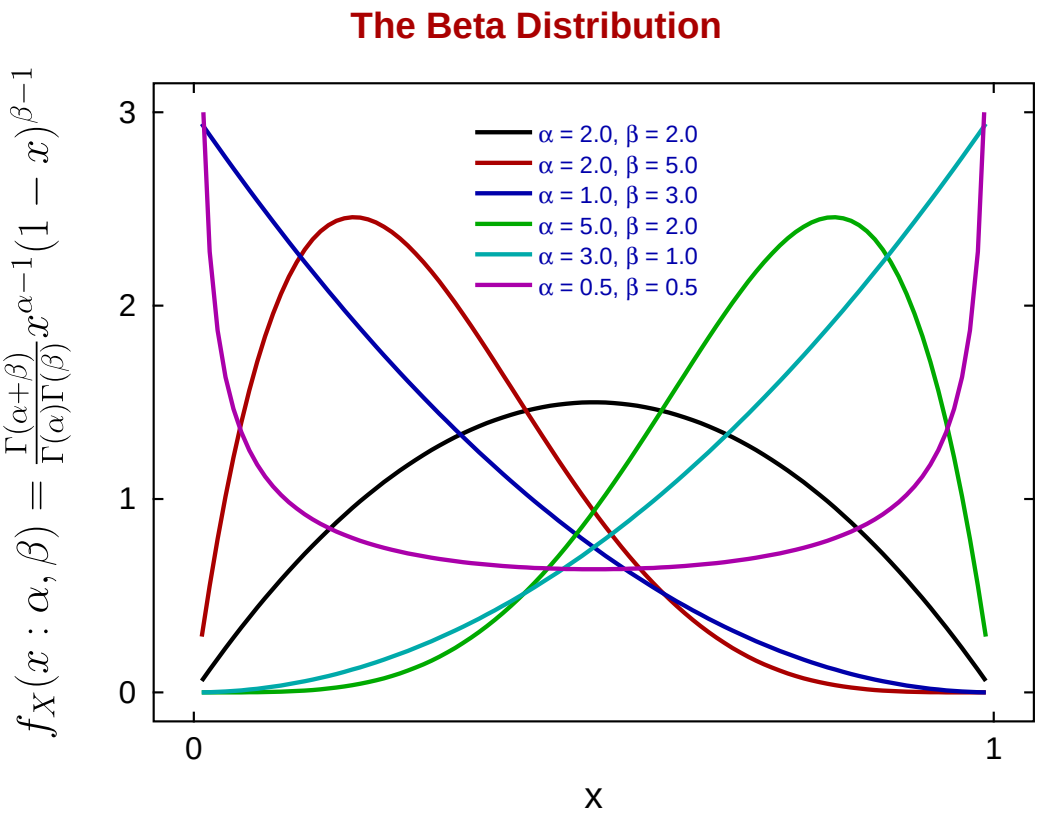
Scalable Vector Graphics, Test Files, and Worked Examples.

w.g.bardsley@gmail.com
https://simfit.org.uk

These collected tutorials illustrate how to use the Silverfrost Software program **EditSVG** written by David Bailey which is bundled with the SIMFIT and SIMDEM DLLs to create fully featured SVG files.

It can also use the **dvips** and **dvisvgm** programs to generate SVG files interactively from LaTeX source, which was inspired by the program **PSfrag** written by David Carlisle at NAG to use LaTeX equations to label EPS plots.



Version 7.5.1

Contents

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

1 SVG: introduction

To appreciate the nature and use of scalable vector graphics files (*.SVG) it is useful to review the document types used to archive graphs and include them into computer generated documents or web pages.

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, and is never device-independent.
- Where antialiasing has been used to smooth out polygons and polylines as in text, curves, or lines, compression can cause fuzzy curves or distorted characters. Traditional computer graphics never looked good on computer screens because hard edges at an angle showed as a series of steps (a distortion known as aliasing). This made 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 when it is displayed at another resolution. SVG sidesteps this problem as the necessary anti-aliasing is performed as an image is displayed (or printed), so that data are displayed correctly. This device–independence is something that sets aside vector graphics from bitmap graphics.

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.

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.

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 cannot be used and cause warning messages. However, in some circumstances a SVG file may use an acceptable subset of SVG facilities. To explore this option you must 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.

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.

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 SimFITand SimDEM, 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 --no-fonts myfile.ps` to create `myfile.svg`.

The argument `-E` indicates that a PostScript file is to be input, while the argument `--no-fonts` directs **dvisvgm** to use Bezier font outlines. 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.

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.

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**.

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.

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp - \left\{ \frac{1}{2} \left(\frac{t-\mu}{\sigma} \right)^2 \right\} dt$$

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.

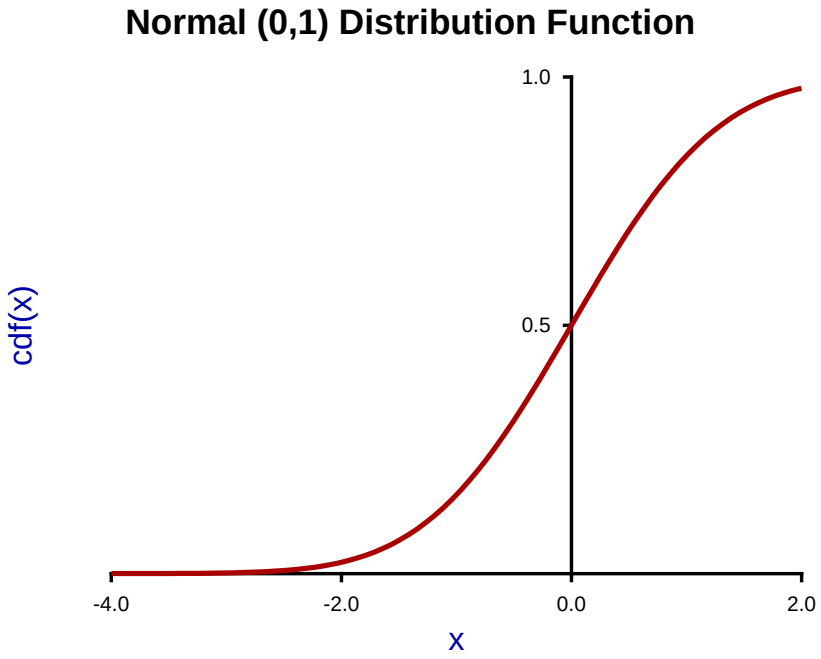
- **latex** `latex_maths_equation.tex`
- **dvips** `latex_maths_equation.dvi`
- **dvisvgm** `-E --no-fonts latex_maths_equation.ps`

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.

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 $-4 \leq x \leq 2$, 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.

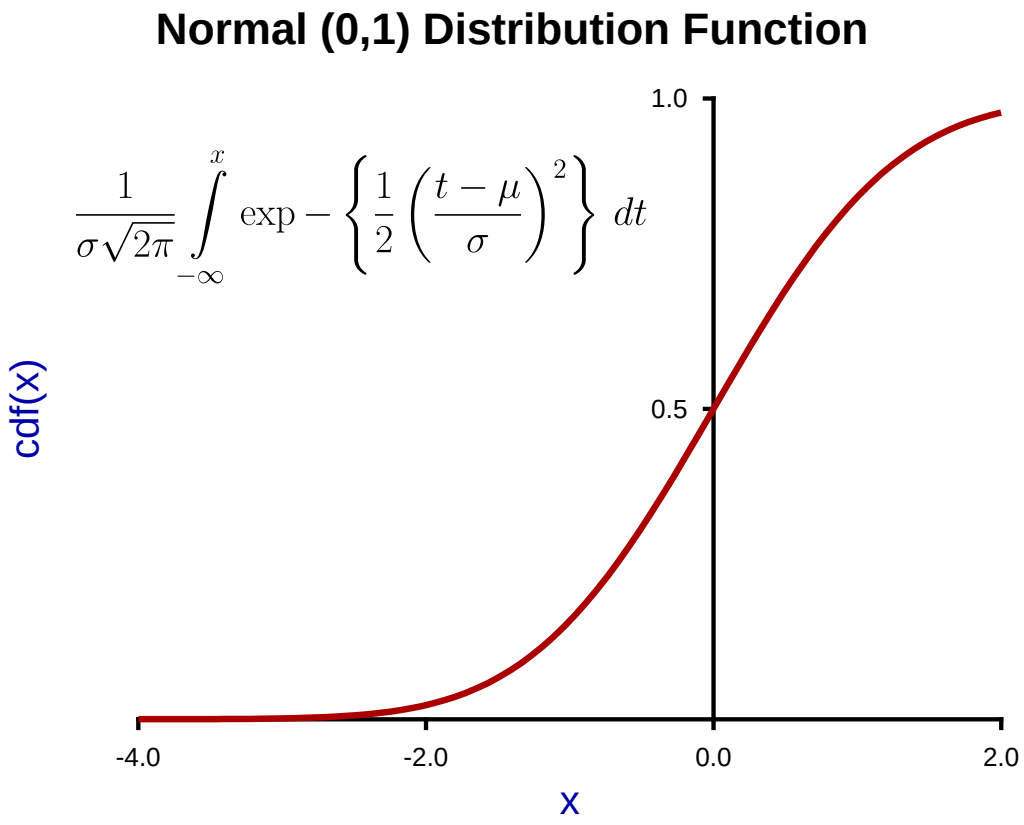
2.3 Joining the SVG files using editSVG

First open program **editSVG** then input the test file `latex_maths_plot.svg` to act as a background, then there are 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.



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

3 SVG: Importing LaTeX chemical formulas

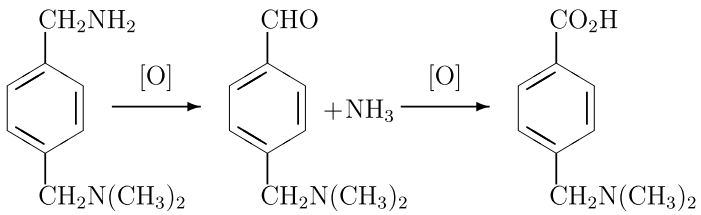
Sometimes it is required use LaTeX to display chemical structures inside a scientific plot, and this document describes how to do this using a condensed scheme for the oxidation of p-dimethylaminomethylbenzylamine. Note that all the files mentioned in this document are distributed as SimFIT test files so that users simply wishing to create the final composed document can proceed directly to the last section describing how to use **EditSVG**

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.



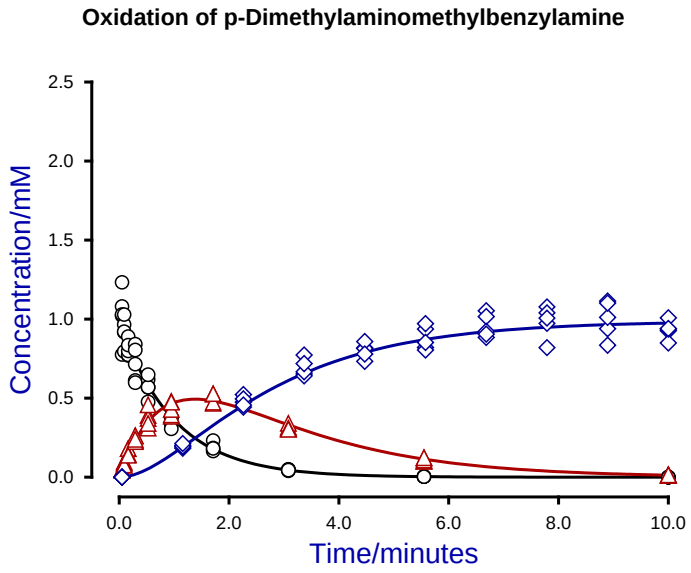
To import this formula into a graph using **EditSVG**, `latex_maths_equation.svg` can be made using the following commands, or `latex_maths_equation.tex` can be input directly into **EditSVG**.

- `latex latex_chemical_formula.tex`
- `dvips latex_chemical_formula.dvi`
- `dvisvgm -E --no-fonts 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.

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`.

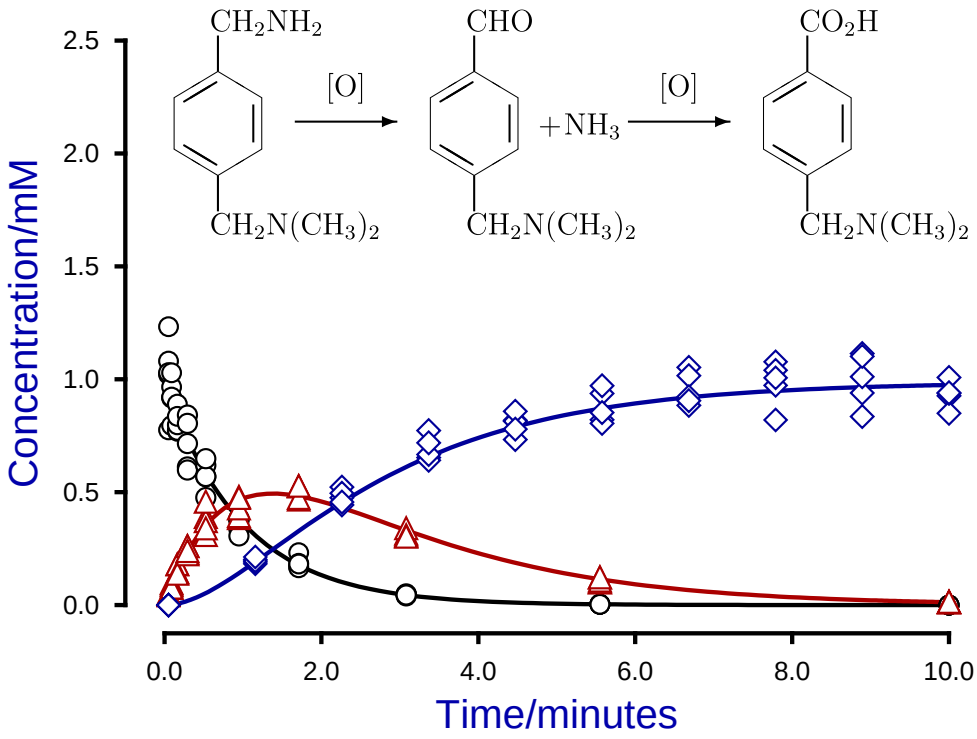
Joining the SVG files using EditSVG

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.



3.3 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

4 SVG: Importing SVG files into SVG files

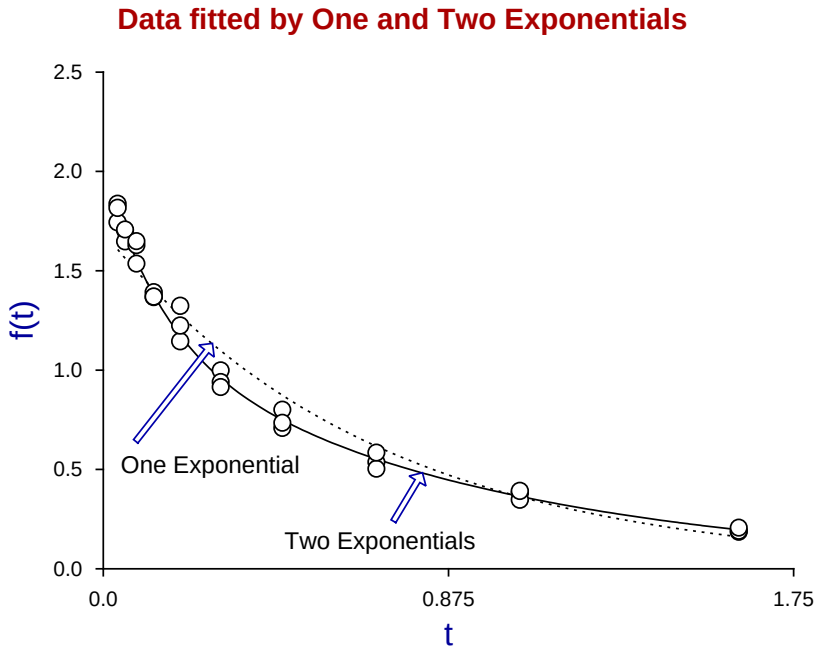
Sometimes it is required to import one SVG file into another SVG file,for example to create overlays, insets, or collages. This document describes how to do this using program **EditSVG** when fitting one then two exponential functions to a data set and plotting the best–fit curves to evaluate the improvement in fit.

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.

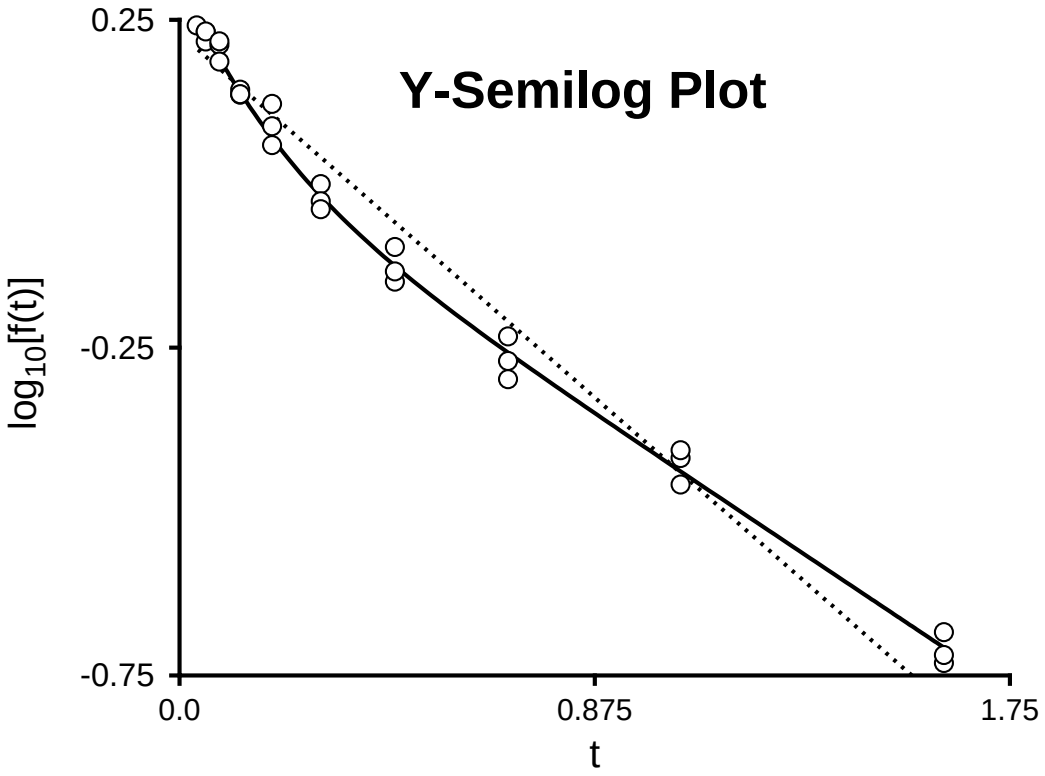
- `exfit.t4f`, the data for input into program **exfit**
- `exfit_normalplot.eps`, the above plot in EPS format for input into program **editps**
- `exfit_normalplot.svg`, the above plot in SVG format for input into program **EditSVG**
- `exfit_normalplot.metafile`, a metafile to create this plot in SimFIT program **simplot** or SimDEM program **simdem70**

The file `exfit_normalplot.svg` was also created at this stage to archive this plot.

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.

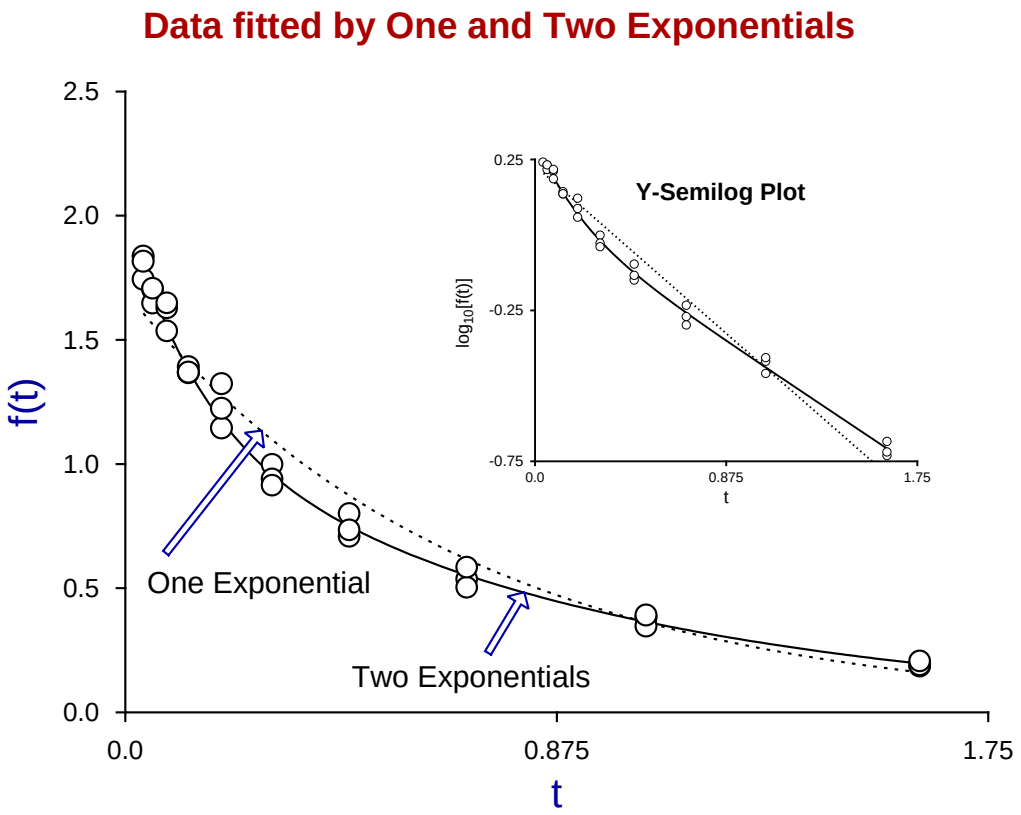
- `exfit_logplot.eps`, the above plot in EPS format for input into program **editps**
- `exfit_logplot.svg`, the above plot in SVG format for input into program **EditSVG**
- `exfit_logplot.metafile`, a metafile to create this plot in SIMFIT program **simplot** or SIMDEM program **simdem70**

The file `exfit.svg` was also created at this stage to archive the compound plot, as described next.

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.



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

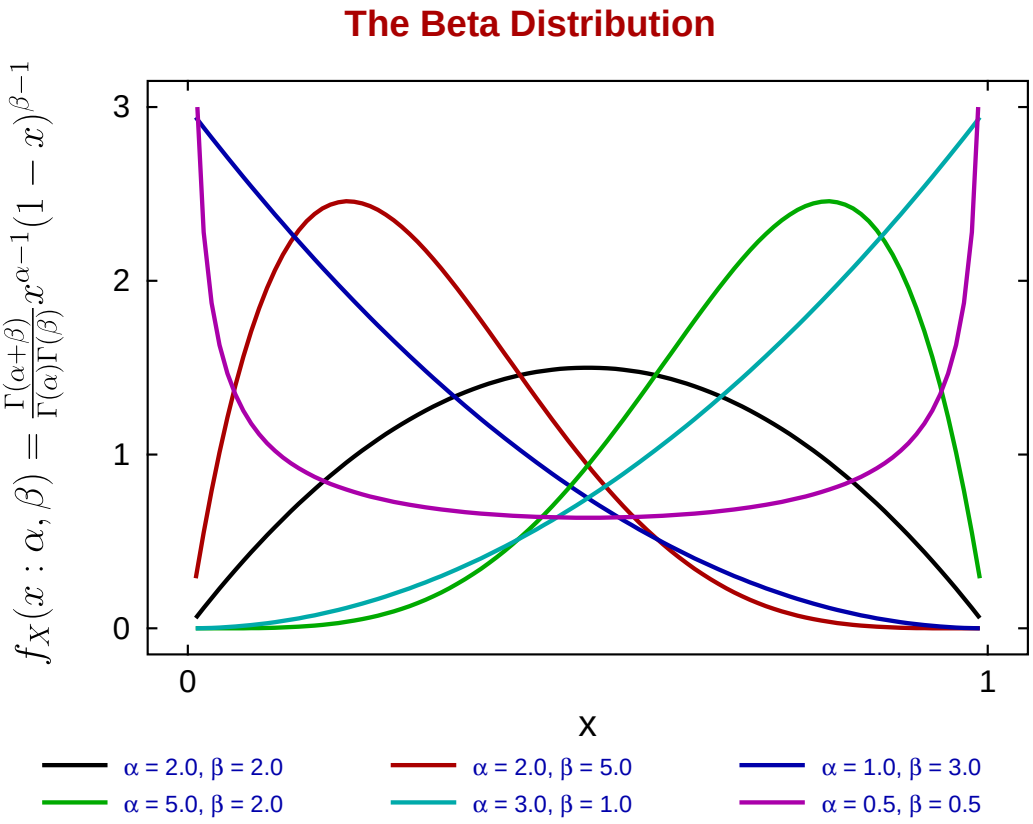
5 SVG: Using LaTeX to label SVG y axes

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**.

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.

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.

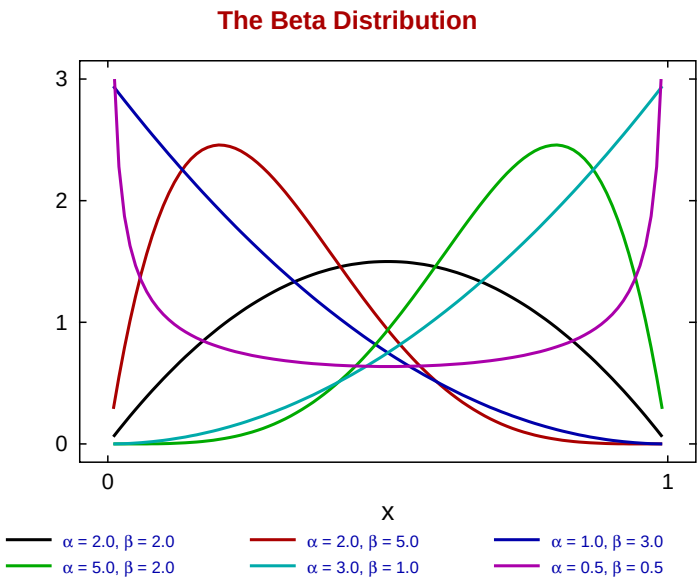
$$f_X(x:\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}$$

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 --no-fonts 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.

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 $0.01 \leq x \leq 0.99$ 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.

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.

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.

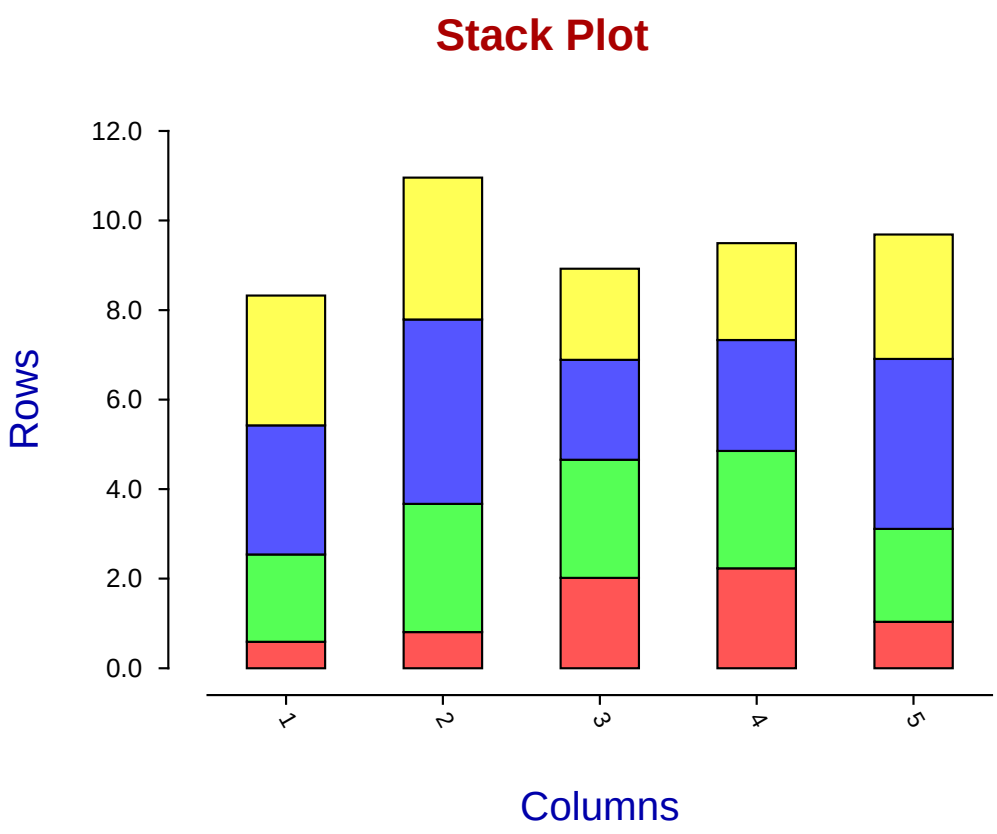
File name	Data included
`beta_pdf_plot.metafile`	SimFIT or SimDEM metafile to create the plot without any equation
`latex_beta_pdf.tex`	LaTeX source file for the beta_pdf equation with no plot
`latex_beta_pdf.svg`	SVG file containing the formula only
`beta_pdf_plot.svg`	SVG file containing the plot only
`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

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.

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**.



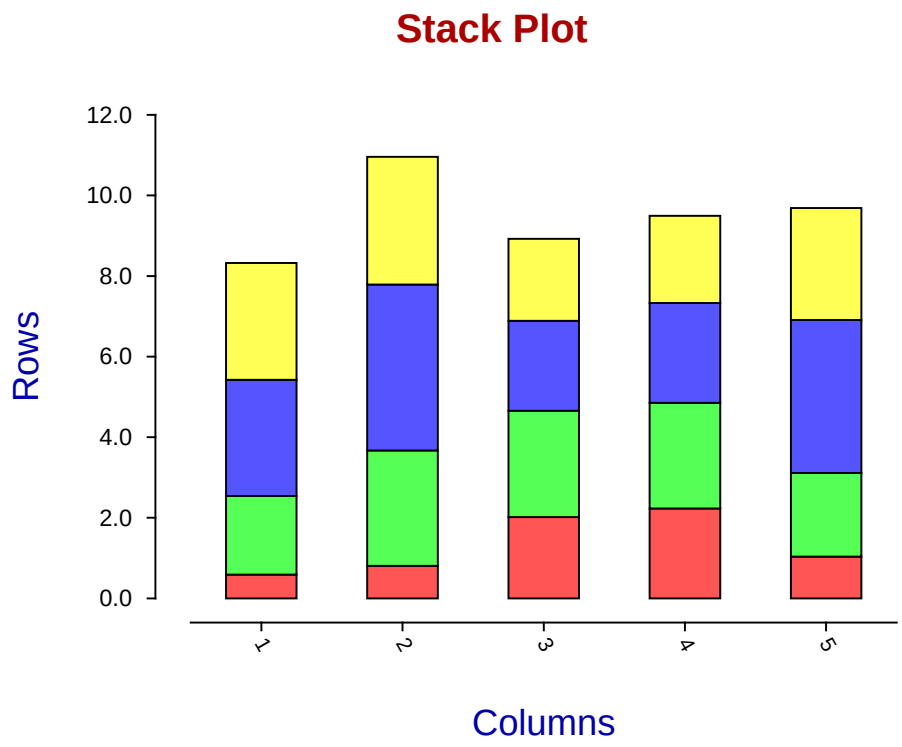
On searching for the string `"Stack Plot"` in the file `stack_plot.svg` using a text editor we find the following markup at line 34, where the tokens have been shown on separate lines for clarity.

```
<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

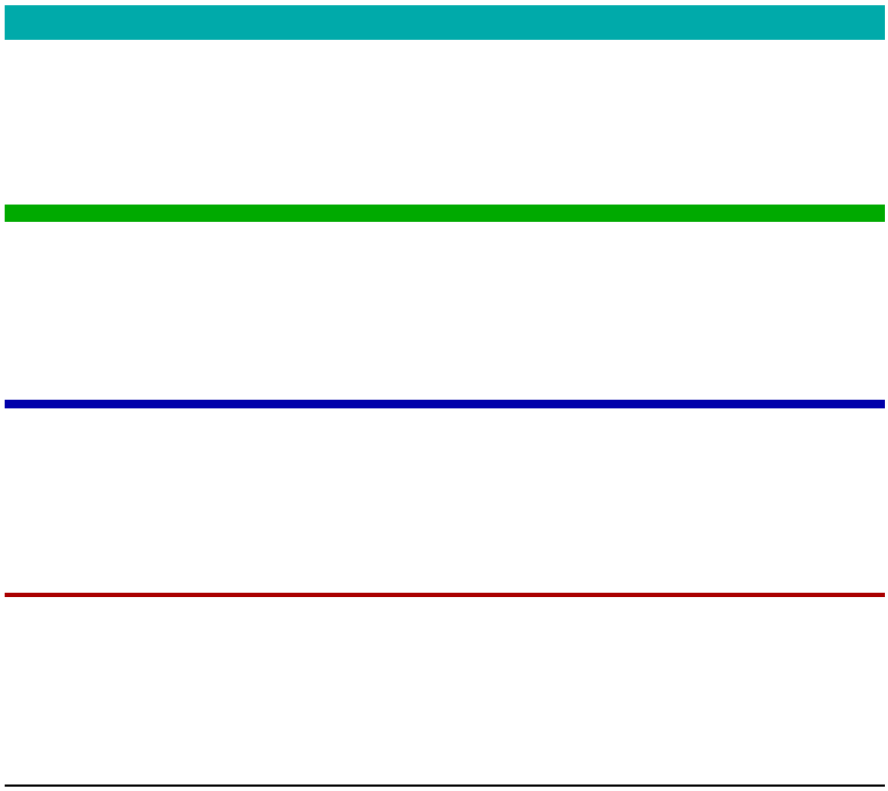
The x-legend color has been changed from blue rgb(0,0,170) to green rgb(0,170,0)

Line 36

The y-legend color has been changed from blue rgb(0,0,170) to green rgb(0,170,0)

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.

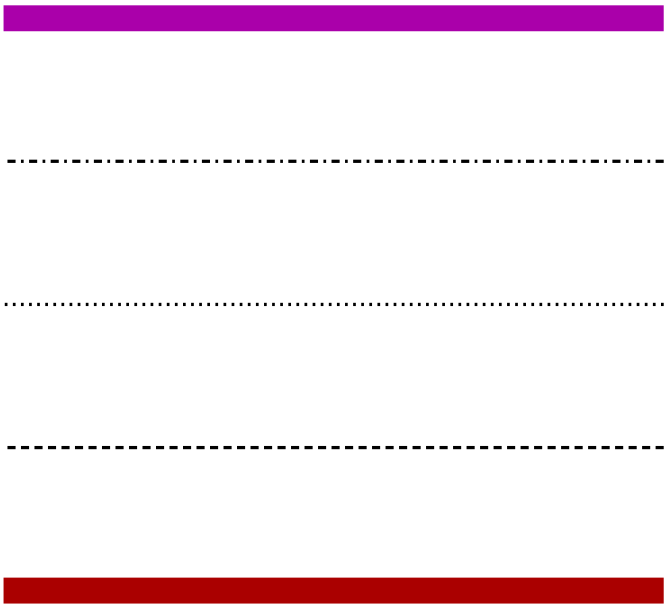
Another change that is often required is to swap between solid lines and dashed, dotted, or dash–dotted lines which requires the definition of a `stroke-dasharray` parameter as follows.

- `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.



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.

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.

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.

7.1 Collage 1: Miscellaneous LaTeX examples

$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \qquad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad \begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix}$$

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \qquad \begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix}$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}}$$

$$\iint\limits_V \mu(v,w)\,du\,dv$$

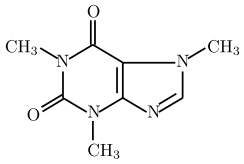
$$\iiint\limits_V \mu(u,v,w)\,du\,dv\,dw$$

$$\int\limits_V \cdots \int \mu(z_1,\ldots,z_k)\,\mathbf{dz}$$

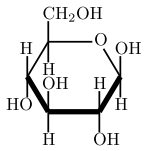
$$\lim_{x\to 0} \frac{\sin^2(x)}{x^2} = 1$$

$$\varliminf_{n\to\infty} |a_{n+1}|/|a_n| = 0$$

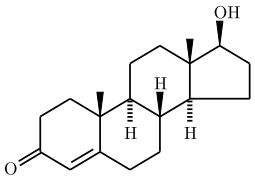
$$\varinjlim (m_i^{\lambda} \cdot M)^* \leq \varprojlim_{A/p\to\lambda(A)} A_p \leq 0$$



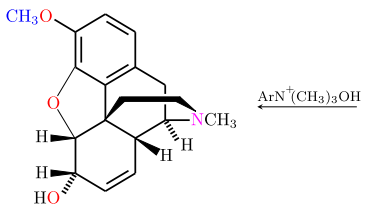
Caffeine



Glucose

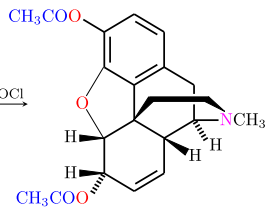


Testosterone



Codeine

Morphine



Heroin

7.2 Collage 2: LaTeX maths

$$S_n = \sum_{k=1}^{n} X_i, \quad k = 0, 1, 2, \ldots, n$$

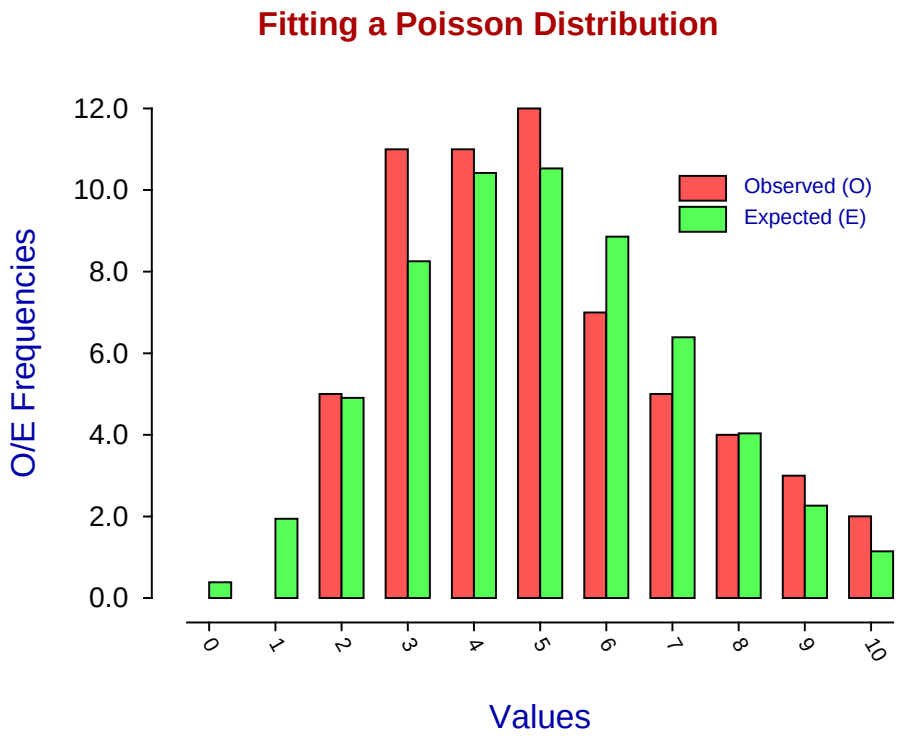
Binomial Distribution: $P(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k}$

$$E(S_n) = np$$
$$V(S_n) = np(1-p)$$

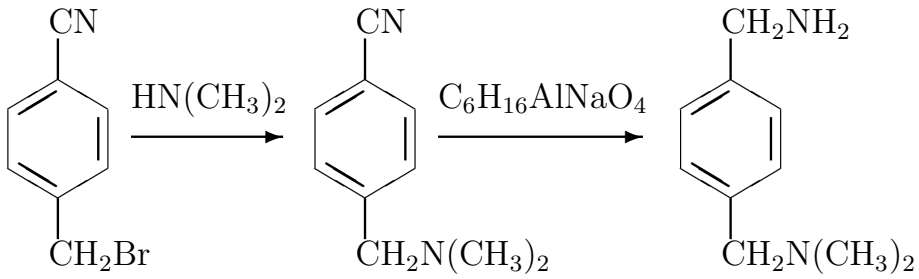
Poisson Distribution: $P(Y = k) = \frac{\lambda^k}{k!} \exp(-\lambda), \quad y = 0, 1, 2, \ldots,$

$$E(Y) = \lambda$$
$$V(Y) = \lambda$$

$$\lim_{n \to \infty, p \to 0} \binom{n}{k} p^k (1-p)^{n-k} = \frac{(np)^k}{k!} \exp(-np), \quad np > 0$$



7.3 Collage 3: LaTeX chemistry

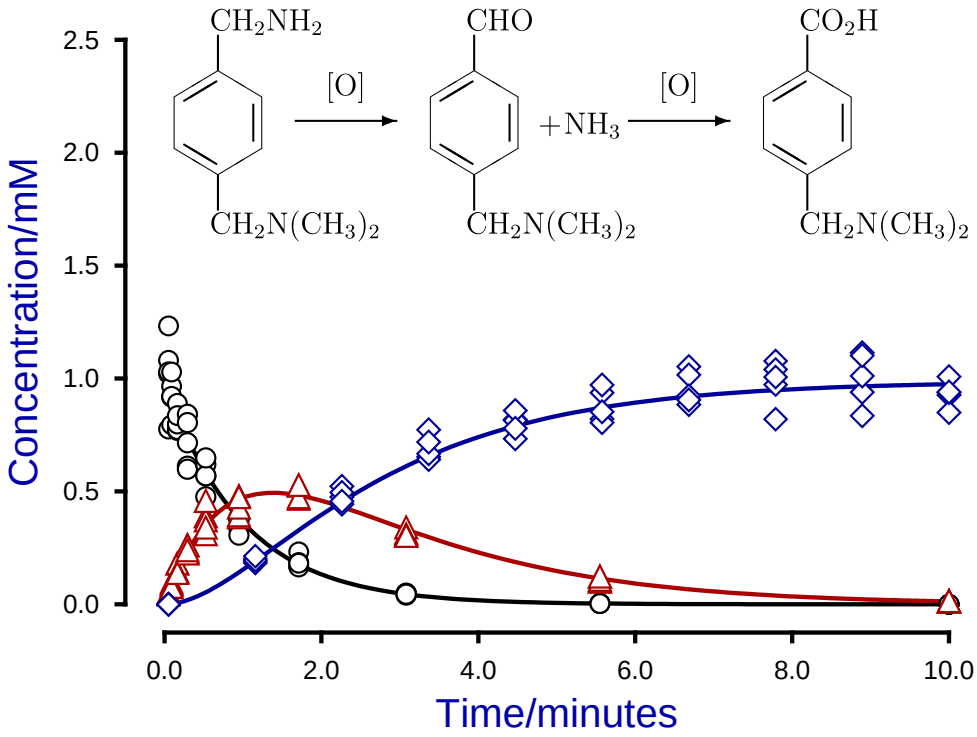


Chemical synthesis of p-dimethylaminomethlbenzylamine

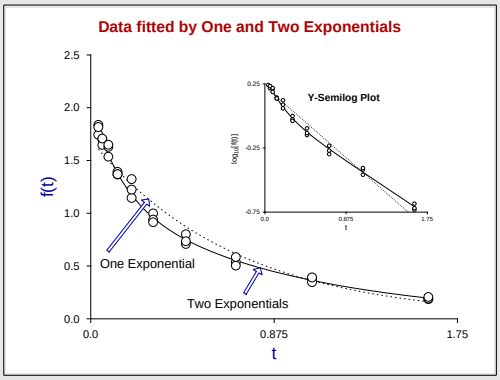
using

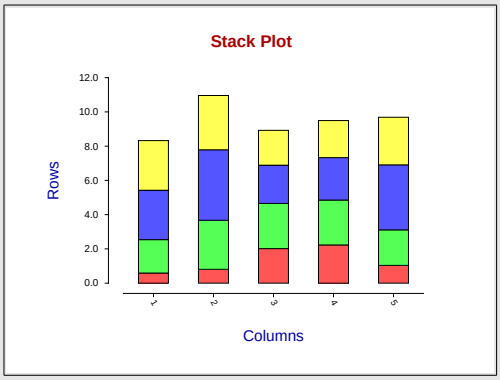
p-cyanobenzyl bromide, dimethylamine and Red-Al

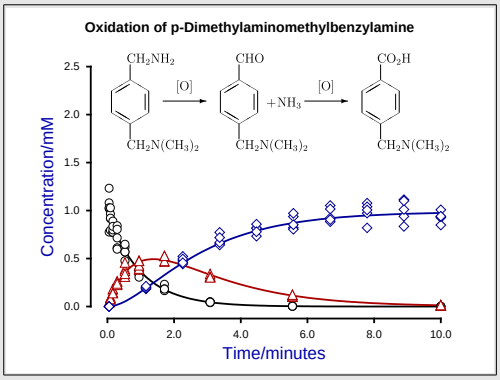
Oxidation of p-Dimethylaminomethylbenzylamine

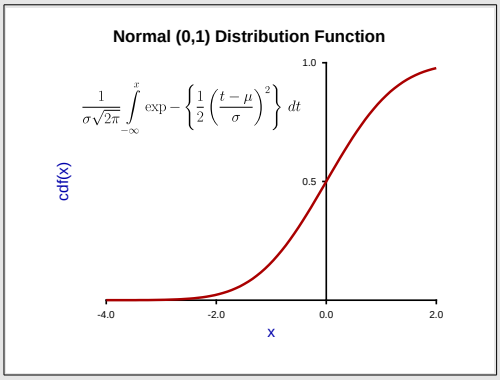


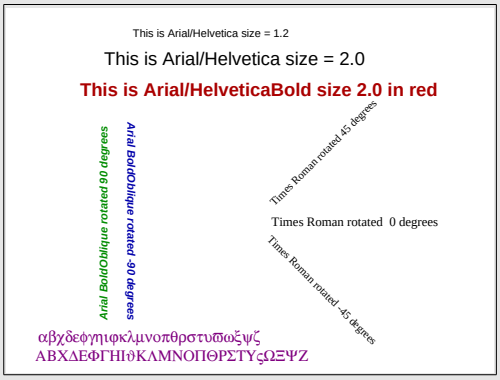
7.4 Collage 4: Tutorial examples

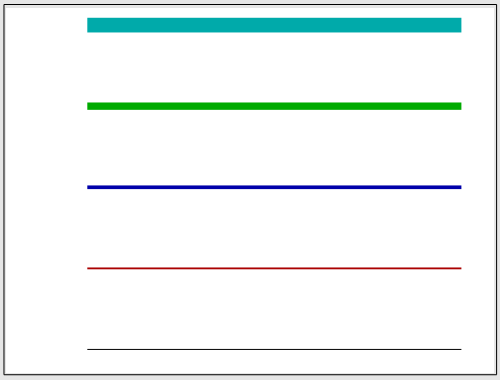












$$S_n = \sum_{k=1}^{n} X_i, \quad k = 0, 1, 2, \ldots, n$$

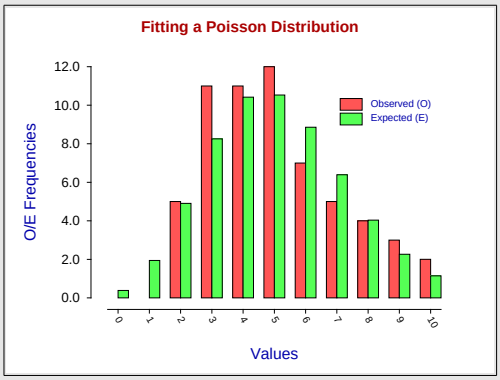
Binomial Distribution: $P(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k}$

$$E(S_n) = np$$
$$V(S_n) = np(1-p)$$

Poisson Distribution: $P(Y = k) = \frac{\lambda^k}{k!} \exp(-\lambda), \quad y = 0, 1, 2, \ldots,$

$$E(Y) = \lambda$$
$$V(Y) = \lambda$$

$$\lim_{n \to \infty, p \to 0} \binom{n}{k} p^k (1-p)^{n-k} = \frac{(np)^k}{k!} \exp(-np), \quad np > 0$$



7.5 Collage 5: Differential scaling to create ribbon graphs