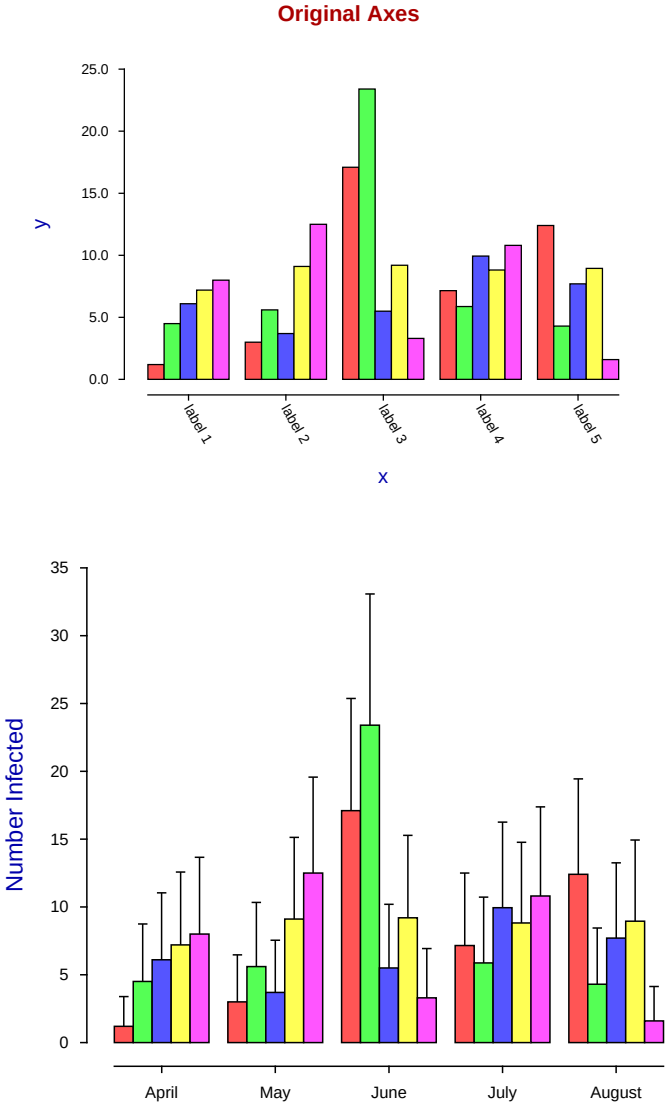
Tutorials and worked examples for simulation, curve fitting, statistical analysis, and plotting.
http://www.simfit.org.uk

Error bars with barcharts

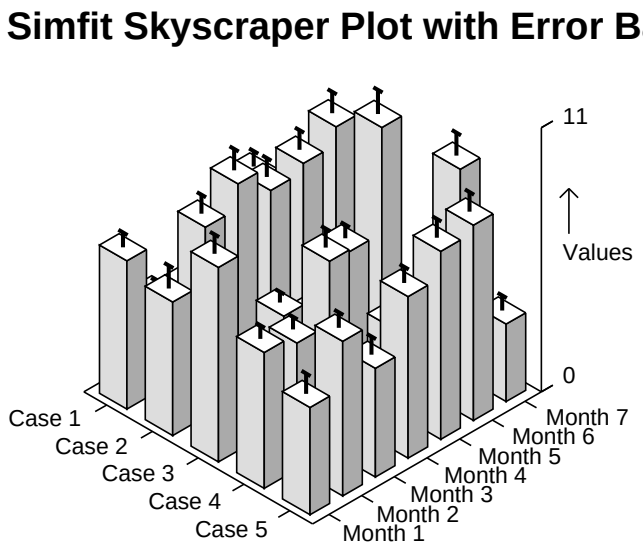
Barcharts with or without error bars can be created interactively from a table of values. For example, the upper figure below was generated from `matrix.tf1` as a default barchart using program **simplot**, while the lower figure has twice the square root of the bar values added.

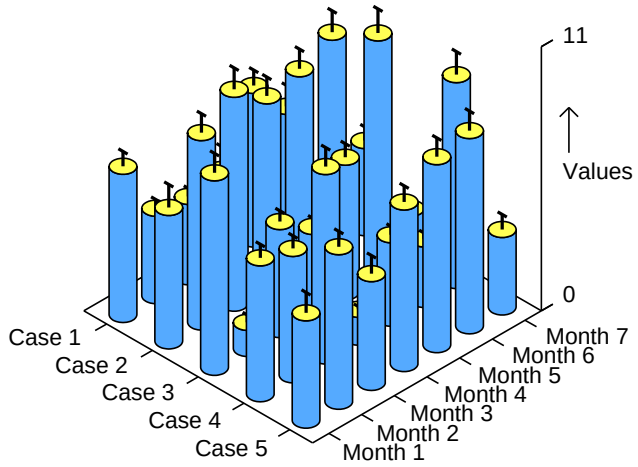


If the elements are measurements, the height of the bars would be means, while error bars would be calculated as 95% confidence limits, i.e., using a t distribution and assuming a normal distribution. Often one standard error of the mean is used instead of confidence limits to make the data look better, which is dishonest. If the elements are counts, approximate error bars could be added as twice the square root of the counts, i.e., assuming a Poisson distribution. Note that after creating barcharts from matrices in program **simplot** the temporary advanced barchart files generated to plot the bar chart can be saved.

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. The next figure shows the plot generated by **simplot** from the test file `matrix.tf2`.





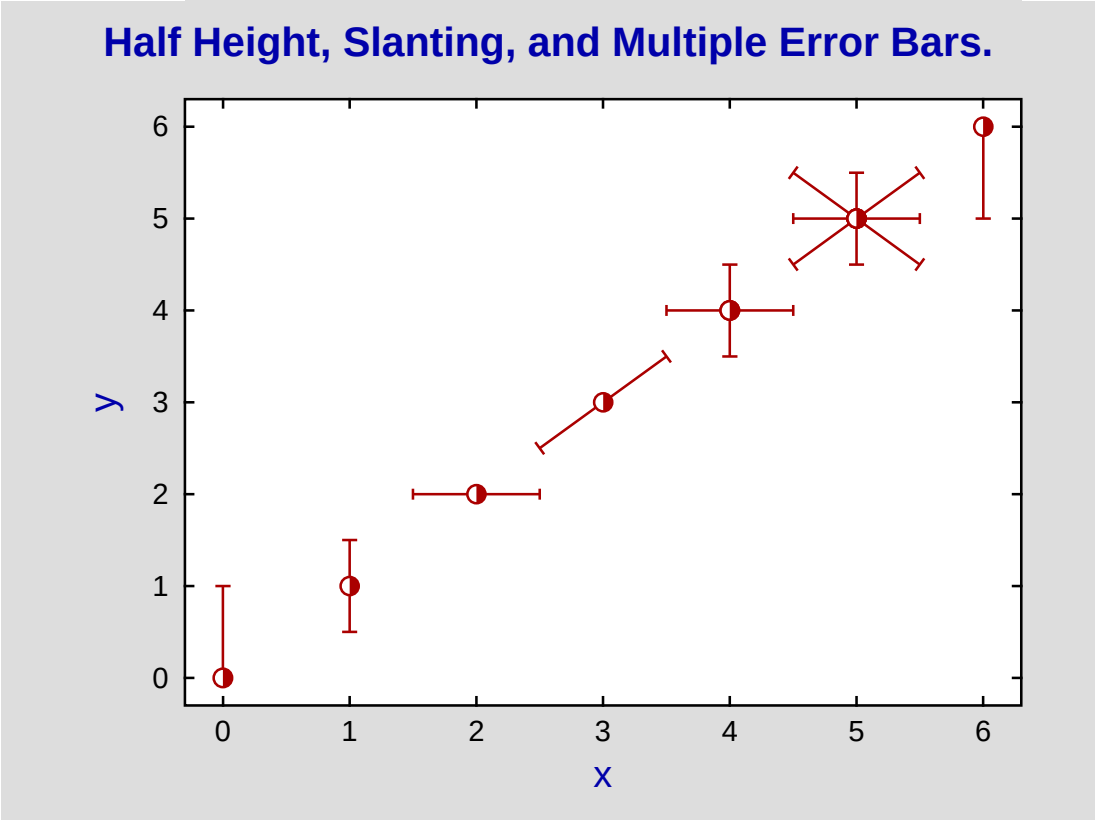
Errors are added from a file, and are calculated according to the distribution assumed. They could be twice square roots for Poisson counts, binomial errors for proportions or percentages, or they could be calculated from sample standard deviations using the t distribution for means. As skyscraper plots 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.

Slanting and multiple error bars

Error bar files can be created by program **editfl** after editing curve fitting files with replicates, and such error bars will be symmetrical, representing central confidence limits in the (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, generate noncentral confidence limits, i.e., unsymmetrical error bars for binomial parameter confidence limits, and Log-Odds plots.

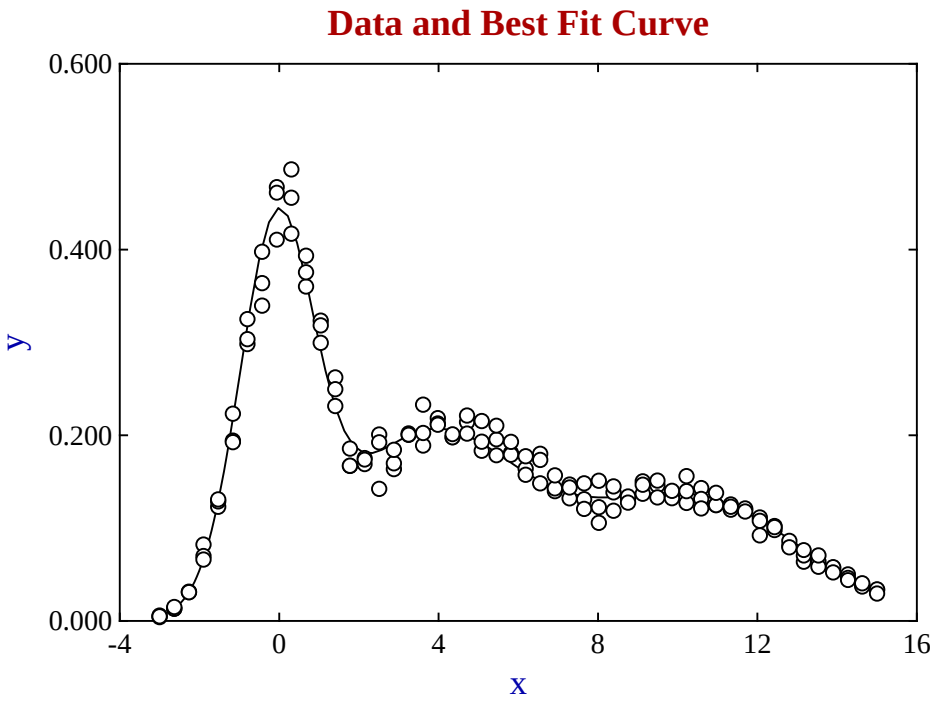
Sometimes it is necessary to plot asymmetrical error bars, slanting error bars or even multiple error bars. To do this, note that the error bar test file `errorbar.tf1` has four columns with the x coordinate for the plotting symbol, then the y-coordinates for the lower bar, middle bar and upper bar. However, the error bar test file `errorbar.tf2` has six columns shown in the table below, 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 the next figure.

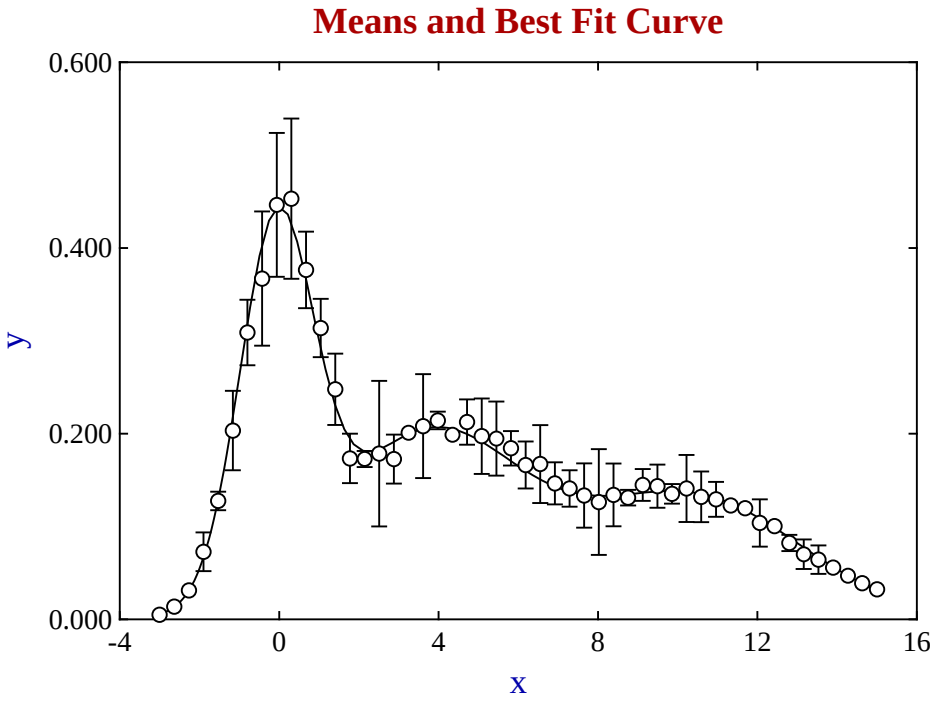
x_1	x_2	x_3	y_1	y_2	y_3
0.0	0.0	0.0	0.0	0.0	1.0
1.0	1.0	1.0	0.5	1.0	1.5
1.5	2.0	2.5	2.0	2.0	2.0
2.5	3.0	3.5	2.5	3.0	3.5
3.5	4.0	4.5	4.0	4.0	4.0
4.0	4.0	4.0	3.5	4.0	4.5
5.0	5.0	5.0	4.5	5.0	5.5
4.5	5.0	5.5	5.0	5.0	5.0
4.5	5.0	5.5	4.5	5.0	5.5
4.5	5.0	5.5	5.5	5.0	4.5
6.0	6.0	6.0	6.0	6.0	5.0



Calculating error bars interactively

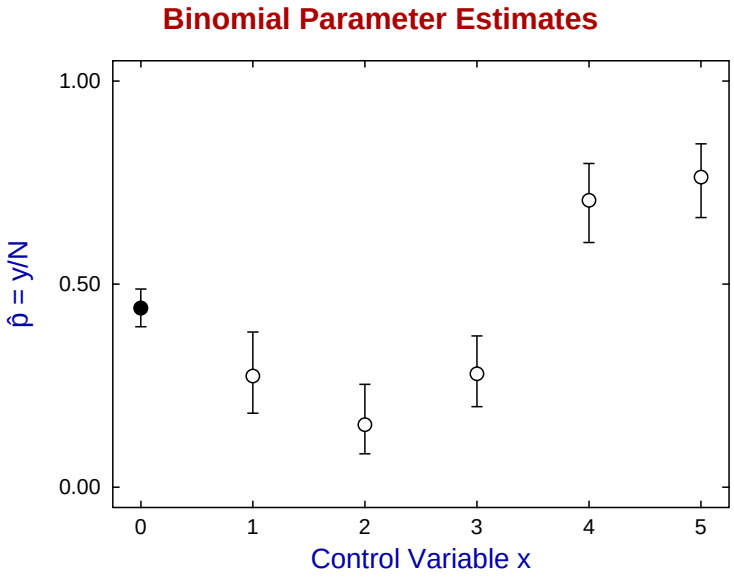
The next figure shows the best fit curve estimated by **qnfit** when fitting a sum of three Gaussians to the test file `gauss3.tf1`. Note that all the data must be used for fitting, not means. Programs **editfl** and **simplot** can generate error bar plotting files from such data files with replicates, as illustrated for 95% confidence limits.





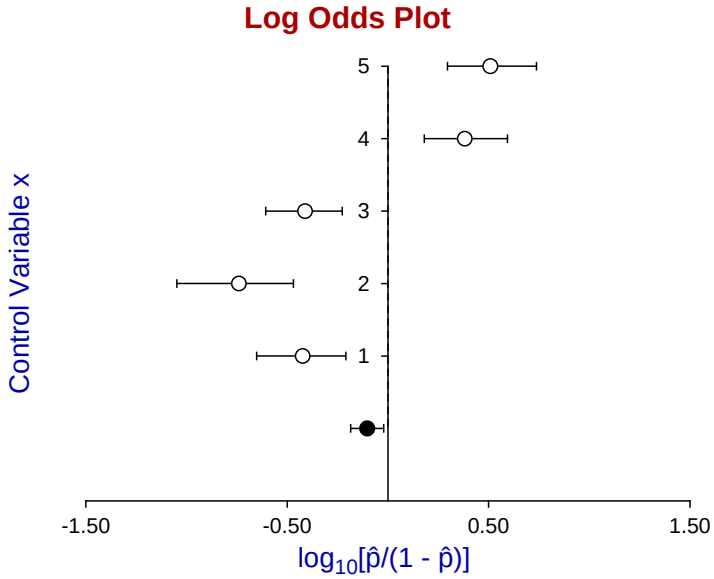
Plotting binomial error bars

The plot below shows binomial parameter estimates for y successes in N trials. The error bars represent exact, unsymmetrical confidence limits, not those calculated using the normal approximation.

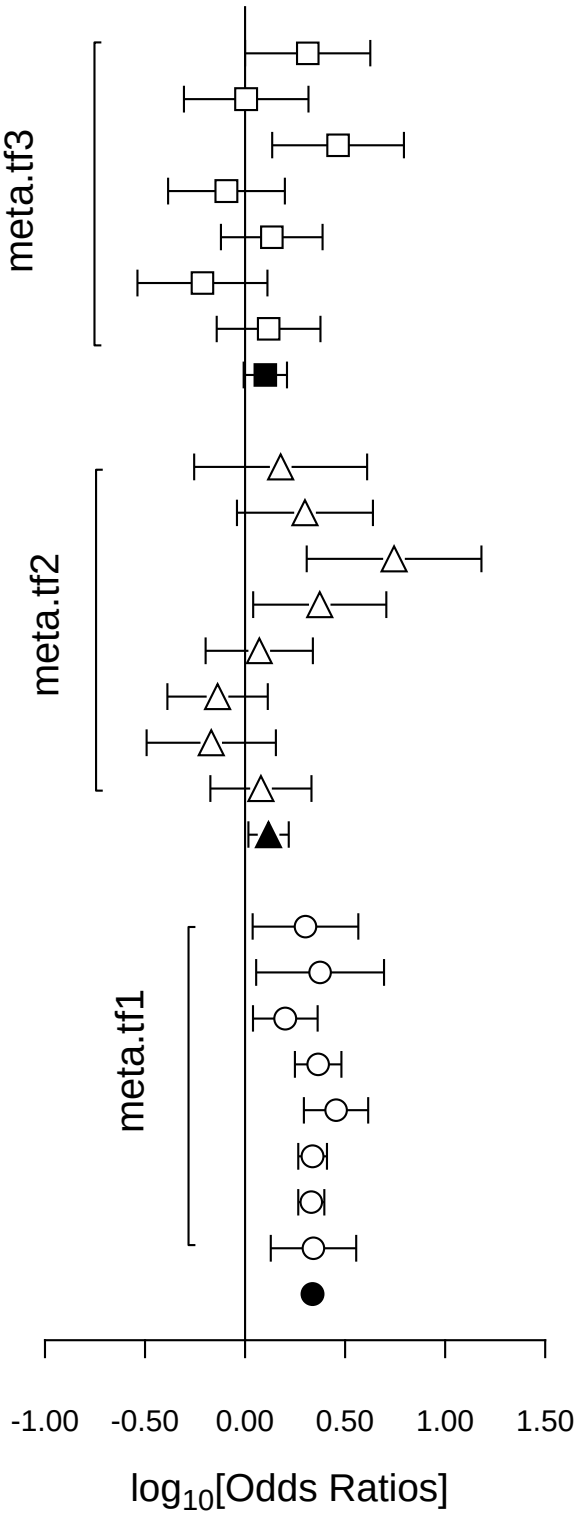


Plotting Log-Odds error bars

Binomial error bars 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 the next figure, to emphasize deviation from a fixed p value, e.g. $p = 0.5$ with a log-odds of 0. This figure 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.



Plotting meta analysis error bars



It is often useful to plot Log-Odds-Ratios, so the creation of the adjacent figure 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. 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, 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.