFIT tutorials, or the document `w_examples.pdf`, where a large number of alternative display styles are demonstrated.

E.8 Printing and importing SimFIT graphs into documents

SimFIT will create high quality graphics files without any other software packages. However, if the free GhostScript package is installed then, not only does this greatly increase the quality of output available, but also many other advanced graphical options are then available. This is because the primary high level graphics format created by SimFIT is the Encapsulated PostScript vector format, i.e., .eps files but, as not all applications can use these directly, you may have to create graphic image files in other formats. However, if you cannot benefit directly from the superior quality of such .eps files, you should never import SimFIT graphics files into documents in .bmp, .pcx, .jpg, or .tif formats; only use .emf or svg exported from SimFIT using the [Win] button, or better .png, or pdf exported using the [PS] button then generated from .eps using GhostScript should be used.

E.8.1 Graphics hardcopy

Any graph displayed by SimFIT can be used directly to drive printers using a high resolution bitmap. The resolution and size can be adjusted but this is only useful for a quick record.

If you have a PostScript printer, then hardcopy driven directly from the display will be of very high quality. If you do not have a PostScript printer, then install Ghostscript and Gsview and similar print quality can be obtained on any printer.

E.8.1.1 Bitmaps and compressed bitmaps

You should never choose to save .bmp, .jpg, .tif, .pcx from the display unless you have a good reason for being satisfied with such large, poor resolution files.

E.8.1.2 Enhanced metafiles (.emf)

Probably the easiest way to use SimFIT graphs in Windows is to save Enhanced Metafiles, i.e., .emf files directly from SimFIT. The quality of .emf files can be improved somewhat by configuring SimFIT to use slightly thicker lines and bolder fonts than the default settings, which you can investigate. If you do not have a PostScript printer, and do not have SimFIT configured to use GSview and Ghostscript, then this is the only course of action open to you, and you will miss out on a great many sophisticated SimFIT plotting techniques.

A further undesirable feature of using .emf files is that they can too easily have their aspect ratio changed within Windows programs, leading to ugly fonts.

E.8.1.3 Scalable vector graphics (.svg)

The W3C recommended format for web line graphics is now the XML based SVG format, and many Windows applications can import these. As SimFIT graphs are line drawings and do not need to save pixel information for the background or filling symbols, then .svg files are an alternative to .png, and have many of the advantages of .eps files.

E.8.2 PostScript graphics files (.eps)

The advantage of storing SimFIT encapsulated PostScript graphics files is that they are very compact, can be printed at any resolution with any number of colors, and SimFIT has many facilities for editing, re-sizing, rotating, overlaying, or making collages from such .eps files. A unique feature is that SimFIT .eps files have a structured format, so they can easily be edited in a text editor, e.g., Notepad, to change sizes, fonts, line types symbols, colors, labels, etc., retrospectively.

E.8.2.1 Ghostscript generated files

If you have Ghostscript installed, then SimFIT can supply .eps files for transformation into other graphics formats such as .bmp, .jpg, .tif, or especially .pdf, or .png.

E.8.2.2 Portable Document graphics files (.pdf)

If you use Adobe Acrobat, or can import Portable Document Format files, i.e., .pdf files generated from SimFIT .eps files by Ghostscript, into your Windows application, then such true .pdf files are an excellent choice. However, beware of the fact that many applications simply embed bitmaps into PostScript or .pdf files, whereupon all the advantages are lost.

E.8.2.3 Portable network graphics files (.png)

Increasingly, the most versatile format for importing graphic image files into Windows programs, such as Microsoft Word and PowerPoint, or LibreOffice Writer or Impress, is the Portable Network Graphics format, as the compression used in .png files results in smaller files than .bmp or .jpg, and edge resolution is far superior. So the best way for users without PostScript printers to use SimFIT graphs in Windows is to store graphs in .eps format, then create .png files at 72dpi for small applications, like the web, where resolution is not important, but at 300dpi or 600dpi if you wish the graph to be printed or displayed at high resolution.

The industry standard for scientific graphs is no longer .gif, it is .png, as these are free from patent problems and are increasingly being accepted by all applications and all operating systems.

E.8.3 Using Encapsulated PostScript (.eps) files directly

If you have a PostScript printer, you can import SimFIT .eps files into Word, but Word may add a low resolution preview, so that what you see in the display may not be what you get on printing. A Word document containing .eps files will print SimFIT graphs at high resolution only on PostScript printers, on non-PostScript printers the resolution may be poor and the graph may not be printed correctly. For this reason, the best way is to save .eps files then create .png or .svg files, either when the .eps file is created or retrospectively. To do this, SimFIT must be configured to use Gsview and Ghostscript, and these can be downloaded from the SimFIT or GSview websites. Note that, for some applications, GSview can add a preview to .eps files in the expectation that this preview will just be used for display and that the PostScript graph will be printed, but not all applications do this correctly. Another advantage of having SimFIT configured to use the GSview package is that your archived .eps files can then be printed as professional quality stand alone graphs at high resolution on any printer.

Appendix F

The SimFIT package

F.1 SimFIT program files

F.1.1 Dynamic Link Libraries

The DLL files must be consistent, that is they must all be compiled and linked together at the same release. If they are upgraded you must replace the whole set, not just one or two of them. If they are not consistent (i.e. not all created at the same time) bizarre effects can result from inconsistent export tables.

numbers.dll
The academic version is a stand-alone library containing the public domain numerical analysis codes required to replace the NAG library routines used by SimFIT. The software included is a selection from: BLAS, Linpack, Lapack, Minpack, Quadpack, Curfit (splines), Dvode and L-BFGS-B. However, the NAG library versions do not contain these numerical analysis codes, but call the NAG library DLLs indirectly through the NAG library version of the maths DLL.

maths.dll
The academic version contains in-line replacement code for the NAG library, and calls to the numbers library to satisfy dependencies. The methods described in the NAG library handbook are used for most of the routines, but some exploit alternative methods described in sources such as AS or ACM TOMS. The NAG library maths versions are mainly front ends to the NAG library DLLs, but they do contain some extra code.

menus.dll
This is the GUI interface to the Windows Win32 API that is responsible for creating the SimFIT menus and all input/output. SimFIT does not use resource scripts and all menus and tables are created on the fly. This DLL consists of a set of subroutines that transform data into a format that can be recognized by the arguments of the winio@(.) integer function of the Salford Software Clearwin Plus Windows Interface. When the calls have been processed, the resulting arguments are passed to the Clearwin interface. So menus is dependent on the clearwin DLL.

graphics.dll
This contains the graphics codes, and is dependent on the menus and clearwin DLLs.

simfit.dll
This consists of all the numerical analysis routines used by SimFIT that are not in numbers.dll or maths.dll. It also contains numerous special routines to check data and advise users of ill-conditioned calculations, unsatisfactory data and so on. It depends on the maths, menus and graphics DLLs.

models.dll
This contains the model subroutines used in simulation and curve fitting. The basic model (Version 2.0) is

rather limited in scope and there are many variations with models dedicated to specific uses. Most of these are consistent with maths.dll and numbers.dll but some use special functions that need enhanced versions of maths.dll. It is possible to upgrade the library of equations and use an enlarged set of user defined functions by upgrading this file alone. It depends on the menus, maths, and numbers DLLs.

clearwin.dll
This contains all the codes that require the Salford-Silverfrost runtime library salflibc.dll. This, and the help DLL, are the only source codes in SimFIT that must be compiled using the FTN95 compiler. Note that some versions of the clearwin DLL are also linked to the graphics DLL for reverse communication.

help.dll
This contains compiled HTML scripts. It depends on the menus DLL and must be compiled using the FTN95 compiler.

salflibc.dll
This contains the Salford-Silverfrost runtime library to interface with the Windows API.

F.1.2 Executables

adderr
This takes in exact data for functions of one, two or three independent variables and adds random error to simulate experimental data. Replicates and outliers can be generated and there is a wide range of choice in the probability density functions used to add noise and the methods to create weighting factors.

average
This takes in x, y data points, calculates means from replicates if required, and generates a trapezoidal model, i.e. a sectional model for straight lines joining adjacent means. This model can then be used to calculate areas or fractions of the data above a threshold level, using extrapolation/interpolation, for any sub-section of the data range.

binomial
This is dedicated to the binomial, trinomial and Poisson distributions. It generates point mass functions, cumulative distributions, critical points, binomial coefficients and their sums and tests if numbers supplied are consistent with a binomial distribution. Estimates of binomial probability values with 95% confidence levels can be calculated by the exact F method or approximate quadratic method, analysis of proportions is carried out and confidence contours for the trinomial distribution parameters can be plotted.

calcurve
This reads in curve-fitting data and creates a cubic spline calibration curve. This can then be used to predict x given y or y given x with 95% confidence levels. There is a wide range of procedures and weighting options that can be used for controlling the data smoothing.

chisqd
This is dedicated to the chi-square distribution. It calculates density and cumulative distribution functions as well as critical points, tests if numbers are consistent with a chi-square distribution, does a chi-square test on paired observed and expected values or on contingency tables and calculates the Fisher exact statistics and chi-square statistics with the Yates correction for 2 by 2 tables.

compare
This fits a weighted least squares spline with user-chosen smoothing factor to data sets. From these best-fit splines the areas, derivatives, absolute curvature and arc length can be estimated and pairs of data sets can be compared for significant differences.

csafit
This is dedicated to estimating the changes in location and dispersion in flow cytometry data so as to express changes in ligand binding or gene expression in terms of estimated parameters.

deqsol
This simulates systems of differential equations. The user can select the method used, range of integration, tolerance parameters, etc. and can plot profiles and phase portraits. The equations, or specified linear combinations of the components can be fitted to data sets.

editfl
This editor is dedicated to editing SimFIT curve fitting files. It has numerous options for fusing and rearranging data sets, changing units of measurement and weighting factors, plotting data and checking for inconsistencies.

editmt
This is a general purpose numerical editor designed to edit SimFIT statistical and plotting data files. It has a large number of functions for cutting and pasting, rearranging and performing arithmetical calculations with selected rows and columns. This program and EDITFL are linked into all executables for interactive editing.

editps
This editor is specifically designed to edit PostScript files. It can change dimensions, rotation, titles, text, etc. as well as overprinting files to form insets or overlays and can group PostScript files together to form collages.

eoqsol
This item is for users who wish to study the effect of spacing and distribution of data points for optimal design in model discrimination.

exfit
This fits sequences of exponential functions and calculates best fit parameters and areas under curves. It is most useful in the field of pharmacokinetics.

ftest
This is dedicated to the F distribution. It calculates test statistics, performs tests for consistency with the F distribution and does the F test for excess variance.

gcfit
This can be run in three modes. In mode 1 it fits sequences of growth curves and calculates best-fit parameters such as maximal growth rates. In mode 2 it fits survival models to survival data. In mode 3 it analyzes censored survival data by generating a Kaplan-Meier nonparametric survival estimate, finding maximum likelihood Weibull models and performing Cox analysis.

help
This item provides on-line help to SimFIT users.

hlfit
This is dedicated to analyzing ligand binding data due to mixtures of high and low affinity binding sites where the response is proportional to the percentage of sites occupied plus a background constant level. It is most useful with dose response data.

inrate
This finds initial rates, lag times, horizontal or inclined asymptotes using a selection of models. It is most useful in enzyme kinetics and transport studies.

linfit
This does multi-linear regression and a provides a variety of linear regression techniques such as overde-

termined L1 fitting, generalized linear interactive modelling, orthogonal fitting, robust regression, principal components, etc.

makcsa
This simulates flow cytometry data for testing program CSAFIT.

makdat
This can generate exact data for functions of one, two or three independent variables, differential equations or user-defined equations. It can also create two and three dimensional plots.

makfil
This is designed to facilitate the preparation of data sets for curve fitting. It has many features to make sure that the user prepares a sensible well-scaled and consistent data file and is also a very useful simple plotting program.

maklib
This collects SIMFIT data files into sets, called library files, to facilitate supplying large data sets for fitting, statistical analysis or plotting.

makmat
This facilitates the preparation of data files for statistical analysis and plotting.

maksim
This takes in tables with columns of data from data base and spread sheet programs and allows the user to create SIMFIT files with selected sub-sets of data, e.g. blood pressure for all males aged between forty and seventy.

mmfit
This fits sequences of Michaelis-Menten functions. It is most useful in enzyme kinetics, especially if two or more isoenzymes are suspected.

normal
This is dedicated to the normal distribution. It calculates all the usual normal statistics and tests if numbers are consistent with the normal distribution.

polnom
This fits all polynomials up to degree six and gives the user all the necessary statistics for choosing the best-fit curve for use in predicting x given y and y given x with 95% confidence limits.

qnfit
This is a very advanced curve-fitting program where the models can be supplied by the user or taken from a library, and the optimization procedures and parameter limits are under the user's control. The best fit curves can be used for calibration, or estimating derivatives and areas. Best-fit surfaces can be plotted as sections through the surface and the objective function can be visualized as a function of any two estimated parameters.

rannum
This generates pseudo random numbers and random walks from chosen distributions.

rffit
This performs a random search, constrained overdetermined L1 norm fit then a quasi-Newton optimization to find the best-fit positive rational function. It is most useful in enzyme kinetics to explore deviations from Michaelis-Menten kinetics.

rstest
This does runs and signs tests for randomness plus a number of nonparametric tests.

run5
This program-manager runs the SimFIT package. The executable is called w_simfit.exe.

sffit
This is used for fitting saturation curves when cooperative ligand binding is encountered. It gives binding constants according to all the alternative conventions and estimates Hill plot extremes and zeros of the binding polynomial and its Hessian.

simplot
This takes in ASCII coordinate files and creates plots, bar charts, pie charts, surfaces and space curves. The user has a wealth of editing options to create publication quality hardcopy.

simstat
This describes the SimFIT statistics options and does all the usual tests. In addition it does numerous statistical calculations such as zeros of polynomials, determinants, eigenvalues, singular value decompositions, time series, power function estimations, etc.

spline
This utility takes in spline coefficients from best-fit splines generated by CALCURVE and COMPARE and uses them for plotting and calculating areas, derivatives, arc lengths and curvatures.

ttest
This is dedicated to the t statistic. It calculates densities and critical values, tests if numbers are consistent with the t distribution, and does t and paired t tests after testing for normality and doing a variance ratio test.

usermod
This utility is used to develop user defined models. It also plots functions, estimates areas by adaptive quadrature and locates zeros of user defined functions and systems of simultaneous nonlinear equations.

change_simfit_version
This program can be used at any time to transform the current version of SimFIT into one of the alternative versions. For instance transforming the academic version into a NAG library version. It does this by overwriting the current copies of the maths and number DLLs, so it is important that SimFIT is not in use when this utility is executed.

F.2 SimFIT data files

To use SimFIT you must first collect your data into tables then create data files. These are ASCII text files with a two-line header section, then a table consisting of your m by n matrix of data values, followed, if required, by an optional trailer section, as follows.

```
Title                        :Header line 1
m   n                        :Header line 2 (no. of rows and columns)
a(1,1) a(1,2) ... a(1,n)     :Data row 1 ...
a(2,1) a(2,2) ... a(2,n)     :Data row 2 ...
...
a(m,1) a(m,2) ... a(m,n)     :Data row m ...
k                            :Trailer line 1 (no. of appended lines)
appended line 1              :Trailer line 2
appended line 2              :Trailer line 3
...
appended line k              :Trailer line k + 1
```

There are special SIMFIT programs to prepare and edit such files: **makfil** or **editfl** for curve fitting data, and **makmat** or **editmt** for statistics. These editors have a very extensive range of functions. However, if you already have data in a spread sheet or data base program like Excel, you can copy selected tables to the clipboard and paste them directly into SIMFIT or into the special program **maksim** which allows you to select rows and columns to be written to a data file in the SIMFIT format.

The next examples are SIMFIT test files chosen to illustrate some features of the SIMFIT data file format. Although all data files have an identical format, the meaning of the rows and columns must be consistent with the use intended. For instance, column 1 must be in nondecreasing order for curve fitting or MANOVA data sets, and this is best appreciated by viewing the test files designated as defaults for each procedure. Note that SIMFIT does not differentiate between floating point numbers and integers in the matrix table section of data files, and either spaces or commas can be used a separators. The title is quite arbitrary, but the row and column dimensions must be exact. The trailer can be omitted altogether, or may be used to store further items, such as labels, indicators, values, or starting estimates and limits. A particularly valuable feature of the trailer section is that there is a begin{. . . } . . . end{. . . } format which can be used to append labels, indicators, starting values, or parameter limits to the end of data files.

F.2.1 Example 1: a vector

Table F.1 illustrates a typical vector file (`vector.tf1`), with title, row and column dimensions, 1 column of data, then an unused trailer section.

```
Vector with components 1, 2, 3, 4, 5
     5     1
   1.0
   2.0
   3.0
   4.0
   5.0
     1
Default line
```

Table F.1: Test file vector.tf1

F.2.2 Example 2: a matrix

Table F.2 illustrates a typical matrix file (`matrix.tf1`), with 5 rows and 5 columns.

F.2.3 Example 3: an integer matrix

Table F.3 has integer entries, and the trailer is being used to describe the meaning of the columns of observations in the test file `binomial.tf3`.

```
Arbitrary 5 by 5 matrix
     5     5
   1.2   4.5   6.1   7.2   8.0
   3.0   5.6   3.7   9.1  12.5
  17.1  23.4   5.5   9.2   3.3
  7.15  5.87  9.94  8.82  1.08
  12.4   4.3   7.7  8.95   1.6
     1
Default line
```

Table F.2: Test file matrix.tf1

```
y,N,x for analysis of proportions
   5   3
  23   84 1
  12   78 2
  31  111 3
  65   92 4
  71   93 5
3
Column 1 = y, number of successes
Column 2 = N, number of Bernoulli trials, N >= y > = 0
Column 3 = x, x-coordinates for plotting
```

Table F.3: Test file binomial.tf3

F.2.4 Example 4: appending labels

Table F.4 originally had the first 12 lines of the trailer for row labels to be plotted corresponding to the 12 rows (i.e., cases in the test file `cluster.tf1`). To use this method, the labels must be the first lines in the trailer section, and there must be at least as many labels as there are columns. Where both row and column labels are required, e.g. with biplots, column labels follow on from row labels. The current version of `cluster.tf1` now uses a more versatile method to add labels which will be described next.

F.2.5 Example 5: using begin ... end to add labels

Table F.5 illustrates the alternative method for adding 12 labels to the trailer section for the 12 rows in test file `piechart.tf1`..

F.2.6 Example 6: various uses of begin ... end

Table F.6 illustrates how to append starting clusters, indicator variables to include or suppress covariates, as well as labels to a data file. When such begin{. . . } . . . end{. . . } techniques are used, the appended data can be placed anywhere in the trailer section. Note that 17 rows and 17 labels have been suppressed to make the table compact, but the complete test file is `kmeans.tf1`.

F.2.7 Example 7: starting estimates and parameter limits

The advanced curve-fitting program **qnfit** must have starting estimates and parameter limits appended to the data file in order to use expert mode model fitting. This example shows how this is done using begin{limits} . . . end{limits} with the test file `gauss3.tf1`, where the model has 10 parameters. Note that each line representing a limit triple must satisfy

[botom limit] $\le$ [starting estimate] $\le$ [upper limit]

```
Cluster analysis data, e.g. dendrogram
    12   8
 1   4   2  11   6   4   3   9
 8   5   1  14  19   7  13  21
...
15  21   8   7  17  12   4  22
    18
A-1
B-2
C-3
D-4
E-5
F-6
G-7
H-8
I-9
J-10
K-11
L-12
...
```

Table F.4: Test file cluster.tf1 (original version)

```
Advanced pie chart 1:  fill styles
   10      4
  1.0  1.0 0.0 15
  1.0  2.0 0.0 15
  ...
  1.0 10.0 0.0 15
    12
begin{labels}
Style 1
Style 2
...
Style 10
end{labels}
```

Table F.5: Test file piechart.tf1

i.e., be in nondecreasing order, otherwise the parameter limits will not be acceptable. Test file `gauss3.tf2` illustrates the alternative way to supply these 10 sets of lower limits, starting estimates, and upper limits for the 10 parameters to be estimated.

F.3 SIMFIT auxiliary files

The test files consist of data sets that can be used to understand how SIMFIT works. You can use a test file with a program, then view it to appreciate the format before running your own data. Library files are just collections of names of test files so you can enter many files at the same time. This is very useful with statistics (e.g., ANOVA, multiple comparisons with **simstat**) and plotting (e.g., supplying ASCII coordinate files to **simplot**). Configuration and default files are used by SIMFIT to store certain parameter values that are relevant to some particular functions. Some files are created automatically and upgraded whenever you make

```
Data for 5 variables on 20 soils (G03EFF, Kendall and Stuart)
20 5
77.3 13.0  9.7 1.5 6.4
82.5 10.0  7.5 1.5 6.5
...
69.7 20.7  9.6 3.1 5.9
...
The next line defines the starting clusters for k = 3
begin{values} <-- token to flag start of appended values
82.5 10.0  7.5 1.5 6.5
47.8 36.5 15.7 2.3 7.2
67.2 22.7 10.1 3.3 6.2
end{values}
The next line defines the variables as 1 = include, 0 = suppress
begin{indicators} <-- token to flag start of indicators
1 1 1 1
end{indicators}
The next line defines the row labels for plotting
begin{labels} <-- token to flag start of row labels
A
B
...
T
end{labels}
```

Table F.6: Test file kmeans.tf1

```
QNFIT EXPERT mode file: 3 Gaussians plus 7.5% relative error
   150     3
  -3.0  0.0041947  0.00085276
  -3.0   0.005899  0.00085276
  ...          ...          ...
  15.0   0.029515   0.0024596
    27
...
begin{limits}
  0  1  2
  0  1  2
  0  1  2
 -2  0  2
  2  4  6
  8 10 12
0.1  1  2
  1  2  3
  2  3  4
  0  0  0
end{limits}
```

Table F.7: Test file gauss3.tf1

significant changes, and some are created only on demand. All such configuration and default files are ASCII text files that can be browsed in the SimFIT viewer. In general, the idea is that when a particular configuration proves satisfactory you make the file read-only to fix the current defaults and prevent SimFIT from altering the settings. SimFIT generates many temporary files and if you exit from a program in an abnormal fashion (e.g., by Ctrl+Alt+Del) these are left in an unfinished state. Usually these would be automatically deleted, but expert users will sometimes want the facility to save temporary files on exit from SimFIT, so this possibility is provided. You should not attempt to edit such files in a text editor but note that, if you suspect a fault may be due to a faulty configuration or default files, just delete them and SimFIT will create new versions.

F.3.1 Test files (Data)

adderr.tf1	Data for adding random numbers using **adderr**
adderr.tf2	Data for adding random numbers using **adderr**
anova1.tf1	Matrix for 1 way analysis of variance in **ftest** or **simstat**
anova2.tf1	Matrix for 2 way analysis of variance in **ftest** or **simstat**
anova2.tf2	Matrix for 2 way analysis of variance in **ftest** or **simstat**
anova3.tf1	Matrix for 3 way analysis of variance in **ftest** or **simstat**
anova4.tf1	Matrix for groups/subgroups analysis of variance in **ftest** or **simstat**
anova5.tf1	Matrix for factorial ANOVA (2 factors, 1 block)
anova5.tf2	Matrix for factorial ANOVA (2 factors, 3 blocks)
anova5.tf3	Matrix for factorial ANOVA (3 factors, 1 blocks)
anova5.tf4	Matrix for factorial ANOVA (3 factors, 3 blocks)
anova6.tf1	Matrix for repeated measures ANOVA (5 subjects, 4 treatments)
average.tf1	Data for program **average**
barchart.tf1	Creates a barchart in **simplot**
barchart.tf2	Creates a barchart in **simplot**
barchart.tf3	Creates a barchart in **simplot**
barchart.tf4	Creates a barchart in **simplot**
barchart.tf5	Creates a barchart in **simplot**
barchart.tf6	Creates a barchart in **simplot**
barchart.tf7	Adds a curve to barchart created from barchart.tf6
barcht3d.tf1	Creates a 3 dimensional barchart in **simplot**
barcht3d.tf2	Creates a 3 dimensional barchart in **simplot**
barcht3d.tf3	Creates a 3 dimensional barchart in **simplot**
beta.tf1	Fifty numbers from a beta distribution with $A = 2.0, B = 2.0$
binomial.tf1	Fifty numbers from a binomial distribution with $N = 50, p = 0.5$
binomial.tf2	Analysis of proportions with no effector values, i.e. X, N
binomial.tf3	Analysis of proportions with effector values, i.e. X, N, t
binomial.tf4	Analysis of proportions with effector values, i.e. X, N, t
binomial.tf5	Analysis of proportions with effector values, i.e. X, N, t
bivariate.tf1	Data for fitting a scaled bivariate normal model
calcurve.tf1	Prepares a calibration curve in EXPERT mode using **calcurve**
calcurve.tf2	Calcurve.tf1 with no weights or EXPERT mode settings
calcurve.tf3	Predicts x given y with calcurve.tf1
chisqd.tf1	Fifty numbers from a chi-square distribution with $\nu = 10$
chisqd.tf2	Vector of observed values to be used with chisqd.tf3
chisqd.tf3	Vector of expected values to be used with chisqd.tf2
chisqd.tf4	Matrix for Fisher exact test in **chisqd** or **simstat**
chisqd.tf5	Contingency table for chi-square test in **chisqd** or **simstat**
cluster.tf1	Data for multivariate cluster analysis in **simstat**
cluster.tf2	Data for multivariate cluster analysis in **simstat**
cochranq.tf1	Matrix for Cochran Q test
column1.tf1	Vector for 1 way ANOVA in **ftest** or **simstat**
column1.tf2	Vector for 1 way ANOVA in **ftest** or **simstat**
column1.tf3	Vector for 1 way ANOVA in **ftest** or **simstat**
column1.tf4	Vector for 1 way ANOVA in **ftest** or **simstat**

column1.tf5	Vector for 1 way ANOVA in **ftest** or **simstat**
column2.tf1	Vector for nonparametric correlation in **rstest** or **simstat**
column2.tf2	Vector for nonparametric correlation in **rstest** or **simstat**
column2.tf3	Vector for nonparametric correlation in **rstest** or **simstat**
compare.tf1	Use with **compare** to compare with compare.tf2
compare.tf2	Use with **compare** to compare with compare.tf1
consec4.tf1	Used by **deqsol** in consec4.TFL for A in 4 differential equations A=B=C=D
consec4.tf2	Used by **deqsol** in consec4.TFL for B in 4 differential equations A=B=C=D
consec4.tf3	Used by **deqsol** in consec4.TFL for C in 4 differential equations A=B=C=D
consec4.tf4	Used by **deqsol** in consec4.TFL for D in 4 differential equations A=B=C=D
consec5.tf1	Used by **deqsol** in consec4.TFL for A in 5 differential equations A=B=C=D=E
consec5.tf2	Used by **deqsol** in consec4.TFL for B in 5 differential equations A=B=C=D=E
consec5.tf3	Used by **deqsol** in consec4.TFL for C in 5 differential equations A=B=C=D=E
consec5.tf4	Used by **deqsol** in consec4.TFL for D in 5 differential equations A=B=C=D=E
consec5.tf5	Used by **deqsol** in consec4.TFL for E in 5 differential equations A=B=C=D=E
cox.tf1	Survival data for Cox proportional hazards model
cox.tf2	Survival data for Cox proportional hazards model
cox.tf3	Survival data for Cox proportional hazards model
cox.tf4	Survival data for Cox proportional hazards model
csadat.tf1	Example of the preliminary flow cytometry format for **csadat**
csadat.tf2	Example of the preliminary flow cytometry format for **csadat**
csafit.tf1	Geometric type data with 15% stretch for **csafit**
csafit.tf2	Arithmetic type data with 5% translation for **csafit**
csafit.tf3	Mixed type data for **csafit**
deqsol.tf1	Library data for fitting LV1.tf1 and LV2.tf1 by **deqsol**
deqsol.tf2	Library data for fitting LV1.tf1 by **deqsol**
deqsol.tf3	Library data for fitting LV2.tf1 by **deqsol**
diversity.tf1	Shannon-Brillouin indices of diversity for **simstat**
editfl.tf1	Data for editing by **editfl**
editfl.tf2	Data for editing by **editfl**
editfl.tf3	Data for editing by **editfl**
editfl.tf4	Data for editing by **editfl**
editmt.tf1	Data for editing by **editmt**
editmt.tf2	Data for editing by **editmt**
editmt.tf3	Data for editing by **editmt**
errorbar.tf1	Normal error bars (4 columns)
errorbar.tf2	Advanced error bars (6 columns)
exfit.tf1	Exact data for 1 exponential for fitting by **exfit**
exfit.tf2	Random error added to exfit.tf1 by **adderr**
exfit.tf3	Exact data for 2 exponentials for fitting by **exfit**
exfit.tf4	Random error added to exfit.tf3 by **adderr**
exfit.tf5	Exact data for Model 5 in **exfit**
exfit.tf6	Exact data for Model 6 in **exfit**
exfit.tf7	Exact data for concave down exponentials in **exfit**
fdr_bh.tf1	Data for false dicovery rate calculations in **simstat**
ftest.tf1	Fifty numbers from the F distribution with $m = 2, n = 5$
gauss3.tf1	3 Gaussians: starting estimates by begin{limits}...end{limits}
gauss3.tf2	3 Gaussians: starting estimates from start of trailer section
gcfit.tf1	Exact data for model 3 in **gcfit**
gcfit.tf2	Random error added to gcfit.tf1 by **adderr**
gcfit.tf3	Random error added to gcfit.tf1 by **adderr**
gcfit.tf4	Random error added to logistic equation by **adderr**
gcfit.tf5	Gompertz growth data
gcfit.tf6	Gompertz decay data by reversing gcfit.tf5

gcfit.tf7	Logistic decay data by reversing gcfit.tf2
glm.tf1	Normal errors, reciprocal link
glm.tf2	Binomial errors, logistic link
glm.tf3	Poisson errors, log link
glm.tf4	Gamma errors, reciprocal link
gompertz.tf1	Data for **gcfit** in survival mode 2
hlfit.tf1	Exact data for 1 site for fitting by **hlfit**
hlfit.tf2	Random error added to hlfit.tf1 by **adderr**
hlfit.tf3	Exact data for 2 sites for fitting by **hlfit**
hlfit.tf4	Random error added to hlfit.tf3
hotcold.tf1	Data for **mmfit/hlfit/qnfit** in isotope displacement mode
hotel.tf1	Data for Hotelling 1-sample T-square test
houses.tf1	Data for constructing a biplot
incomplete.tf1	Incomplete matrix with missing values (comma-separated)
incomplete.tf2	Incomplete matrix with missing values (semicolon-separated)
incomplete.tf3	Incomplete matrix with missing values (tab-separated)
incomplete.mv1	Copy of incomplete.tf1
incomplete.mv2	Copy of incomplete.tf2
incomplete.mv3	Copy of incomplete.tf3
inhibit.tf1	Data for fitting mixed inhibition as v = f(S,I)
inrate.tf1	Data for models 1 and 2 in **inrate**
inrate.tf2	Data for model 3 in **inrate**
inrate.tf3	Data for model 4 in **inrate**
inrate.tf4	Data for model 5 in **inrate**
latinsq.tf1	Latin square data for 3 way ANOVA in **ftest** or **simstat**
iris.tf1	Iris data for K-means clustering (see manova1.tf5)
iris.tf2	Starting K-means clusters for iris.tf2
kmeans.tf1	Data for K-means cluster analysis
kmeans.tf2	Starting clusters for kmeans.tf1
ld50.tf1	Dose-response data for LD50 by GLM as y,N,x
ld50.tf2	Dose-response data for LD50 by GLM as x,y,N,s
ld50.tf3	Dose-response data for LD50 by GLM as x,N-y,N,s
line.tf1	Straight line data
line.tf2	Straight line data
linfit.tf1	Multilinear regression data for **linfit**
linfit.tf2	Multilinear regression data for **linfit**
linfit.tf3	Weighted linear regression data for **linfit**
logistic.tf1	Data for binary logistic regression
logistic.tf2	Data to predict p after fitting logistic.tf1
logistic.tf3	Data for logistic regression
logistic.tf4	Data for logistic regression
logistic.tf5	Data for logistic regression
loglin.tf1	Data for log-linear contingency table analysis
lognor.tf1	Fifty numbers from a lognormal distribution with $A = 0.0, B = 1.0$
lv1.tf1	Data for $y(1)$ in the Lotka-Volterra differential equations
lv2.tf1	Data for $y(2)$ in the Lotka-Volterra differential equations
maksim.tf1	Matrix for editing by **maksim**
maksim.tf2	Matrix for editing by **maksim**
manova1.tf1	MANOVA data: 3 groups, 2 variables
manova1.tf2	MANOVA data: 3 groups, 2 variables
manova1.tf3	MANOVA data: 2 groups, 5 variables
matrix_a.tf1	30 by 3 Directed correlation data
matrix_b.tf1	30 by 4 Directed correlation data
matrix_p.tf1	3 by 4 Matrix of probabilities

matrix.tf1	5 by 5 matrix for **simstat** in calculation mode
matrix.tf2	7 by 5 matrix for **simstat** in calculation mode
matrix.tf3	Positive-definite symmetric 4 by 4 matrix for **simstat** in calculation mode
matrix.tf4	Symmetric 4 by 4 matrix for **simstat** in calculation mode
matrix.tf5	25 by 4 matrix for **simstat** in correlation mode
mcnemar.tf1	Data for McNemar test
meta.tf1	Data for Cochran-Mantel-Haentzel Meta Analysis test
meta.tf2	Data for Cochran-Mantel-Haentzel Meta Analysis test
meta.tf3	Data for Cochran-Mantel-Haentzel Meta Analysis test
mmfit.tf1	Exact data for 1 Michaelis-Menten isoenzyme in **mmfit**
mmfit.tf2	Random error added to mmfit.tf1 by **adderr**
mmfit.tf3	Exact data for 2 Michaelis Menten isoenzymes in **mmfit**
mmfit.tf4	Random error added to mmfit.tf3 by **adderr**
negexp.tf1	Fifty numbers from a negative exponential distributiom with mean $A = 2.0$
normal.tf1	Fifty numbers from a normal distribution with $\mu = 0, \sigma = 1$
normal.tf2	Fifty numbers from a normal distribution with $\mu = 1, \sigma = 2$
normal.tf3	Fifty numbers from N(-1.5,1) plus fifty from N(1.5,1)
npcorr.tf1	Matrix for nonparametric correlation in **rstest** or **simstat**
pacorr.tf1	Correlation matrix for partial correlation in **simstat**
pabst-einstein.tf1	log10(Pabst-Einstein) equation
piechart.tf1	Creates a piechart in **simplot**
piechart.tf2	Creates a piechart in **simplot**
piechart.tf3	Creates a piechart in **simplot**
plot2.tf1	LHS axis data for double plot in **simplot**
plot2.tf2	LHS axis data for double plot in **simplot**
plot2.tf3	RHS axis data for double plot in **simplot**
pls_x.tf1	Partial least squares X matrix in **simstat**
pls_y.tf1	Partial least squares Y matrix in **simstat**
pls_z.tf1	Partial least squares Z matrix in **simstat**
poisson.tf1	Forty numbers from a Poisson distribution
poisson.tf2	Death from horse kicks in Prussian cavalry
polnom.tf1	Data for a quadratic in **polnom**
polnom.tf2	Predict x given y from polnom.tf1
polnom.tf3	Predict y given x from polnom.tf1
polnom.tf4	Fit after transforming to $x = \log(x), y = \log(y/(1-y))$
primes.tf1	Single prime numbers up to 10000
primes.tf2	Twin prime numbers up to 10000
qnfit_data.tf1	Data for **qnfit** tutorials: Example 1 Quadratic
qnfit_data.tf2	Data for **qnfit** tutorials: Example 2 2D linear
qnfit_data.tf3	data for **qnfit** tutorials: Example 3 3D linear
qnfit_data.tf4	data for **qnfit** tutorials: Example 4 Sum of 2 MM curves
qnfit_data.tf5	data for **qnfit** tutorials: Example 5 B ib A->B->C
qnfit_data.tf6	data for **qnfit** tutorials: Example 6 Sum of 2 normal pdfs
qnfit_data.tf7	data for **qnfit** tutorials: Example 6 Sum of 2 normal cdfs
qnfit.tf1	Quadratic in EXPERT mode for **qnfit**
qnfit.tf2	Reversible Michaelis-Menten data in EXPERT mode for **qnfit**
qnfit.tf3	Linear function of 3 variables in EXPERT mode for **qnfit**
qnfit_ode.tf1	Michaelis-Menten substrate depletion data in EXPERT mode for **qnfit**
qnfit_ode.tf2	Von Bertallanffy growth data in EXPERT mode for **qnfit**
qnfit_ode.tf3	Von Bertallanffy growthdecay data in EXPERT mode for **qnfit**
rainfall.tf1	Rainfall in England and Wales from 1766 to 2015
recurrent.tf1	Data for y(1) in the recurrent epidemic scheme
recurrent.tf1	Data for y(2) in the recurrent epidemic scheme
recurrent.tf1	Data for y(3) in the recurrent epidemic scheme

rffit.tf1	2:2 Rational function data for **rffit**
rffit.tf2	1:2 Rational function data for **rffit**
rffit.tf3	2:2 Rational function data for **rffit**
rffit.tf4	2:3 Rational function data for **rffit**
rffit.tf5	4:4 Rational function data for **rffit** (2 turning points)
rffit.tf6	3:4 Rational function data for **rffit** (3 turning points)
rffit.tf7	rffit6 with triplicates and 7.5% relative error
robust.tf1	Normal.tf1 with 5 outliers
rstest.tf1	Residuals for runs test in **rstest**
sffit.tf1	Exact data for 1 site in **sffit**
sffit.tf2	Random error added to sffit.tf1 by **adderr**
sffit.tf3	Exact data for 2 sites in **sffit**
sffit.tf4	Random error added to sffit.tf3 by **adderr**
simplot.tf1	Error-bar data for **simplot**
simplot.tf2	Best-fit 1:1 to simplot.tf1 for **simplot**
simplot.tf3	Best-fit 2:2 to simplot.tf1 for **simplot**
spiral.tf1	Creates a 3 dimensional curve in **simplot**
spiral.tf2	Creates a 3 dimensional curve in **simplot**
spline.tf1	Spline coefficients for **spline**-e02baf.tf1
spline.tf2	Spline coefficients for **spline**-compare.tf1
spline.tf3	Spline coefficients for **spline**-compare.tf2
strata.tf1	Data for stratified binomial logistic regression
surface.tf1	Creates a surface in **simplot**
surface.tf2	Creates a surface in **simplot**
surface.tf3	Creates a surface in **simplot**
surface.tf4	Creates a surface in **simplot**
survive.tf1	Survival data for **gcfit** in mode 3
survive.tf2	Survival data to pair with survive.tf1
survive.tf3	Survival data for **gcfit** in mode 3
survive.tf4	Survival data to pair with survive.tf3
survive.tf5	Survival data for **gcfit** in mode 3
survive.tf6	Survival data to pair with survive.tf5
swarm.tf1	Data with error in x and y
times.tf1	Data for time series analysis in **simstat**
trinom.tf1	Trinomial contour plots in **binomial**
trinom.tf2	Trinomial contour plots in **binomial**
trinom.tf3	Trinomial contour plots in **binomial**
ttest.tf1	Fifty numbers from a t distribution with $\nu = 10$
ttest.tf2	t test data for **ttest** or **simstat**
ttest.tf3	Data paired with ttest.tf2
ttest.tf4	t test data for **ttest** or **simstat**
ttest.tf5	Data paired with ttest.tf4
ttest.tf6	Data for t test on rows of a matrix
tukeyq.tf1	matrix for ANOVA then Tukey Q test
uniform.tf1	Fifty numbers from uniform distribution with $A = 0.0, B = 1.0$
ukmap.tf1	coordinates for K-means clustering
ukmap.tf2	starting centroids for ukmap.tf2
ukmap.tf3	uk coastal outline coordinates
vector.tf1	Vector (5 by 1) consistent with matrix.tf1
vector.tf2	Vector (7 by 1) consistent with matrix.tf2
vector.tf3	Vector (4 by 1) consistent with matrix.tf3
vfield.tf1	vector field file (4 columns)
vfield.tf2	vector field file (9 columns, i.e. a biplot)
weibull.tf1	Survival data for **gcfit** in mode 2

weibull.tf2	Fifty numbers from a Weibull distribution with $A = 1.0, B = 1.0$
wilcoxon.tf1	Data for Wilcoxon signed ranks paired with wilcoxon.tf2
wilcoxon.tf2	Data for Wilcoxon signed ranks paired with wilcoxon.tf1
zeros.tf1	Zeros of the Riemann zeta function
zigzag.tf1	Zig-zag data to illustrate clipping to boundaries

F.3.2 Library files (Data)

anova1.TFL	1-way ANOVA in **ftest** or **simstat**
consec3.TFL	Data for fitting by **qnfit** using `consec3.mod`
consec4.TFL	Used by **deqsol** for 4 differential equations A=B=C=D
consec5.TFL	Used by **deqsol** for 5 differential equations A=B=C=D=E
convolv3.TFL	Data for fitting by **qnfit** using `convolv3.mod`
deqsol.TFL	Curve fitting data for **deqsol** (Identical to deqsol.tf1)
editps.TFL	PostScript files for EDITPS
epidemic.TFL	Data for fitting epidemic differential equations
images.TFL	PostScript files for EDITPS
inhibit.TFL	Data for plotting mixed inhibition results
npcorr.TFL	Nonparametric correlation data for **rstest** or **simstat**
recurrent.TFL	Data for 3 components in the recurrent epidemic scheme
simfig1.TFL	Creates figure 1 in **simplot**
simfig2.TFL	Creates figure 2 in **simplot**
simfig3.TFL	Creates figure 3 in **simplot**
simfig4.TFL	Creates figure 4 in **simplot**
simplot.TFL	Identical to simfig1.TFL
spiral.TFL	Creates a spiral in **simplot**
qnfit.TFL	Parameter limits library file for **qnfit**
line3.TFL	Data for fitting three lines simultaneously by **qnfit**

F.3.3 Test files (Models in reverse Polish)

camalot.mod	Model for Logarithmic Spiral as used in Camalots
cheby.mod	Model for Chebyshev expansion
consec3.mod	Model for irreversible chemical kinetics A to B to C
convolve.mod	Model for a convolution between an exponential and gamma function
convolv3.mod	Version of `convolve.mod` for all components
dble_exp.mod	Chemical kinetic double exponential model
d01fcf.mod	Model with four variables for integration
ellipse.mod	Model for an ellipse in **makdat/simplot/usermod**
family2d.mod	Two dimensional family of diffusion equations
family3d.mod	Three dimensional family of diffusion equations
helix.mod	Model for a helix in **makdat/simplot/usermod**
if.mod	Model illustrating logical commands
impulse.mod	Model illustrating 5 single impulse functions
line3.mod	Model for 3 lines in **qnfit**
optimum.mod	Model for optimizing Rosenbrock's 2-dimensional test function in **usermod**
periodic.mod	Model illustrating 7 periodic impulse functions
rose.mod	Model for a rose in **makdat/simplot/usermod**
tangent.mod	Tangent to logarithmic spiral defined in camalot.mod
twister.mod	Projection of a space curve onto coordinate planes
updown.mod	Model that swaps definition at a cross-over point
updownup.mod	Model that swaps definition at two cross-over points
user1.mod	Model illustrating arbitrary models
deqmat.tf1	How to transform a system of differential equations

deqmat.tf2	How to transform a system of differential equations
deqmod1.tf1	1 DE Michaelis-Menten substrate depletion
deqmod1.tf2	1 DE Michaelis-Menten product accumulation
deqmod1.tf3	1 DE Generalised substrate depletion
deqmod1.tf4	1 DE Generalised product accumulation
deqmod1.tf5	1 DE Membrane transport corrected for osmosis
deqmod1.tf6	1 DE von Bertalannfy allometric growth
deqmod2.tf1	2 DE Order 2 equation expressed as two equations
deqmod2.tf2	2 DE Lotka-Volterra predator-prey model
deqmod2.tf3	2 DE Competing species ecological model
deqmod3.tf1	3 DE Epidemic model (with Jcobian)
deqmod3.tf2	3 DE Epidemic model (without Jacobian)
deqmod4.tf1	4 DE Comprehensive Michaelis-Menten reversible model
deqpar1.tf1	Parameters for deqmod1.tf1
deqpar1.tf2	Parameters for deqmod1.tf2
deqpar1.tf3	Parameters for deqmod1.tf3
deqpar1.tf4	Parameters for deqmod1.tf4
deqpar1.tf5	Parameters for deqmod1.tf5
deqpar1.tf6	Parameters for deqmod1.tf6
deqpar2.tf1	Parameters for deqmod2.tf1
deqpar2.tf2	Parameters for deqmod2.tf2
deqpar2.tf3	Parameters for deqmod2.tf3
deqpar4.tf1	Parameters for deqmod4.tf1
usermod1.tf1	Function of 1 variable: line
usermod1.tf2	Function of 1 variable: quadratic
usermod1.tf3	Function of 1 variable: cubic
usermod1.tf4	Function of 1 variable: 2:2 rational function
usermod1.tf5	Function of 1 variable: one exponential
usermod1.tf6	Function of 1 variable: two exponentials
usermod1.tf7	Function of 1 variable: normal integral
usermod1.tf8	Function of 1 variable: capillary diffusion
usermod1.tf9	Function of 1 variable: damped simple harmonic motion
usermod2.tf1	Function of 2 variables: linear
usermod3.tf1	Function of 3 variables: linear
usermod4.tf1	Function of 4 variables: integrand for D01FCF
usermodd.tf1	Differential equation for **usermod**
usermodn.tf1	Four functions for plotting by **usermod**
usermodn.tf2	Two functions of 2 variables for **usermod**
usermodn.tf3	Three functions of 3 variables for **usermod**
usermodn.tf4	Nine functions of 9 variables for **usermod**
usermods.tf1	Special functions with one argument
usermods.tf2	Special functions with two arguments
usermods.tf3	Special functions with three arguments
usermodx.tf1	Using a sub-model for function evaluation
usermodx.tf2	Using a sub-model for quadrature
usermodx.tf3	Using a sub-model for root-finding
usermodx.tf4	Using three sub-models for root-finding of an integral
usermodx.tf5	Using a sub-model to evaluate a multiple integral

F.3.4 Test files (Models using expressions)

camalot_e.mod	Model for Logarithmic Spiral as used in Camalots
convolv3_e.mod	Version of `convolve.mod` for all components
deqmod2_e.tf1	Model for 2 differential equations

ellipse_e.mod	Model for an ellipse
gauss3_e.mod	Model for a sum of three Gaussians
helix_e.mod	Model for a helix
line3_e.mod	Model for 3 lines
optimum_e.mod	Model for optimizing Rosenbrock's 2-dimensional test function
qnfit_model.tf1	Model for **qnfit** tutorials: Example 1 Quadratic
qnfit_model.tf2	Model for **qnfit** tutorials: Example 2 2D linear
qnfit_model.tf3	Model for **qnfit** tutorials: Example 3 3D linear
qnfit_model.tf4	Model for **qnfit** tutorials: Example 4 Sum of 2 MM curves
qnfit_model.tf5	Model for **qnfit** tutorials: Example 5 B in A->B->C
qnfit_model.tf6	Model for **qnfit** tutorials: Example 6 Sum of 2 normal pdfs
qnfit_model.tf7	Model for **qnfit** tutorials: Example 7 Sum of 2 normal cdfs
rose_e.mod >Model for a eight-leaved rose	
usermod1_e.tf1	Function of 1 variable: line
usermod1_e.tf2	Function of 1 variable: quadratic
usermod1_e.tf3	Function of 1 variable: cubic
usermod1_e.tf4	Function of 1 variable: 2:2 rational function
usermod1_e.tf5	Function of 1 variable: one exponential
usermod1_e.tf6	Function of 1 variable: two exponentials
usermod1_e.tf7	Function of 1 variable: normal integral
usermod1_e.tf8	Function of 1 variable: capillary diffusion
usermod1_e.tf9	Function of 1 variable: damped simple harmonic motion
usermod2_e.tf1	Function of 2 variables: linear
usermod3_e.tf1	Function of 3 variables: linear
usermod4_e.tf1	Function of 4 variables: integrand for D01FCF
usermodd_e.tf1	Differential equation
usermodn_e.tf1	Four functions for plotting
usermodn_e.tf2	Two functions of 2 variables
usermodn_e.tf3	Three functions of 3 variables
usermodn_e.tf4	Nine functions of 9 variables
usermods_e.tf1	Special functions with one argument
usermods_e.tf2	Special functions with two arguments
usermods_e.tf3	Special functions with three arguments
deqmod1_e.tf1	1 DE Michaelis-Menten substrate depletion
deqmod1_e.tf2	1 DE Michaelis-Menten product accumulation
deqmod1_e.tf3	1 DE Generalised substrate depletion
deqmod1_e.tf4	1 DE Generalised product accumulation
deqmod1_e.tf5	1 DE Membrane transport corrected for osmosis
deqmod1_e.tf6	1 DE von Bertalannfy allometric growth
deqmod2_e.tf1	2 DE Order 2 equation expressed as two equations
deqmod2_e.tf2	2 DE Lotka-Volterra predator-prey model
deqmod2_e.tf3	2 DE Competing species ecological model
deqmod3_e.tf1	3 DE Epidemic model (with Jcobian)
deqmod3_e.tf2	3 DE Epidemic model (without Jacobian)
deqmod3_e.tf3	3 DE Recurrent Epidemic model (with Jacobian)
deqmod4_e.tf1	4 DE Comprehensive Michaelis-Menten reversible model

F.3.5 Miscellaneous data files

cheby.data	Data required by `cheby.mod`
consec3_A.data	Data for component A in `consec3.mod`
consec3_B.data	Data for component B in `consec3.mod`
consec3_C.data	Data for component C in `consec3.mod`

convolv3.data	Data for `convolv3.mod`
inhibit?.data	Data for `inhibit.tfl`
line?.data	line1.data, line2.data and line3.data for line3.tfl
simfig3?.data	Data for `simfig3.tfl`
simfig4?.data	Data for `simfig4.tfl`
y?.data	y1.data, y2.data and y3.data for `epidemic.tfl`

F.3.6 Graphics configuration and metafiles

These files can be created on demand from program **simplot** in order to save plotting parameters from the current plot for subsequent re-use.

logodds.cfg	Configure a logodds plot
logoddsratios.cfg	Configure a logoddsratios plot
metafile.tf1	Metafile for multiple plots
metafile.tf2	Metafile for a double plot
metafile.tf3	Metafile for a barchart
metafile.tf4	Metafile for a piechart
metafile.tf5	Metafile for a labels plot
metafile.tf6	Metafile for a vector field plot
metafile.tf7	Metafile for a biplot
metafile.tf8	Metafile for a dendrogram
metafile.tf9	Metafile for logodds plot
w_simfig1.cfg	Configures **simplot** to use simfig1.tfl
w_simfig2.cfg	Configures **simplot** to use simfig2.tfl
w_simfig3.cfg	Configures **simplot** to use simfig3.tfl
w_simfig4.cfg	Configures **simplot** to use simfig4.tfl

F.3.7 Parameter limits files

These files consist of lowest possible values, starting estimates and highest possible values for parameters used by **qnfit** and **deqsol** for constraining parameters during curve fitting. They are usually referenced by library files such as qnfit.tfl. See, for example, positive.plf, negative.plf and unconstrained.plf.

F.3.8 Error message files

When programs like **deqsol**, **makdat** and **qnfit** start to execute they open special files like w_deqsol.txt and w_qnfit.txt to receive all messages generated during the current curve fitting and solving of differential equations. Advanced SIMFIT users can inspect these files and other files like iterate.txt to get more details about any singularities encountered during iterations. If any serious problems are encountered using **deqsol** or **qnfit**, you can consult the appropriate *.txt file for more information.

F.3.9 PostScript example files

pscodes.ps	PostScript octal codes
psgfragx.ps	Illustrating psfragex.tex/psfragex.ps1
simfig1.ps	Example
simfig2.ps	Example
simfig3.ps	Example
simfig4.ps	Example
simfonts.ps	Standard PostScript fonts
ms_office.ps	Using MS Excel and Word
pspecial.i	Example PS specials (i = 1 to 10)

*.eps	Assorted Encapsulated PostScript files

F.3.10 SimFIT configuration files

These files are created automatically by SimFIT and should not be edited manually unless you know exactly what you are doing, e.g., setting the PostScript color palette.

g_recent.cfg	Recent graphics files
l_simfit.cfg	This stores information for configuring the Linux version
pspecial.cfg	Configuration file for PostScript specials
w_clpbrd.cfg	This holds the last file number x as in clipboard_x.txt
w_colors.cfg	Colors for simple graphs
w_filter.cfg	This contains the current search patterns used to configure the file selection and creation controls
w_fsizes.cfg	This holds the graphics font sizes
w_ftests.cfg	This holds the last NPTS, NPAR, WSSQ values used for F tests
w_graphs.cfg	This holds the graph configuration parameters
w_input.cfg	This holds the last filenames used for data input
w_labels.cfg	This holds the labels and other bar and pie chart details
w_matrix.cfg	This holds the last file number x as in matrix_x.txt
w_modules.cfg	This holds the names of current modules
w_output.cfg	This holds the last filenames used for data output
w_params.cfg	Current parameters for editing data sets
w_pathto.cfg	Paths
w_ps.cfg	This stores all the PostScript configuration details
w_result.cfg	This holds the filename of the latest results file
w_simfit.cfg	This stores all the important details needed to run SimFIT from the program manager w_simfit.exe
w_symbol.cfg	Symbol and Line types
w_vector.cfg	This holds the last file number x as in vector_x.txt
a_recent.cfg	Recently selected project files (all types)
c_recent.cfg	Recently selected project files (covariance matrices)
f_recent.cfg	Recently selected project files (curve fitting)
g_recent.cfg	Recently selected project files (graphics)
m_recent.cfg	Recently selected project files (matrix))
p_recent.cfg	Recently selected project files (PostScript)
v_recent.cfg	Recently selected project files (vector)

F.3.11 Default files

These files save details of changes made to the SimFIT defaults from several programs.

w_labels.cfg	Stores default plotting labels
w_module.cfg	Stores file names of executable modules
w_params.cfg	Stores default editing parameters
w_symbol.cfg	Stores default plotting symbols

F.3.12 Temporary files

These next two files are deleted then re-written during each SimFIT session. You may wish to save them to disk after a session as a permanent record of files analyzed and created.

w_in.tmp	Stores the list of files accessed during the latest SimFIT session
w_out.tmp	Stores the list of files created during the latest SimFIT session

The results log file `f$result.tmp` is created anew each time a program is started that performs calculations, so it overwrites any previous results. You can save results retrospectively either by renaming this file, or else you can configure SimFIT to ask you for a file name instead of creating this particular results file. SimFIT also creates a number of temporary files with names like `f$000008.tmp` which should be deleted. If you have an abnormal exit from SimFIT, the current results file may be such a file and, in such circumstances, you may wish to save it to disk. SimFIT sometimes makes other temporary files, such as `f$simfit.tmp` with the name of the current program, but you can always presume that it is safe to delete any such files

F.3.13 NAG library files (contents of list.nag)

Models	
c05adf.mod	1 function of 1 variable
c05adf_e.mod	1 function of 1 variable
c05nbf.mod	9 functions of 9 variables
c05nbf_e.mod	9 functions of 9 variables
d01ajf.mod	1 function of 1 variable
d01ajf_e.mod	1 function of 1 variable
d01eaf.mod	10 functions of 4 variables
d01eaf_e.mod	10 functions of 4 variables
d01fcf.mod	1 function of 4 variables
d01fcf_e.mod	1 function of 4 variables
e04fyf.mod	1 function of 3 variables
e04fyf_e.mod	1 function of 3 variables
Data	
c02agf.tf1	Zeros of a polynomial
e02adf.tf1	Polynomial data
e02baf.tf1	Data for fixed knot spline fitting
e02baf.tf2	Spline knots and coefficients
e02bef.tf1	Data for automatic knot spline fitting
e04fyf.tf1	Data for curve fitting using e04fyf.mod
f01abf.tf1	Inverse: symposdef matrix
f01blf.tf1	Pseudo inverse of a matrix
f02fdf.tf1	A for Ax = (lambda)Bx
f02fdf.tf2	B for Ax = (lambda)Bx
f02wef.tf1	Singular value decomposition
f02wef.tf2	Singular value decomposition
f03aaf.tf1	Determinant by LU
f03aef.tf1	Determinant by Cholesky
f07fdf.tf1	Cholesky factorisation
f08kff.tf1	Singular value decomposition
f08kff.tf2	Singular value decomposition
g02baf.tf1	Correlation: Pearson
g02bnf.tf1	Correlation: Kendall/Spearman
g02bny.tf1	Partial correlation matrix
g02caf.tf1	Unweighted linear regression
g02daf.tf1	Multiple linear regression
g02gaf.tf1	GLM normal errors
g02gbf.tf1	GLM binomial errors
g02gcf.tf1	GLM Poisson errors
g02gdf.tf1	GLM gamma errors

g02haf.tf1	Robust regression (M-estimates)
g02laf.tf1	Partial Least squares X-predictor data
g02laf.tf2	Partial Least Squares Y-response data
g02laf.tf3	Partial Least Squares Z-predictor data
g02wef.tf1	Singular value decomposition
g02wef.tf2	Singular value decomposition
g03aaf.tf1	Principal components
g03acf.tf1	Canonical variates
g03adf.tf1	Canonical correlation
g03baf.tf1	Matrix for Orthomax/Varimax rotation
g03bcf.tf1	X-matrix for procrustes analysis
g03bcf.tf2	Y-matrix for procrustes analysis
g03caf.tf1	Correlation matrix for factor analysis
g03ccf.tf1	Correlation matrix for factor analysis
g03daf.tf1	Discriminant analysis
g03dbf.tf1	Discriminant analysis
g03dcf.tf1	Discriminant analysis
g03eaf.tf1	Data for distance matrix: calculation
g03ecf.tf1	Data for distance matrix: clustering
g03eff.tf1	K-means clustering
g03eff.tf2	K-means clustering
g03faf.tf1	Distance matrix for classical metric scaling
g03ehf.tf1	Data for distance matrix: dendrogram plot
g03ejf.tf1	Data for distance matrix: cluster indicators
g04adf.tf1	ANOVA
g04aef.tfl	ANOVA library file
g04caf.tf1	ANOVA (factorial)
g07bef.tf1	Weibull fitting
g08acf.tf1	Median test
g08acf.tf2	Median test
g08aef.tf1	ANOVA (Friedman)
g08aff.tfl	ANOVA (Kruskall-Wallis)
g08agf.tf1	Wilcoxon signed ranks test
g08agf.tf2	Wilcoxon signed ranks test
g08ahf.tf1	Mann-Whitney U test
g08ahf.tf2	Mann-Whitney U test
g08baf.tf1	Mood and David dispersion tests
g08baf.tf2	Mood and David dispersion tests
g08cbf.tf1	Kolmogorov-Smirnov 1-sample test
g08daf.tf1	Kendall coefficient of concordance
g08eaf.tf1	1000 U(0,1) psedo-random numbers
g08raf.tf1	Regression on ranks
g08rbf.tf1	Regression on ranks
g10abf.tf1	Data for cross validation spline fitting
g10caf.tf1	Data for T4253H smoothing
g11caf.tf1	Stratified logistic regression
g12aaf.tf1	Survival analysis
g12aaf.tf2	Survival analysis
g12baf.tf1	Cox regression
g13dmf.tf1	Auto- and cross-correlation matrices
j06sbf.tf1	Time series

F.4 Acknowledgements

History of SimFIT

Early on Julian Shindler used analogue computing to simulate enzyme kinetics but SimFIT really started with mainframes, cards and paper tape; jobs were submitted to be picked up later, and visual display was still a dream. James Crabbe and John Kavanagh became interested in computing and several of my colleagues, notably Dennis Waight, used MINUITS for curve fitting. This was an excellent program allowing random searches, Simplex and quasi-Newton optimization; imposing constraints by mapping sections of the real line to $(-\infty, \infty)$. Bob Foster and Tom Sharpe helped us to make this program fit models like rational functions and exponentials, and Adrian Bowman gave valuable statistical advice. By this time we had a mainframe terminal and Richard Woolfson and Jean Pierre Mazat used the NAG library to find zeros of polynomials as part of our collaboration on cooperativity algebra, while Francisco Solano, Paul Leff and Jean Wardell used the NAG library random number generators for simulation. Andrew Wright advanced matters when we started to use the NAG library differential equation solvers and optimization routines to fit differential equations to enzyme kinetic problems, and Mike Pettipher and Ian Gladwell provided hints on how to do this. Phil McGinlay took on the task of developing pharmacokinetic and diffusion models, Manuel Roig joined us to create the optimal design programs, while Naveed Buhkari spent time developing the growth curve fitting models. When the PC came along, Elina Melikhova worked on the flow cytometry analysis programs, Jesus Cachaza helped with improving the goodness of fit and plotting functions, Ralph Ackerman explained the need for a number of extra features, while Robert Burrows and Igor Plesner provided feedback on the development of the user supplied model routines, which were tested by Naveed Prasad.

The Windows and Linux versions of SimFIT

SimFIT has been subjected to much development on numerous platforms, and the latest project has been to write substitutes for NAG routines using the public domain code that is now available for linear algebra, optimization, and differential equation solving, so that there are now two SimFIT versions.

- ❍ The academic version
  This is a completely free version, designed for student use.
- ❍ The professional version
  This has more features than the academic version, but it requires the NAG library DLLs.

Geoff Morgan, Sven Hammarling, David Sayers, and John Holden from NAG have been very helpful here. Also, the move into the Windows environment guided by Steve Bagley and Abdul Sattar, was facilitated by the Clearwin Plus interface provided by Salford Software, helped by their excellent support team, particularly David Bailey, Paul Laidler, Ivan Lucas, and Richard Putman. I thank Mark Ferguson for his interest and support during this phase of the project, and John Clegg for developing the Excel macros. Mikael Widersten was very helpful in the development of the Linux/Wine version, Stephen Langdell suggested numerous improvements which were incorporated at Version 6, Martin Wills helped develop the website, and Samad Bonakdar helped to improve the graphics interface.

Important collaborators

Although I am very grateful for the help that all these people have given I wish to draw attention to a number of people whose continued influence on the project has been of exceptional value. Reg Wood has been an unfailing source of help with technical mathematics, Eos Kyprianou has provided constant advice and criticism on statistical matters, Len Freeman has patiently answered endless questions about optimization software, Keith Indge has been a great source of assistance with computational techniques, David Carlisle showed me how to develop the SimFIT PostScript interface and Francisco Burguillo has patiently tested and given valuable feedback on each revision. Above all, I thank Robert Childs who first persuaded me that mathematics, statistics and computing make a valuable contribution to scientific research and who, in so doing, rescued me from having to do any more laboratory experiments.

Public domain code

SimFIT has a computational base of reliable numerical methods constructed in part from public domain code that is contained in `w_maths.dll` and `w_numbers.dll`. The main source of information when developing these libraries was the comprehensive NAG Fortran library documentation, and the incomparable Handbook of Mathematical Functions (Dover, by M.Abramowitz and I.A.Stegun). Numerical Recipes (Cambridge University Press, by W.H.Press, B.P.Flannery, S.A.Teukolsky, and W.T.Vetterling) was also consulted, and some codes were used from Numerical methods of Statistics (Cambridge University Press, by J.F.Monahan). Editing was necessary for consistency with the rest of SimFIT but I am extremely grateful for the work of the numerical analysts mentioned below because, without their contributions, the SimFIT package could not have been created.

BLAS, LINPACK, LAPACK [Linear algebra]
T.Chen, J.Dongarra, J. Du Croz, I.Duff, S.Hammarling, R.Hanson, R.J.Kincaid, F.T.Krogh, C.L.Lawson, C.Moler, G.W.Stewart, and others.

MINPACK [Unconstrained optimization]
K.E.Hillstrom, B.S.Garbou, J.J.More.

LBFGSB [Constrained optimization]
R.H.Byrd, P.Lu-Chen, J.Nocedal, C.Zhu.

DVODE [Differential equation solving]
P.N.Brown, G.D.Byrne, A.C.Hindmarsh.

SLATEC [Special function evaluation]
D.E.Amos, B.C.Carlson, S.L.Daniel, P.A.Fox, W.Fullerton, A.D.Hall, R.E.Jones, D.K.Kahaner, E.M.Notis, R.L.Pexton, N.L.Schryer, M.K.Weston.

CURFIT [Spline fitting]
P.Dierckx

QUADPACK [Numerical integration]
E.De Doncker-Kapenga, D.K.Kahaner, R.Piessens, C.W.Uberhuber

ACM [Collected algorithms]
487 (J.Durbin), 493 (M.A.Jenkins), 495 (I.Barrodale, C.Philips), 516 (J.W.McKean, T.A.Ryan), 563 (R.H.Bartels, A.R.Conn), 642 (M.F.Hutchinson), 698 (J.Berntsen, A.Genz), 707 (A.Bhalla, W.F.Perger), 723 (W.V.Snyder), 745 (M.Goano), 757 (A.J.McLeod).

AS [Applied statistics]
6 and 7 (M.J.R.Healey), 63 (G.P.Bhattacharjee, K.L.Majumder), 66 (I.D.Hill), 91 (D.J.Best, D.E.Roberts), 94, 177 and 181 (J.P.Royston), 97 (D.M.Munro), 109 (G.W.Cran, K.J.Martin, G.E.Thomas), 111 (J.D.Beasley, S.G.Springer), 136 (J.A.Hartigan, M.A.Wong), 171 (E.L.Frome), 176 (B.W.Silverman) using ASR50 (M.C.Jones and H.W.Lotwick), 190 (R.E.Lund), 196 (M.D.Krailko, M.C.Pike), 226 and 243 (R.V.Lenth), 275 (C.G.Ding), 280 (M.Conlon, R.G.Thomas).

Subroutines
varmx (D.M.Hawkins)

Index