LaTeX Course 2011

Part 5: Graphics

Arho Virkki

VTT Technical Research Centre of Finland

Adding images

Adding images to text became a routine task around the beginning of 1990's.[1]

Because of this, the commands to handle graphics are loaded separately to LaTeX even today:

```
\usepackage{graphicx}
\usepackage{color}
```

or

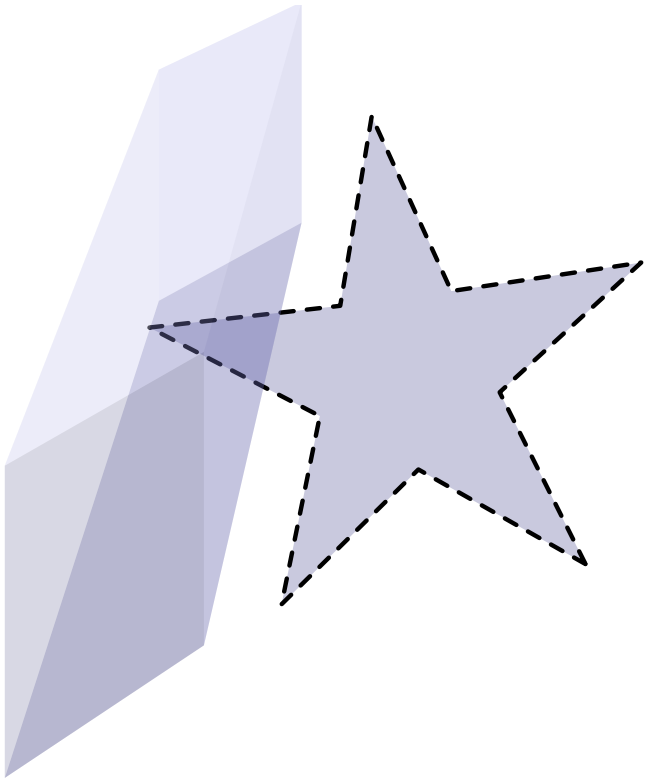
```
\usepackage{graphicx,color}
```

[1] At least to what the author can remember from those early days of computing e.g. without stuff like Internet.

Adding images...

A simple example:

```
\includegraphics{img/foo}
```



Adding images...

- `\includegraphics{file_with_no_extension}` adds an image to the document
- The extension of the file (.eps, .pdf, .png, .jpg) was intentionally left out: The compiler (e.g. `latex` or `pdflatex`) will choose the file with the correct extension on its own.
- Writing the extension explicitly will force the selection (and cause an error on certain cases:
  - The traditional LaTeX understands only the `.eps` files
  - PDFLaTeX can read `.pdf`, `.png` and `.jpg` files, but **not** the `.eps` files $\Rightarrow$ brilliant!...

Adding images...

Changing the format between

`.pdf` ↔ `.eps` ↔ `.png` ↔ `.jpg`

is nevertheless easy.

- In unix, the work can be done with the commands `epstopdf` or `pdftops`[2] or using
- the GIMP or ImageMagic programs (freely available for Windows, Linux and Mac).

[2] The conversion between the 2005 and 2011 LaTeX course graphics files (including several eps illustrations) was done in command line with the single loop: `for f in *eps; do epstopdf $f; done`

Adding images...

By default, the image files are searched from the same folder where the manuscript resides. If there are plenty of images, it might be a good option to store them on a separate sub-folder. In this case, the images must be linked with

- the relative path `\includegraphics{image_path/foo}` or
- using the command `\graphicspath{{}{}...{}}`

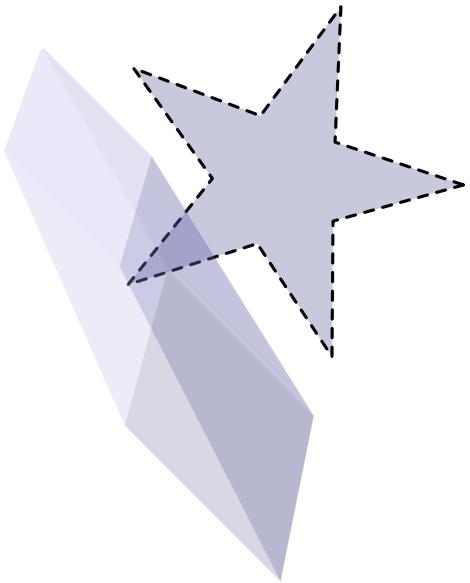
Example:

```
\graphicspath{{/home/arho/images/}
              {images/}}
\includegraphics{images/foo}
```

\includegraphics[]{}

`\includegraphics` accepts quite a number of optional key-value parameters. For example:

```
\includegraphics[width=0.3\textwidth,
                 angle=45]{img/foo}
```



The most common optional parameters include:

- `scale=number` scaling the image as compared to the original
- `width=length` setting the width
- `height=lenght` setting the height
- `angle=degrees` turning the image counterclockwise (=to the positive direction in the mathematical sense)

Note! The lenght must incorporate the unit, e.g. `1.5cm` or `0.2\textwidth` and the scaling factors are with no units.

Figure environments

Sometimes, adding the images to the proper position can be left to the computer, and it is always a good idea to let the machine to keep record of the actual image number. Example:

```
\begin{figure}[!htbp]
\begin{center}
\includegraphics[width=0.4\textwidth]{img/foo}
\caption{A schematic illustration of the Gadget}
\label{fg:foo}
\end{center}
\end{figure}
```

Figure environments...

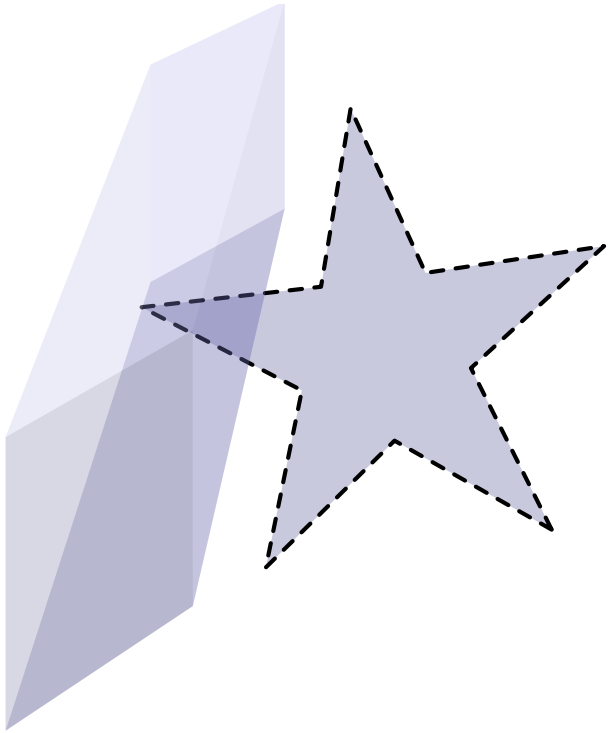


Figure: A schematic illustration of the Gadget

The image can be cited later with `\ref{fg:foo}` (which will be replaced with the actual image number in the compiled document).

Figure environments...

The optional parameter `[!htbp]` means, that

- I really would like(**!**) to insert
- the picture right **h**ere or, if not,
- to the **t**op or
- to the **b**ottom of this page and,
- if this also fails, to a separate **p**icture page.

Figure environments...

There exists an additional package `floatflt` for making the text to surround the images[3]

```
\begin{floatingfigure}{4cm}
\includegraphics[width=3cm]{img/foo}
\caption{StarBox3D}
\label{fg:fooii}
\end{floatingfigure}
By inspecting the figure, we see that the optimal
decision threshold for separating the control and
positive classes is obviously at $b$, where also
the conditional probability $p(P|b)=0.5$. However,...
```

[3] Apparenty there are some issues with the freedom of the license of this package, but it can be installed by hand

http://blogs.fau.de/johanneshabich/2010/05/20/latex-floatflt-sty-missing-on-ubuntu-lucid-10-04/

Figure environments...

By inspecting the figure,
we see that the optimal decision threshold for separating the control and positive classes is obviously at b, where also the conditional probability $p(P|b) = 0.5$. However,...

Figure: StarBox3D

Colors

Colors are easiest to add with the package

```
\usepackage{color}
```

but the number of colors is rather limited — but should at least include white, black, red, green, blue, cyan, magenta and yellow[4]. To define a custom color, we need the additional command

```
\definecolor{col_name}{col_system}{col_def}
```

For example

```
\definecolor{mygreen}{rgb}{0.61,0.78,0.05}
\colorbox{vihrea}{Nice Green Backround!}
```

Nice Green Backround!

[4] http://en.wikibooks.org/wiki/LaTeX/Colors

Colors...

Using the pre-defined colors is easy

```
\colorbox{orange}{Orange}
\fcolorbox{red}{green}{Pearl}
\textcolor{blue}{
\[ \sum_{k=1}^\infty a_k \]}
```

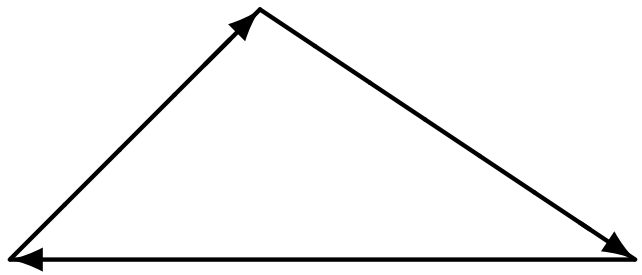
Orange Pearl

$$\sum_{k=1}^{\infty} a_k$$

Using LaTeX to draw images

LaTeX has a built-in system to draw simple images. The following example is from the book of Kopka and Daly.

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(5,2)\thicklines
  \put(5,0){\vector(-1,0){5}}
  \put(0,0){\vector(1,1){2}}
  \put(2,2){\vector(3,-2){3}}
\end{picture}
```



Drawing images...

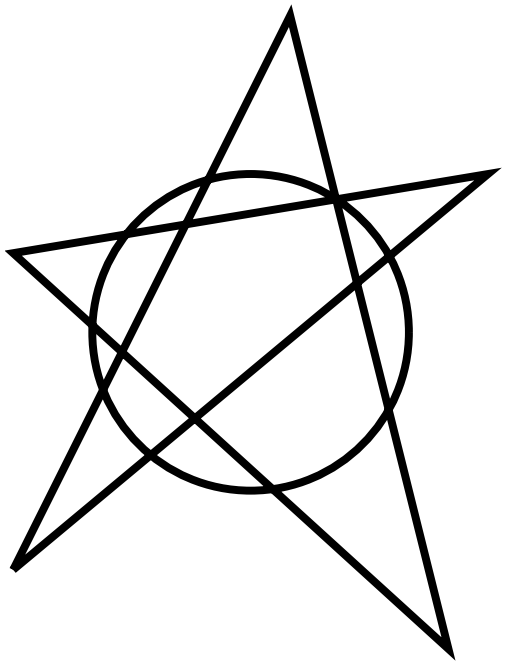
For a comparison, some PostScript code[5] (again from K&D):

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 169 158 233 242
220 200 moveto
200 200 20 0 360 arc
170 170 moveto
230 220 lineto
170 210 lineto
225 160 lineto
205 240 lineto
170 170 lineto
stroke
showpage
```

[5] PostScript is nowadays superseded by the pdf format

Drawing images...

This code produces the image

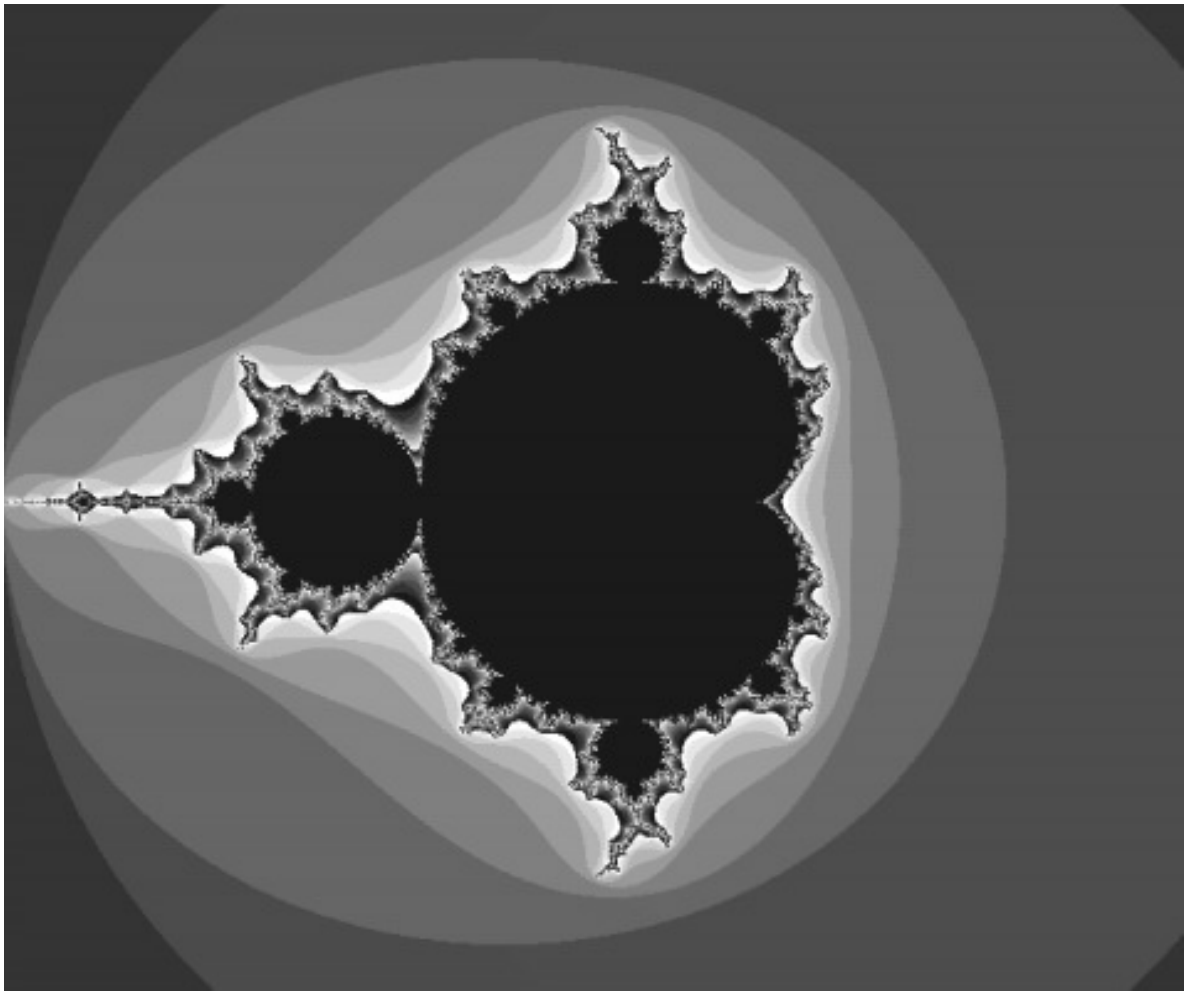


The following example is a bit more complicated. . .

Drawing images...

```
%!ps
/iter 60 def /reso .005 def /sq { dup mul } def
/mod { 2 copy div floor mul sub } def /plot {
newpath moveto 1 0 rlineto stroke } def gsave
280 420 translate 260 2 div dup scale 2 260 div
setlinewidth -2 reso 2 { /x exch def -2 reso 2 {
/y exch def /r 0 def /i 0 def /n 0 def iter { r
sq i sq add 4 gt { exit } if /rr r sq i sq sub x
add def /i 2 r mul i mul y add def /r rr def /n
n 1 add def } repeat n 10 mod .1 mul .1 add
setgray x y plot } for } for grestore showpage
```

Drawing images...



Drawing images...

- The command languages are precise, because the images are not stored as pixels, but as lines, circles, Bézier curves,. . . , and text (with the corresponding size and font).
- For some purposes, using svg or similar language might be beneficial.
- The LaTeX default language is rather poor, as e.g. the angles of the line can only have some fixed values
- A better investment for a casual used is to learn a drawing tool (e.g. Inkscape[6] for vector graphics or Gimp[7] for digital photographs)

[6] http://inkscape.org/
[7] http://www.gimp.org

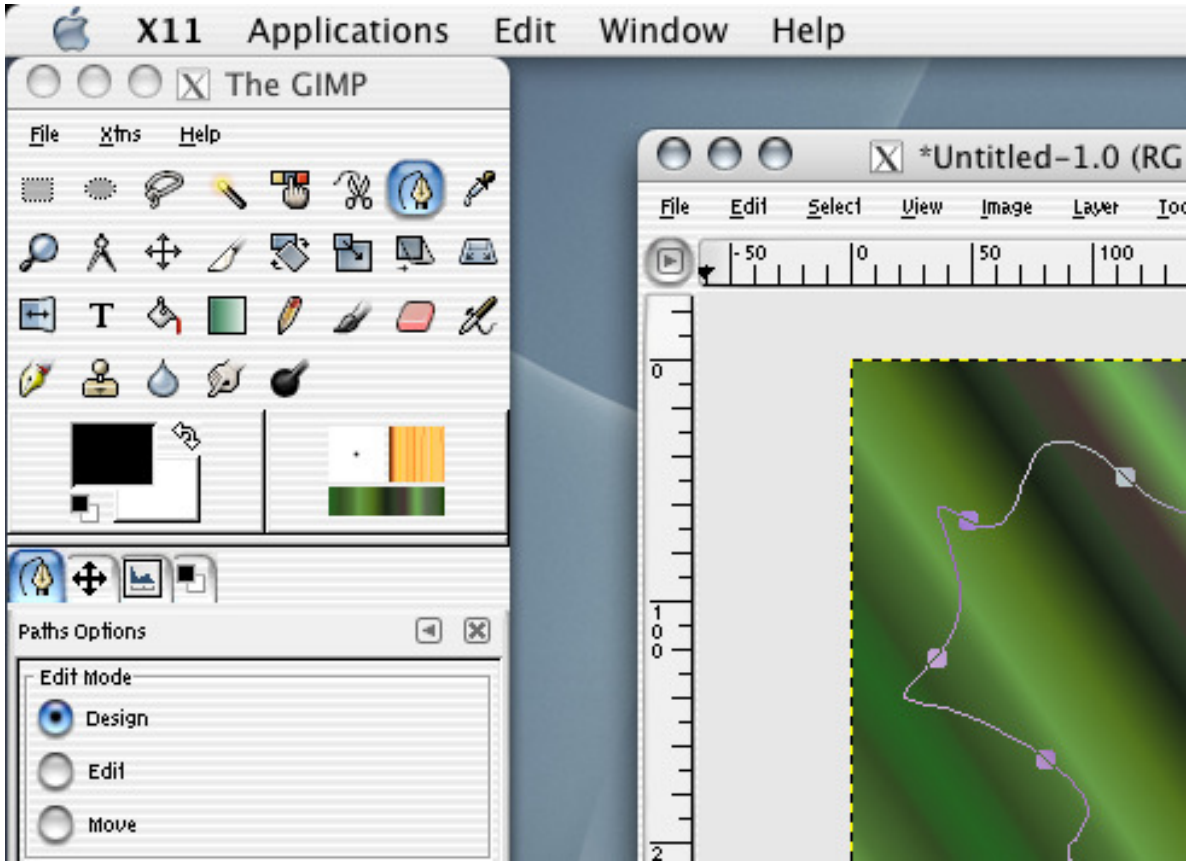
Drawing images...

Which program to use? The answer depend on the personal taste (as well as the willingness to pay for commercial software). Nevertheless, if we restrict ourselves only to the free options,

- GIMP is the choice for painting (pixel map images like digital photos)
- Inkscape is the choice for vector graphics (where e.g. the circles are truly Platonian, and the device (= printer or screen) only does its best to approximate the idea)
- Programming environments like R, Matlab and gnuplot all produce high-quality plots in pdf format (which can be further polished with Inkscape, if desired).

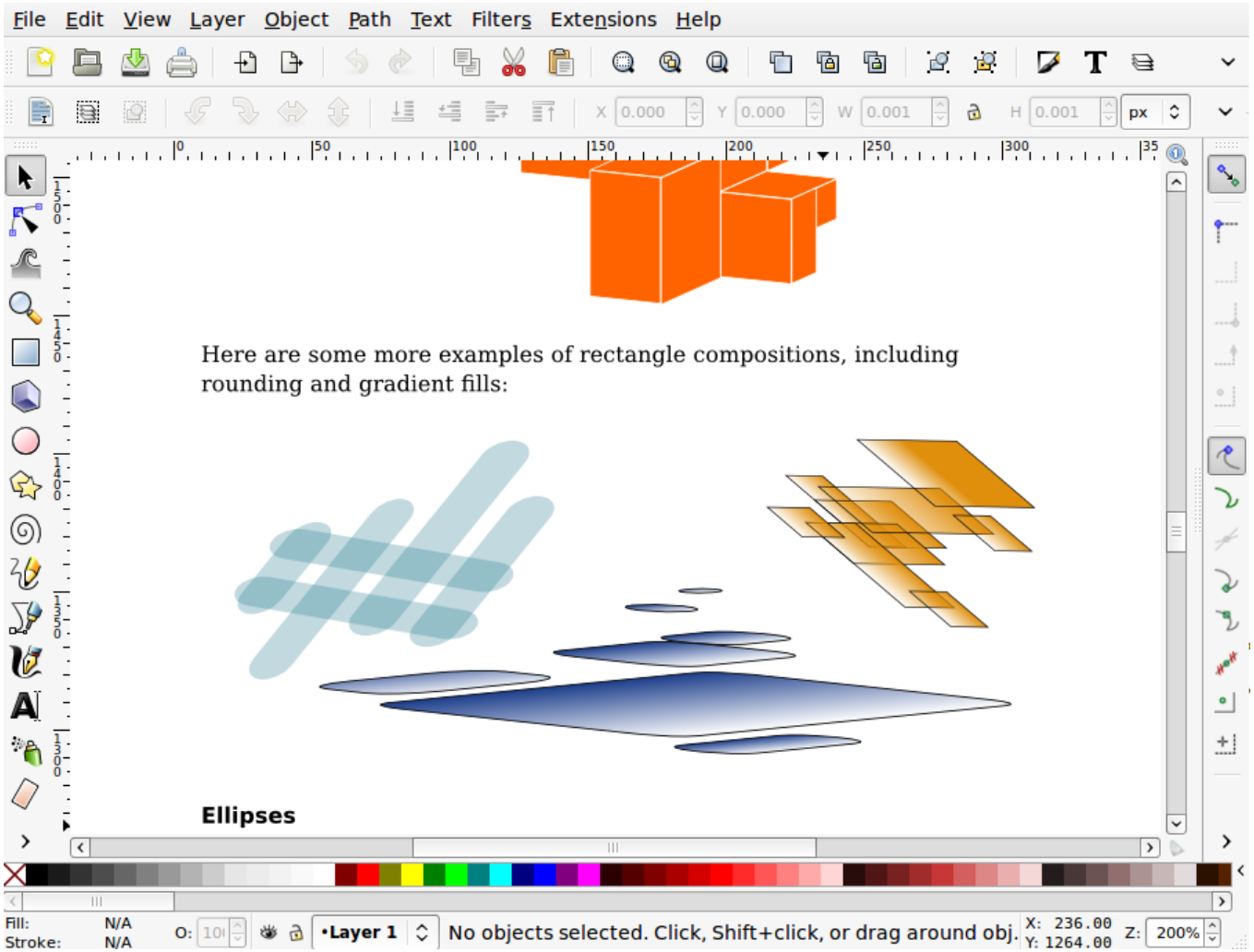
Drawing images...

Gimp is purely a program for painting (bitmap images, including digital photographs and scanned text pages)



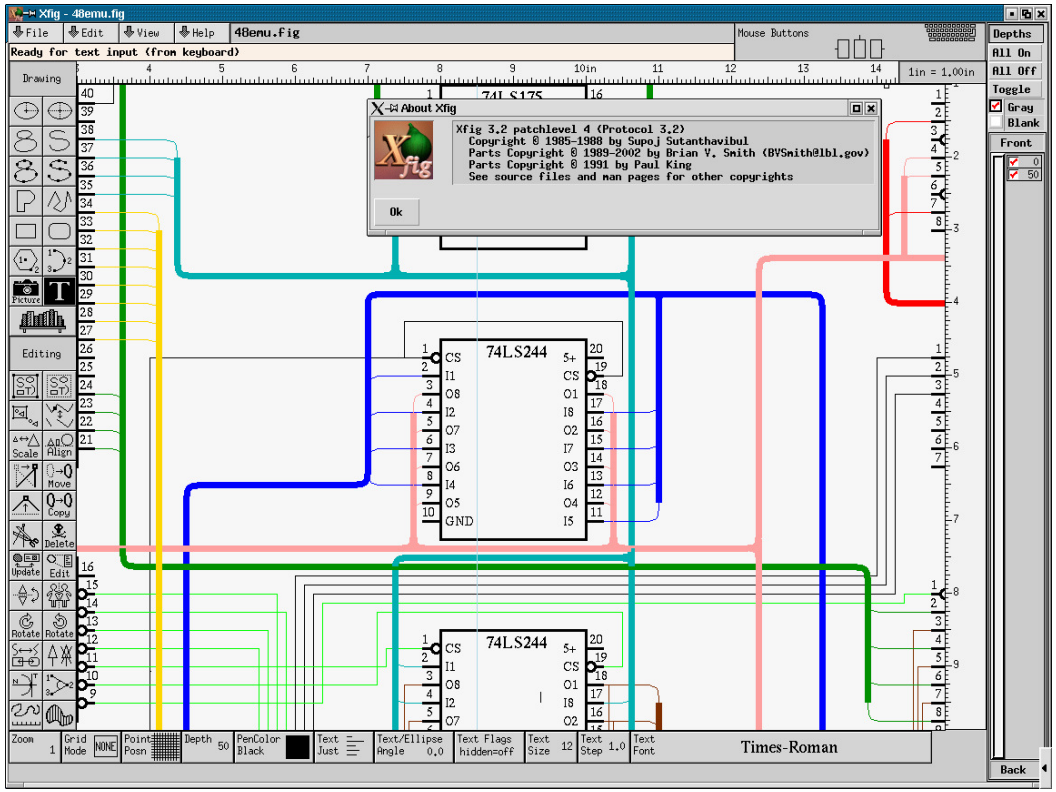
Drawing images...

Inkscape is for vector graphics and resembles the CorelDraw! software (originated at the early 90's)



Drawing images...

Xfig has not been much updated since 2002 (and the latest change is from 2007) but some people are so used to it, that the program can still be found e.g. from the default Debian repositories (and has also been ported to Mac OS X).



Drawing images...

`gnuplot` is yet another “traditional tool”[8] which uses its own command language[9]

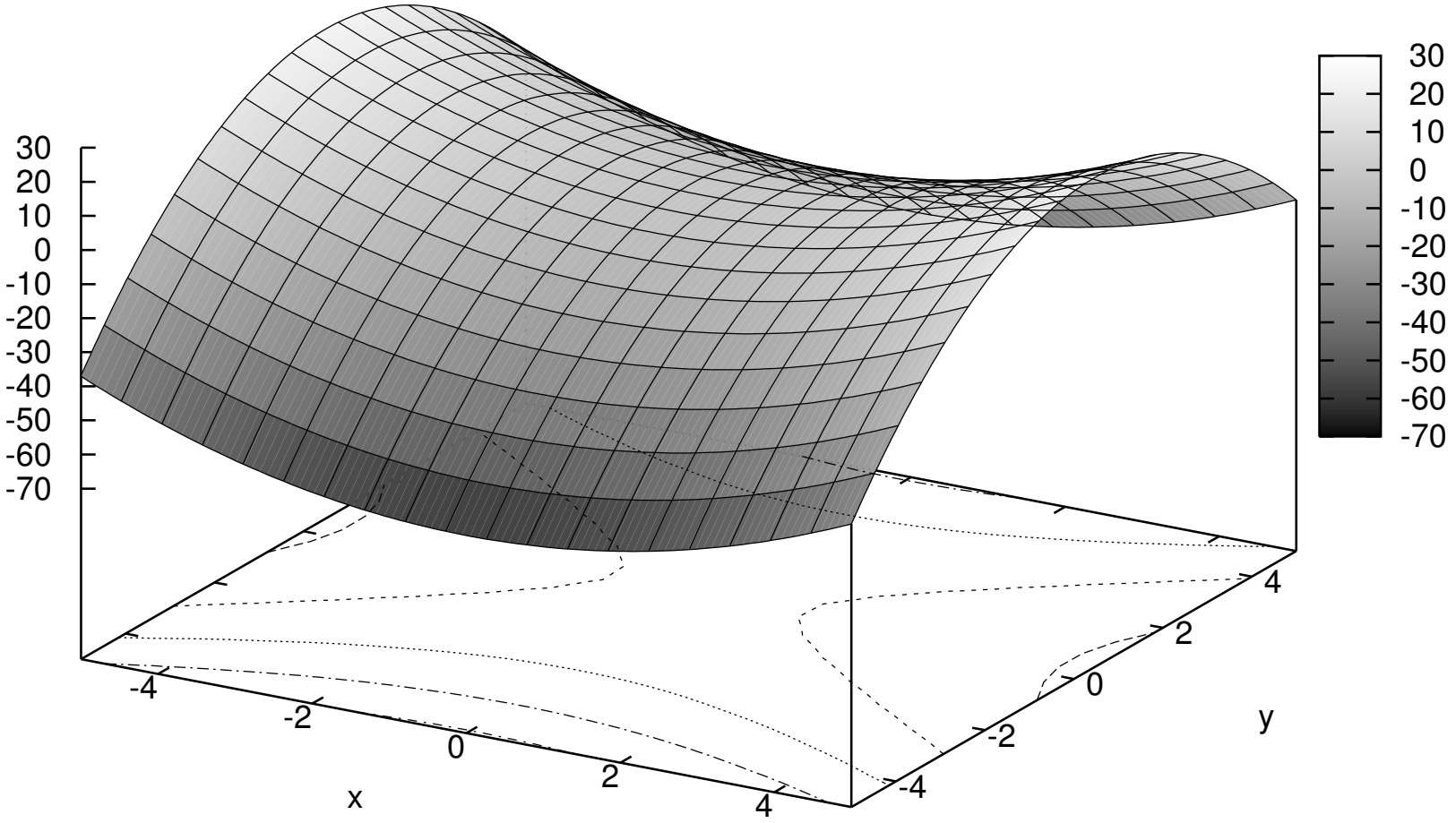
```
set pm3d
set contour base
set xrange [-5:5]
set yrange [-5:5]
set isosamples 20,20
set xlabel "x"
set ylabel "y"
unset key
set term post eps enhanced
set output "gnuplotex.eps"
splot x**2-2*y**2 + 2*y -2
```

[8] Observe that all these “obsolete” tools are younger than LaTeX...

[9] `set pm3d` refers to the OS/2 presentation manager.

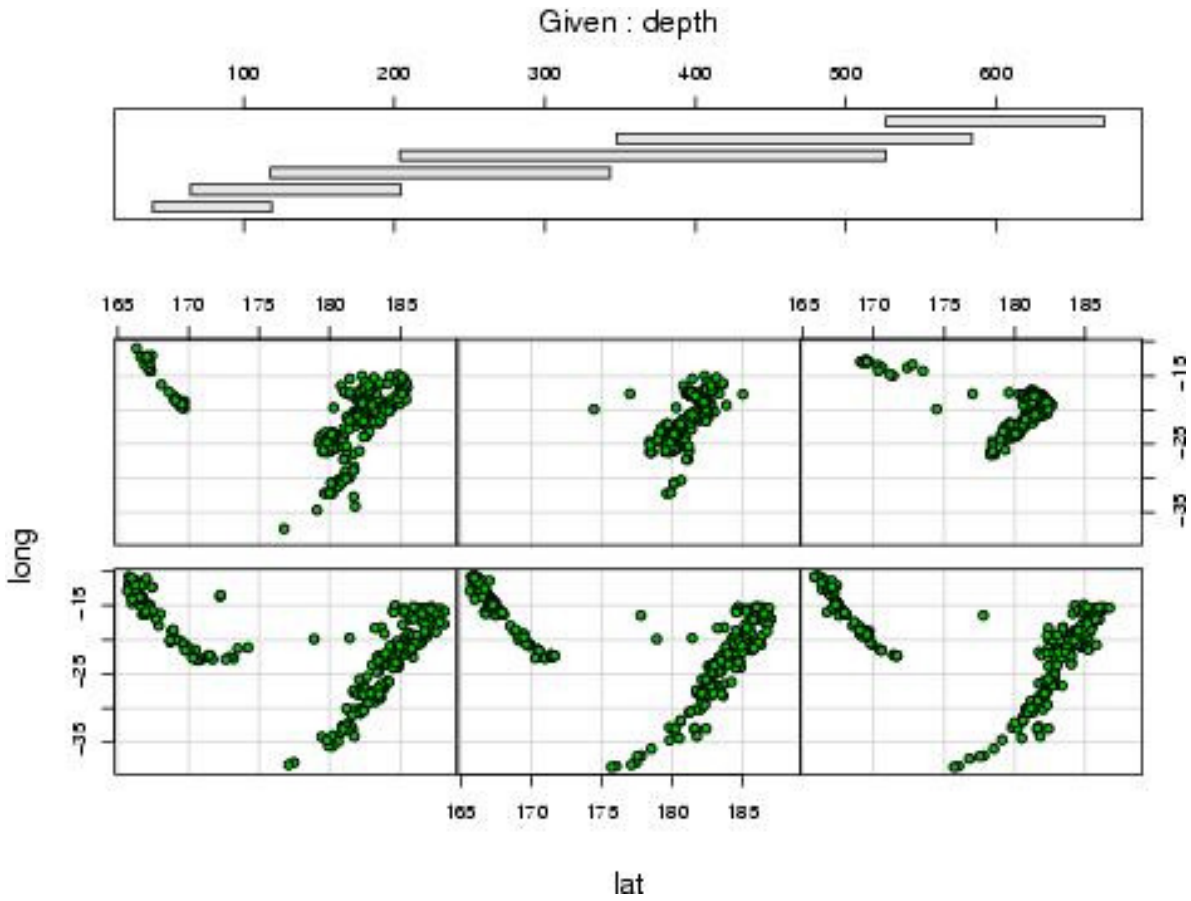
Drawing images...

The previous code produces the image



Drawing images...

The R language and environment for statistical computing[10] is an excellent choice for drawing statistical images



[10] http://www.r-project.org/

Drawing images...

Of course, there are commercial options for vector graphics:

- CorelDraw!,
- Adobe Photoshop,
- Adobe Illustrator,...

and software for mathematical illustrations:

- Maple,
- Mathematica,
- Matlab,...

Inkacape use case: Images with equations

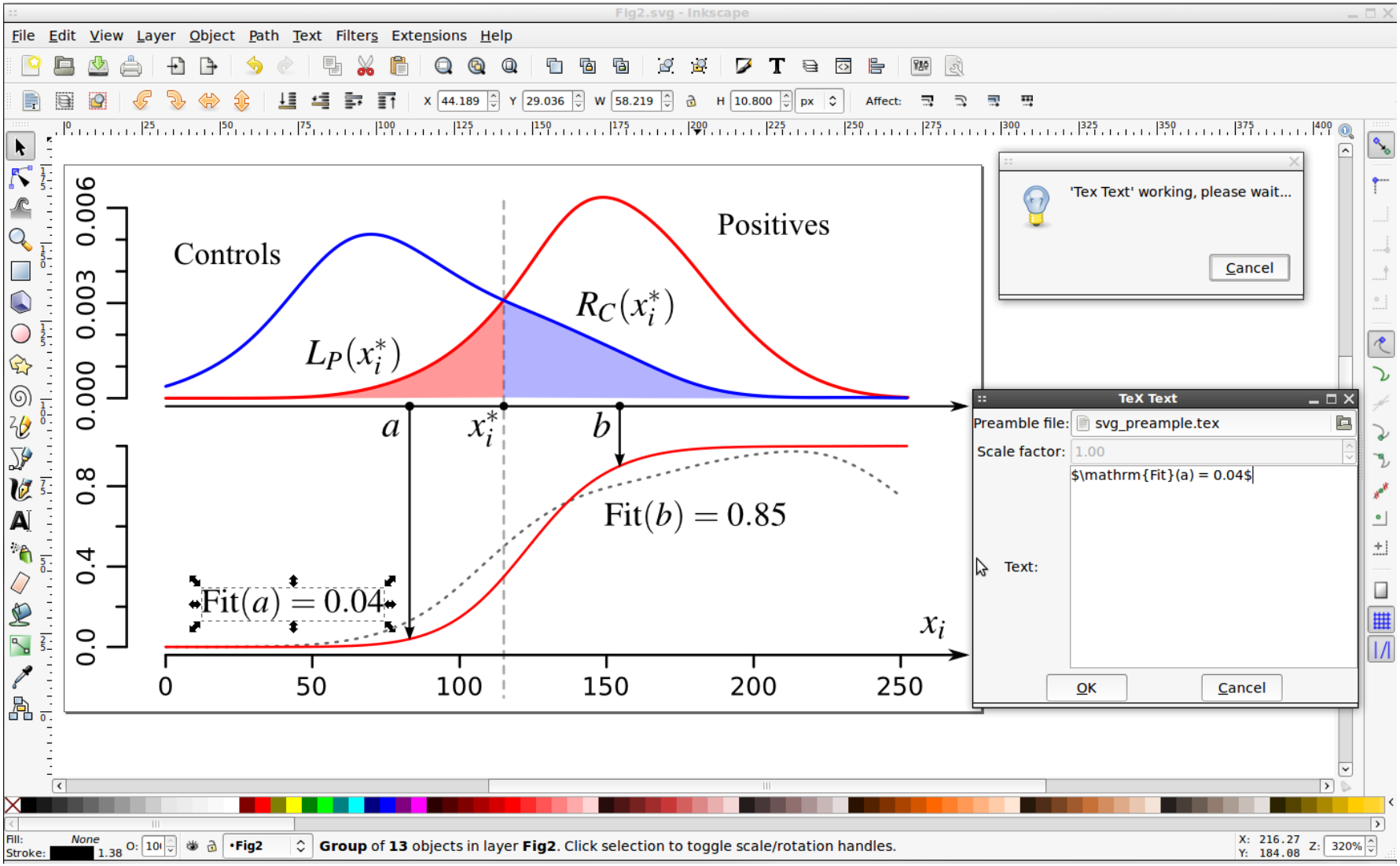
Suppose that you are producing an A0-sized poster presentation for a conference and would e like to include text and images with equations embedded. The two feasible options are:

- Use Inkscape to produce the whole thing or
- Use whichever software you like for the presentation and the images, but polish the separate figures using Inkscape

with Pauli Virtanen's `textext` plug-in installed from http://pav.iki.fi/software/textext/[11]

[11] The installation instructions are non-trivial and different for each platform, but the for superior quality, the work is worth of it. . .

Inkscape use case: Images with equations...



Inkacape use case: Images with equations. . .

The final result in: "J Mattila, J Koikkalainen, A Virkki, M van Gils, and J Lotjonen. Design and application of a generic clinical decision support system for multi-scale data. Transactions on Biomedical Engineering, (in press), 2011"[12]

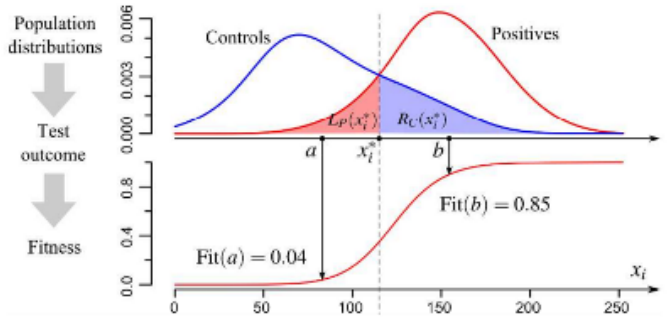


Figure 1. Probability density functions of C_i and P_i, the resulting *fitness* (with examples at test outcome values a and b), and the optimal classification threshold x_i*.

The software library uses a statistical approach to analyze multi-scale data and combine them into an aggregate representation interpretable by a clinician. It supports heterogeneous patient data of virtually any type and scale and allows clinicians to study the system simultaneously as a collection of components and as a whole. The library has been designed to easily support several diseases, requiring minimal amount of configuration. The first application prototype developed using the proposed decision support library is a CDSS tool for early diagnosis of AD. The statistical methods are validated using data from several medical datasets and the clinical applicability of our proposed system is demonstrated by evaluating the implementation of the CDSS tool.

The main contributions of this work are the description of

$$DSI(x_1, x_2, \cdots, x_n) := \frac{\sum_{i=1}^{n} Rel(i) Fit(x_i)}{\sum_{i=1}^{n} Rel(i)}, \quad (1)$$

where $Rel(i)$ is a *relevance* function providing the weighting between [0,1] for variable i and $Fit(x_i)$ is a *fitness* function providing a non-linear transformation of value x_i into *fitness* space [0,1].

A *fitness* function computes the location, i.e. rank, of an individual variable x_i relative to values of the same variable in two different populations, denoted as controls C_i and positives P_i. Our system currently supports scalar, ordinal, and categorical (including boolean) variables, but could be extended to support others, such as value lists and complex values, by deriving appropriate *fitness* functions. Let us consider a scalar variable where the progression of a disease tends to increase its value (see Figure 1). For these, *fitness* is defined as a monotonically increasing function

$$Fit(x_i) := \frac{L_P(x_i)}{L_P(x_i) + R_C(x_i)}, \quad (2)$$

where $L_P(x_i)$ is the left integral of probability density function (PDF) for positive class values P_i and $R_C(x_i)$ is the right integral of PDF for control class values C_i. Derivation of the *fitness* function can be conducted in an analogous manner for ordinal variables. For a categorical variable $x_i \in \{\Omega_1, \cdots, \Omega_n\}$, we use as *fitness* the conditional probability of the subject belonging to the positive population in the case of observing $\Omega = x_i$.

[12] http://www.vtt.fi/aivotutkimus