Graphics with LaTeX and R

Latex Smart
6356 Agricultural Road
University of British Columbia
Vancouver BC
V6T 1Z2

March 13, 2003

Outline

- How to import graphics into LaTeX?
  - tips on resizing and rotating figures without distortion in LaTeX
  - creating side-by-side and stacked graphics in LaTeX
  - using the “minipage” environment to fit graphics and text within a single page more effectively
  - customizing captions
  - landscape figures
- How to use R to generate nice graphics?
  - controlling graphics parameters
  - adding legends and text to plots
  - using “plotmath” to do LaTeX-like typesetting in R
  - using the “lattice” package for trellis graphics
  - setting “layout”
  - exporting graphics from R

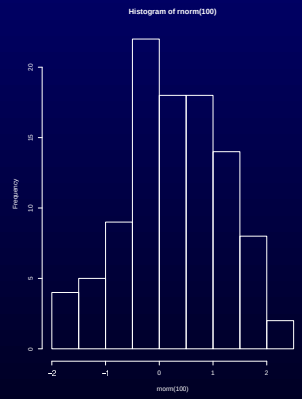
How to import graphics into LaTeX?

package: `\usepackage{graphicx}`

command: `\includegraphics[options]{filename}`

example:

```
\begin{center}
 \includegraphics[height=0.3\textheight]{hist}
\end{center}
```



Options of `includegraphics`

When using `graphicx` package, we can use the following options:

width specifies the width to which the figure should be scaled to; if the option `height` not given, LaTeXwill automatically scale the height of the figure so that the height and width have the same ratio as that in the original figure;

height specifies the height to which the figure should be scaled to; if the option `width` not given, LaTeXwill automatically scale the width of the figure so that the width and height have the same ratio as that in the original figure;

angle specifies the angle of rotation, in degrees, with a counter-clockwise (anti-clockwise) rotation being positive.

Page Layout of LaTeX

```
\includegraphics[height=0.3\textheight]{hist}
```

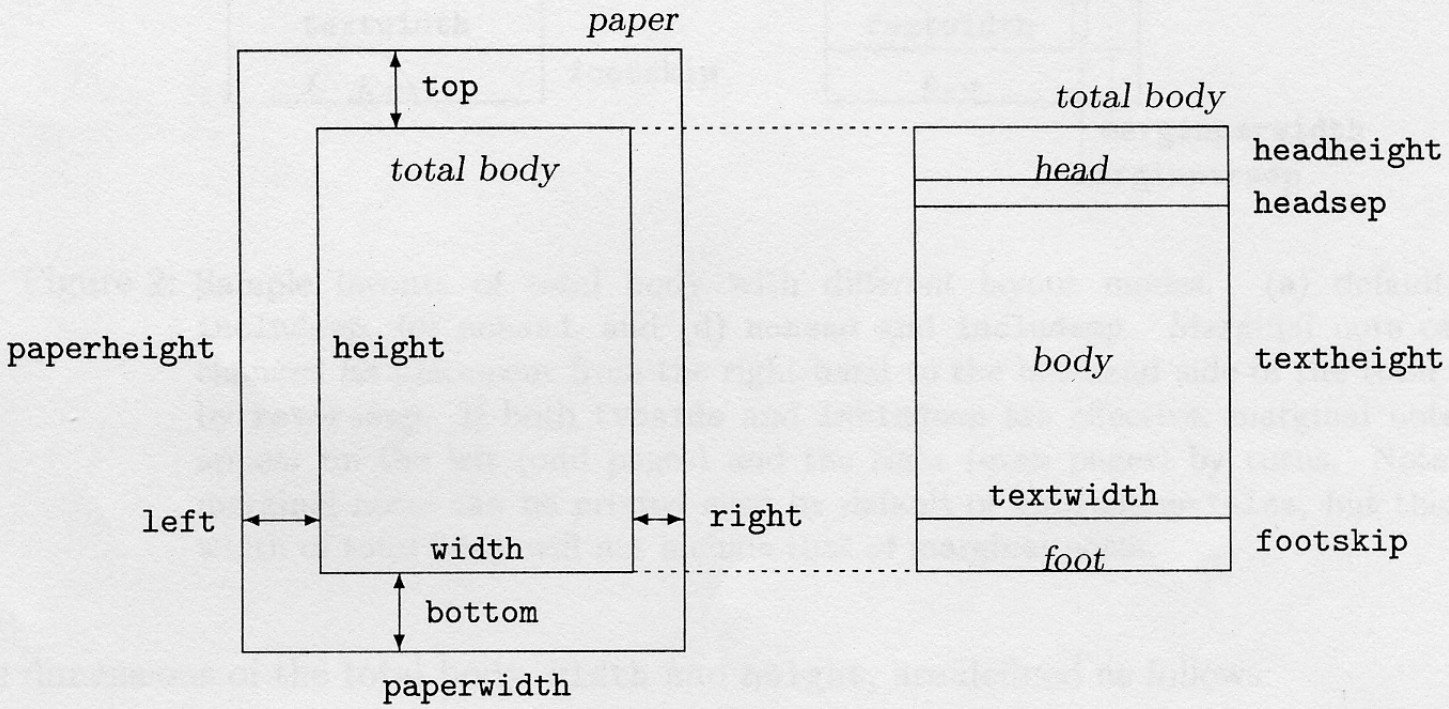


Figure 1: Dimension names for page geometry. If `includemp` is `false` (default), `width=textwidth`.

Sample LaTeX box

A box is any LaTeX object (characters, graphics, etc.) that is treated as a unit.

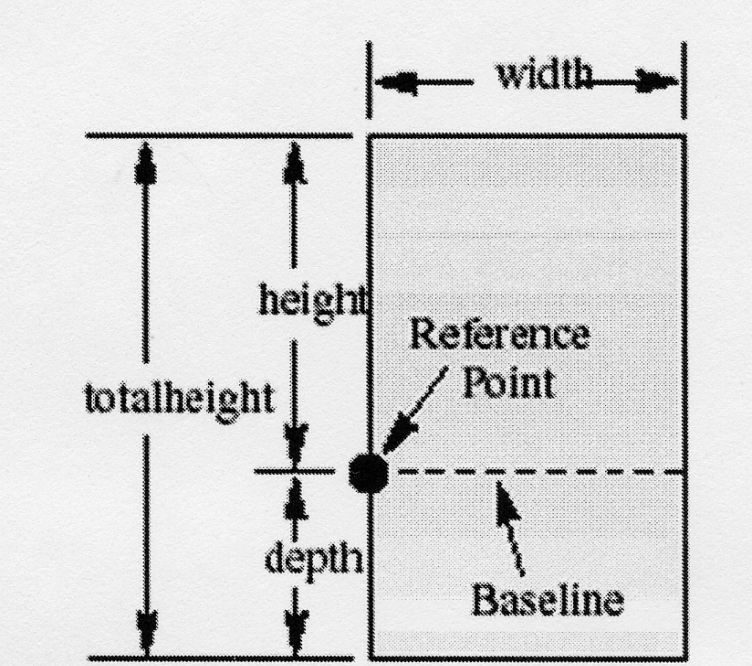


Figure 1: Sample LaTeX Box

Rotated LaTeX Boxes

The reference point of a non-rotated EPS graphic is its lower-left corner.

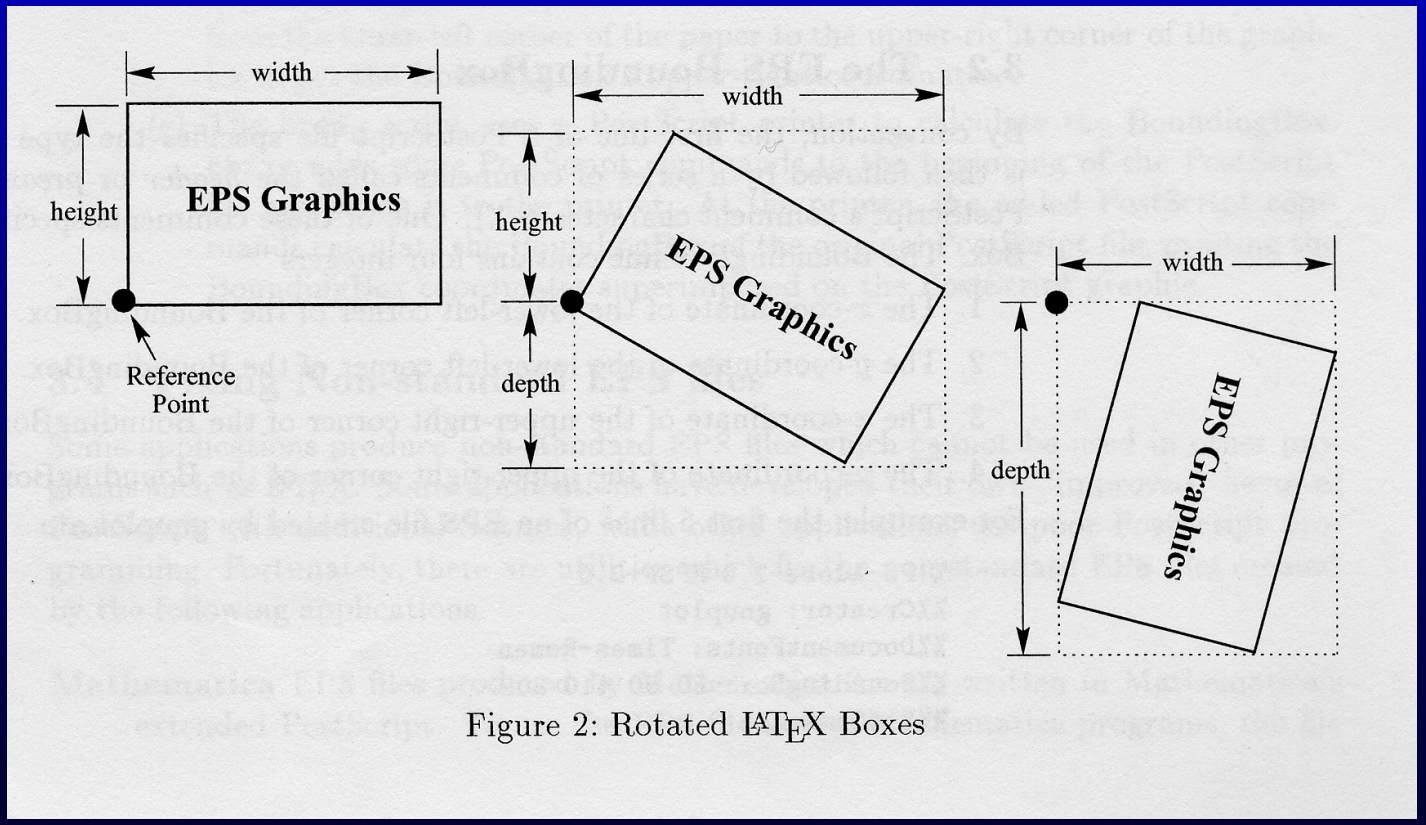
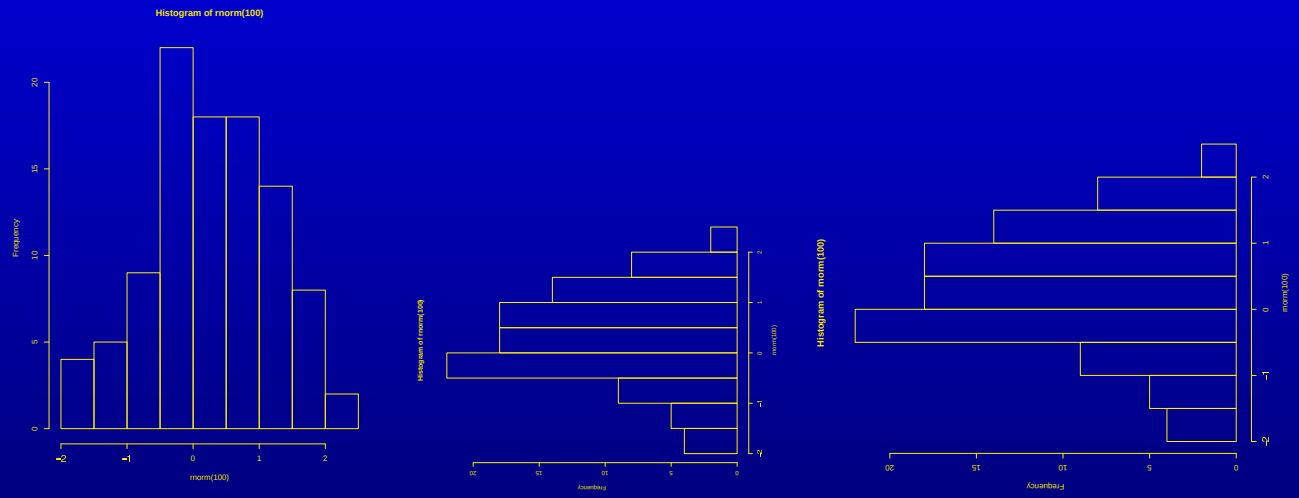


Figure 2: Rotated LaTeX Boxes

The order of `angle` and `width` or `height` matters.




```
\begin{figure}[h]
\centering
\includegraphics[angle=0,width=0.2\textwidth]{hist}
\includegraphics[angle=90,width=0.2\textwidth]{hist}
\includegraphics[width=0.2\textwidth,angle=90]{hist}
\end{figure}
```

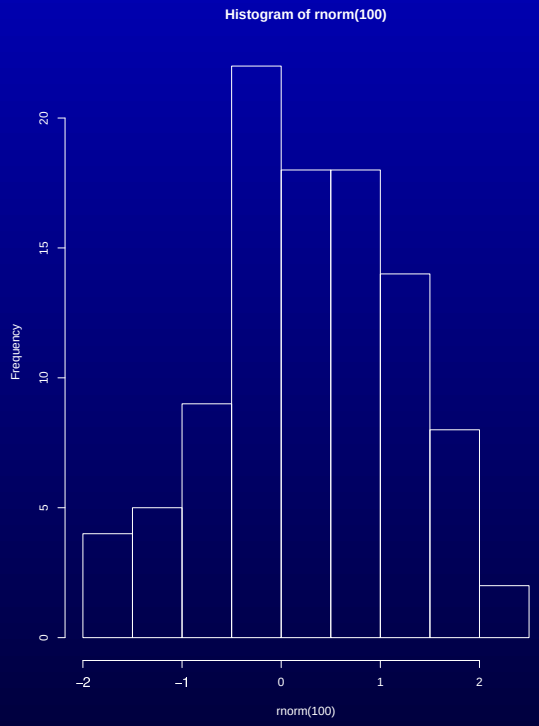
The second box is first rotated 90 degrees and then scaled such that its width is 0.2 times text width. The third box is first scaled such that its width is 0.2 times text width and then it is rotated 90 degrees.

What graphic format latex can import?

- When TeX was written, POSTSCRIPT/EPS, JPEG, GIF, and other graphic formats did not exist.
- DVI files are often converted to POSTSCRIPT.
- The best-supported imported-graphic format is Encapsulated PostScript (EPS).
- To insert PDF graphics, use `pdflatex` command to compile the LaTeX documents.
- Other non-EPS graphics (e.g. JPEG, GIF) can be directly imported into LaTeX when using `pdflatex` command. In case of creating DVI files, such graphics must be converted to EPS for insertion into LaTeX documents.

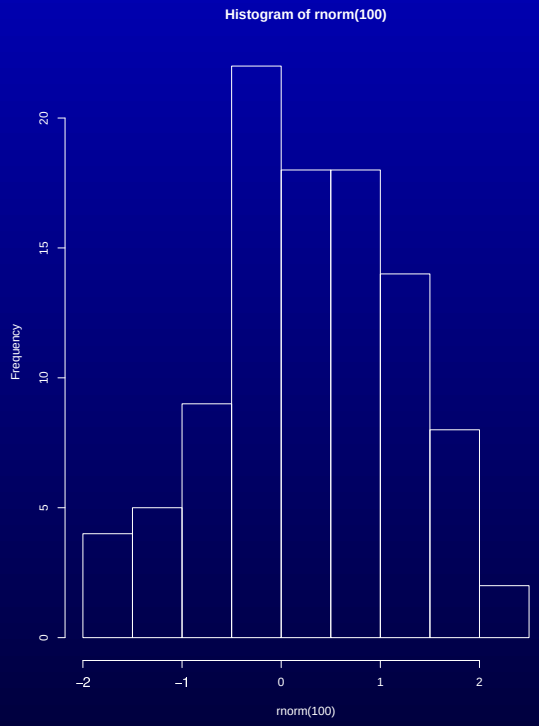
Side-by-Side Graphics

```
\includegraphics[width=0.3\textwidth]{hist}%
\includegraphics[width=0.3\textwidth]{lake1}%
\includegraphics[width=0.3\textwidth]{whistler}
```



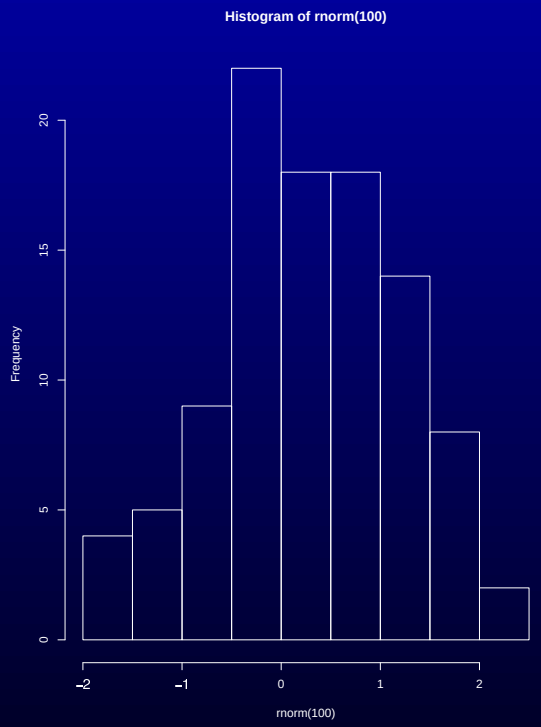
Side-by-Side Graphics (Ct'd)

```
\includegraphics[width=0.3\textwidth]{hist}
\includegraphics[width=0.3\textwidth]{lake1}
\includegraphics[width=0.3\textwidth]{whistler}
```



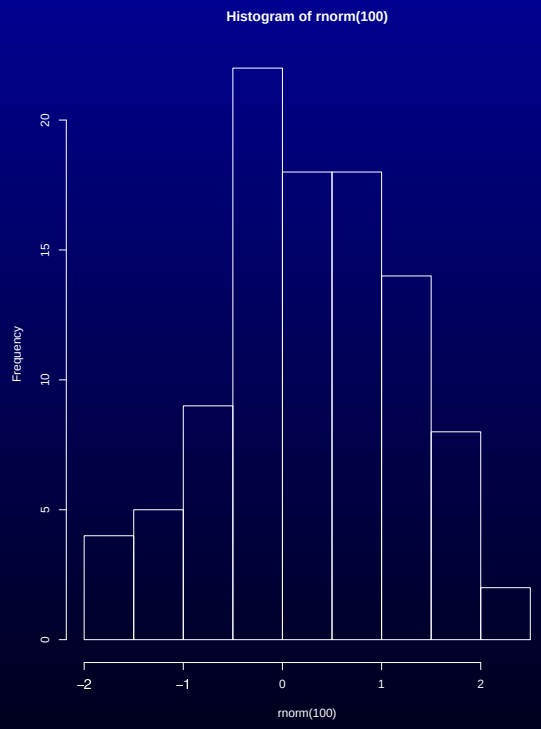
Side-by-Side Graphics (Ct'd)

```
\includegraphics[width=0.3\textwidth]{hist}%
\hspace{0.2cm}%
\includegraphics[width=0.3\textwidth]{lake1}%
\hspace{0.2cm}%
\includegraphics[width=0.3\textwidth]{whistler}
```



Side-by-Side Graphics (Ct'd)

```
\hfill
\includegraphics[width=0.3\textwidth]{hist}%
\hfill%
\includegraphics[width=0.3\textwidth]{lake1}%
\hfill%
\includegraphics[width=0.3\textwidth]{whistler}
\hfill
```



Stacked Graphics

Figure 1: Buntzen lake

Figure 2: Whistler

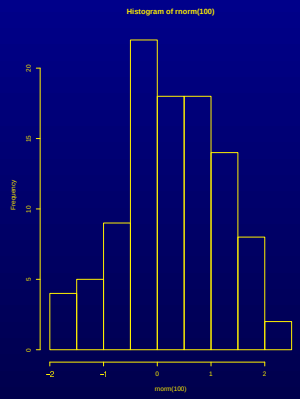


Figure 3: histogram

Syntax of minipage

```
\begin{minipage}[position]{width}
 text
\end{minipage}
```

The code for stacked graphics using minipage:

```
\begin{figure}
    \begin{minipage}[c]{0.5\textwidth}
      \centering
      \includegraphics[height=0.1\textheight]{lake1}
      \caption{Buntzen lake}
    \end{minipage}%
    \begin{minipage}[c]{0.5\textwidth}
      \centering
      \includegraphics[height=0.2\textheight]{whistler}
      \caption{Whistler}
    \end{minipage}\\[0.2pt]
    \begin{minipage}[c]{\textwidth}
      \centering
      \includegraphics[height=0.3\textheight]{hist}
      \caption{histogram}
    \end{minipage}
\end{figure}
```

figure environment

- syntax

```
\begin{figure}{options}
:
\end{figure}
```

- `options` in figure environment

h *Here*: Place the figure in the text where the figure command is located. This option cannot be executed if there is not enough room remaining on the page.
t *Top*: Place the figure at the top of a page.
b *Bottom*: Place the figure at the bottom of a page.
p *Float Page*: Place the figure at a separate page.

Notes on figure placement

- The default setting is `[tbp]`
- The placement options are always attempted in the order `h-t-b-p`.
- Single-location options `[t], [b], [p], [h]` are problematic.
- To force figures to appear after its occurrence in the text, simply add `\usepackage{flafter}` in the preamble and no command is necessary to activate `flafter`.

caption and label

- `\caption` command usually have to be used in `figure` and `table` environments.
- when labeling figures or tables, the `\label` command should be after the `\caption` command.
- You can customize captions with `caption2` package. You can change
  - caption style (e.g. flushleft).
  - caption fontsize (e.g. small)
  - caption label font shape (e.g. italic). **Does not affect caption text**.
  - caption label font series (e.g. bold). **Does not affect caption text**.
  - caption label font family (e.g. typewriter) **Does not affect caption text**.
  - caption delimiter (e.g. use `Figure 1.` instead of `Figure 1:`)

Usage of captionstyle

- global effect. \usepackage[flushleft]{caption2}
- local environment effect.

```
\begin{figure}
    \begin{minipage}[b]{0.5\textwidth}
      \captionstyle{flushleft}
      \onelinecaptionsfalse
      \centering
      \includegraphics[height=0.4\textheight]{lake1}
      \caption{Buntzen lake}
    \end{minipage}%
    \begin{minipage}[b]{0.5\textwidth}
      \renewcommand{\captionlabeldelim}{$>>$~}
      \centering
      \includegraphics[height=0.4\textheight]{whistler}
      \caption{Whistler}
    \end{minipage}
\end{figure}
```

Figure 4: Buntzen lake

Figure 5>> Whistler

Change of Caption font

- global effect. `\usepackage[scriptsize, sl, bf, sf]{caption2}`
- local environment effect.
  - ⋆ The `\captionfont` sets the font for the caption label *and* the caption text.
  - ⋆ The `\captionlabelfont` sets the font for *only* the caption label.

```
\begin{figure}
    \begin{minipage}[b]{0.5\textwidth}
      \renewcommand{\captionfont}{\Large \bfseries \sffamily}
      \centering
      \includegraphics[height=0.4\textheight]{lake1}
      \caption{Buntzen lake}
    \end{minipage}%
    \begin{minipage}[b]{0.5\textwidth}
      \centering
      \includegraphics[height=0.4\textheight]{whistler}
      \caption{Whistler}
    \end{minipage}
\end{figure}
```

Figure 6: Buntzen lake

Figure 7: Whistler

Non-Floating Figures

Directly use `\includegraphics` command without `figure` environment. But cannot have captions and labels.

To have captions and labels, redefine `\@captype` command.

- Add the following in the preamble:

```
\makeatletter
  \newcommand\figcaption{\def\@captype{figure}\caption}
  \newcommand\tabcaption{\def\@captype{table}\caption}
\makeatother
```

- The `minipage` environment is needed to prevent a page break between within the figure.
- The `\\[\intextsep]` commands start new lines and add vertical space before and after the figure.
- The `\FloatBarrier` command forces all floating figures to be processed to avoid the problem that figures do not appear in numerical order. To use `\FloatBarrier` command, you need to add `\usepackage[below]{placeins}` in the preamble.

An Example

```
Text before Figure \ref{nonfloathist}
\\[\intextsep]
  \begin{minipage}{\textwidth}
    \renewcommand{\captionfont}{\footnotesize}
    \centering
    \FloatBarrier
    \includegraphics[height=0.5\textheight]{hist}%
    \figcaption{This is a non-floating figure}
    \label{nonfloathist}
  \end{minipage}
\\[\intextsep]
Text after Figure \ref{nonfloathist}
```

Text before Figure 8

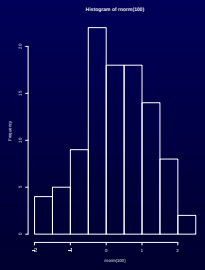


Figure 8: This is a non-floating figure

Text after Figure 8

subfigure package

Add `\usepackage{subfigure}` in the preamble.

Example:

Subfigure 9(a) and Subfigure 9(b) in Figure 9.

(a) Buntzen lake

(b) Whistler

Figure 9: Minipages Inside Subfigures

Add \verb+\usepackage{subfigure}+ in the preamble.

Example:

```
Subfigure \ref{lake1} and Subfigure \ref{whistler}
  in Figure \ref{mini:subfigure}.
\begin{figure}
  \subfigure[Buntzen lake]{\label{lake1} % label for first subfigure
    \begin{minipage}[b]{0.5\textwidth}
      \centering
      \includegraphics[width=\textwidth]{lake1}
    \end{minipage}}%
  \subfigure[Whistler]{\label{whistler} % label for second subfigure
    \begin{minipage}[b]{0.5\textwidth}
      \centering
      \includegraphics[width=\textwidth]{whistler}
    \end{minipage}}
  \caption{Minipages Inside Subfigures}
  \label{mini:subfigure} % label for entire figure
\end{figure}
```

Landscape Figures

package `\usepackage{lscape}`

environment `\begin{landscape}...\end{landscape}`

effects

- places landscape pages in a portrait document.
- the landscape pages are rotated such that the left edge of the portrait page is the top edge of the landscape page.
- the landscape figure is placed on a separate page.
- can produce landscape pages containing any combination of text, tables, and figures.
- multiple landscape pages can be produced.
- figures produced cannot float. However, figures can float within the landscape pages.

```
\begin{landscape}
This page shows a landscape figure
  \footnote{this is footnote}.
  \begin{figure}
    \centering
    \includegraphics[width=0.5\textwidth]{hist}
    \caption{Landscape Figure}
  \end{figure}
\end{landscape}
```

The above code produce a rotated page. Since the default foil is landscape, we get portrait figure when we use `landscape` environment. The header and footer do not change. However the footnote is rotated.

This page shows a landscape figure[1].

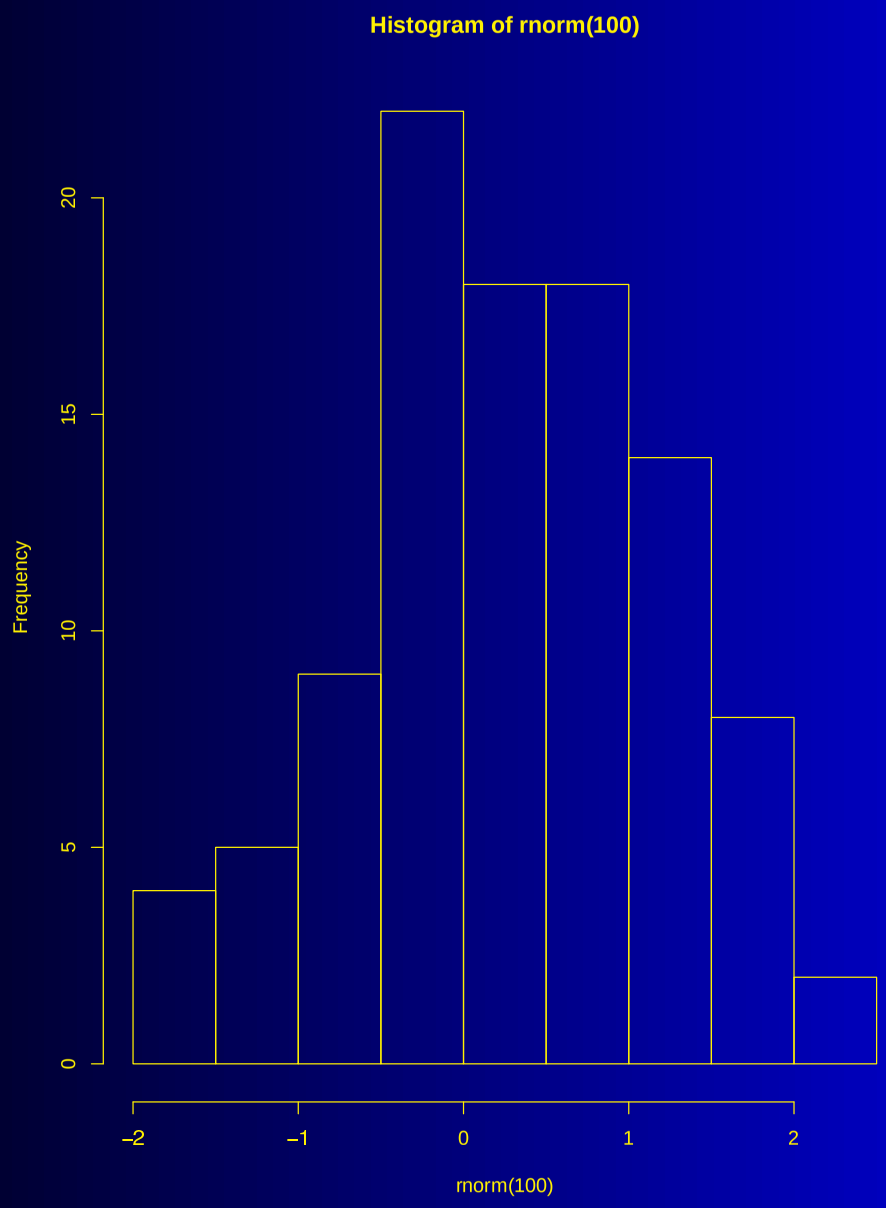


Figure 10: Landscape Figure

[1]this is footnote

What topics we already talked?

How to import graphics into LaTeX?

- `\includegrapics` command
- tips on resizing and rotating figures without distortion in LaTeX
- figure formats can be imported into LaTeX
- creating side-by-side and stacked graphics in LaTeX
- using the “minipage” environment to fit graphics and text within a single page more effectively
- customize captions
- landscape figures

The second part of the workshop

How to use R to generate nice graphics?

The second part of the workshop

How to use R to generate nice graphics?

- controlling graphics parameters

The second part of the workshop

How to use R to generate nice graphics?

- controlling graphics parameters
- adding legends and text to plots

The second part of the workshop

How to use R to generate nice graphics?

- controlling graphics parameters
- adding legends and text to plots
- using “plotmath” to do LaTeX-like typesetting in R

The second part of the workshop

How to use R to generate nice graphics?

- controlling graphics parameters
- adding legends and text to plots
- using “plotmath” to do LaTeX-like typesetting in R
- using the “lattice” package for trellis graphics

The second part of the workshop

How to use R to generate nice graphics?

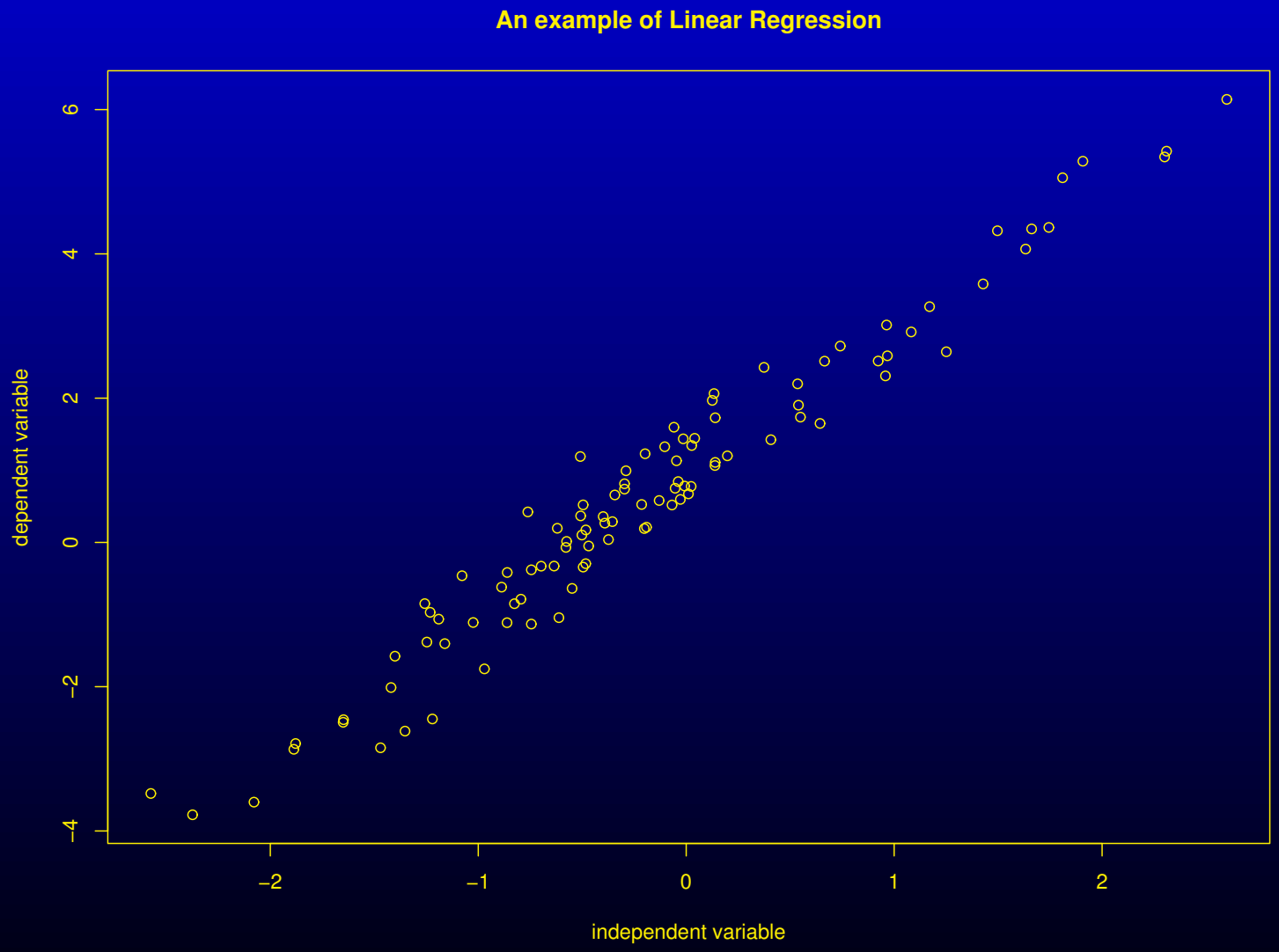
- controlling graphics parameters
- adding legends and text to plots
- using “plotmath” to do LaTeX-like typesetting in R
- using the “lattice” package for trellis graphics
- setting “layout”

The second part of the workshop

How to use R to generate nice graphics?

- controlling graphics parameters
- adding legends and text to plots
- using “plotmath” to do LaTeX-like typesetting in R
- using the “lattice” package for trellis graphics
- setting “layout”
- exporting graphics from R

A simple example



A simple example (Ct'd)

```
x<-rnorm(100)
y<-1+2*x+rnorm(100, mean=0, sd=0.5)
plot(x, y, xlab="independent variable",
     ylab="dependent variable",
     xlim=range(x), ylim=range(y), type="p")
title("An example of Linear Regression")
```

Options of `plot`

x, y the x and y coordinates of points in the plot.

xlab, ylab labels for the x and y axes respectively.

xlim, ylim the ranges to be encompassed by the x and y axes.

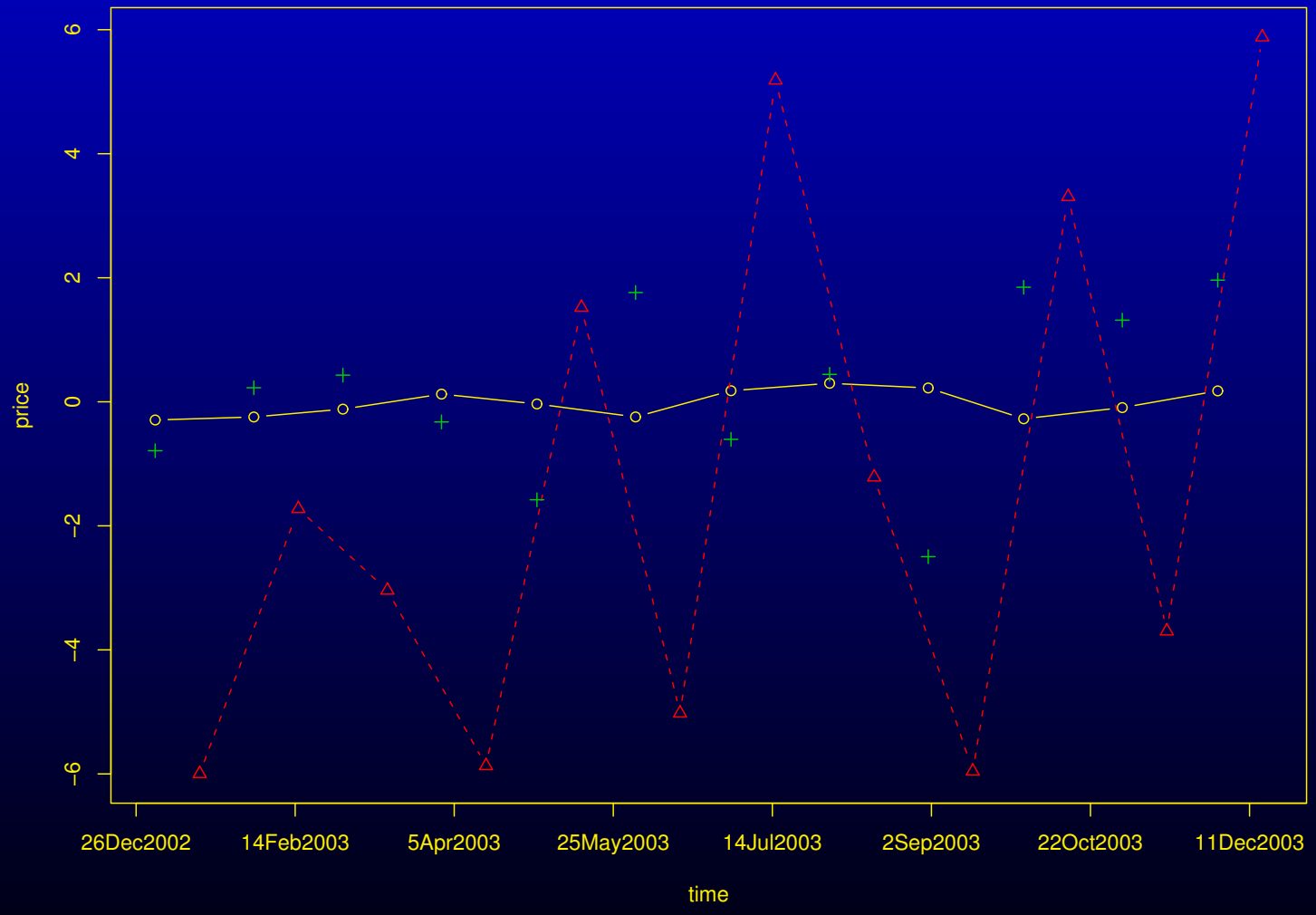
type

- **p** points
- **l** lines
- **b** both
- **s** stair steps

main an overall title for the plot (on top).

sub sub-title (at bottom).

Adding points, lines



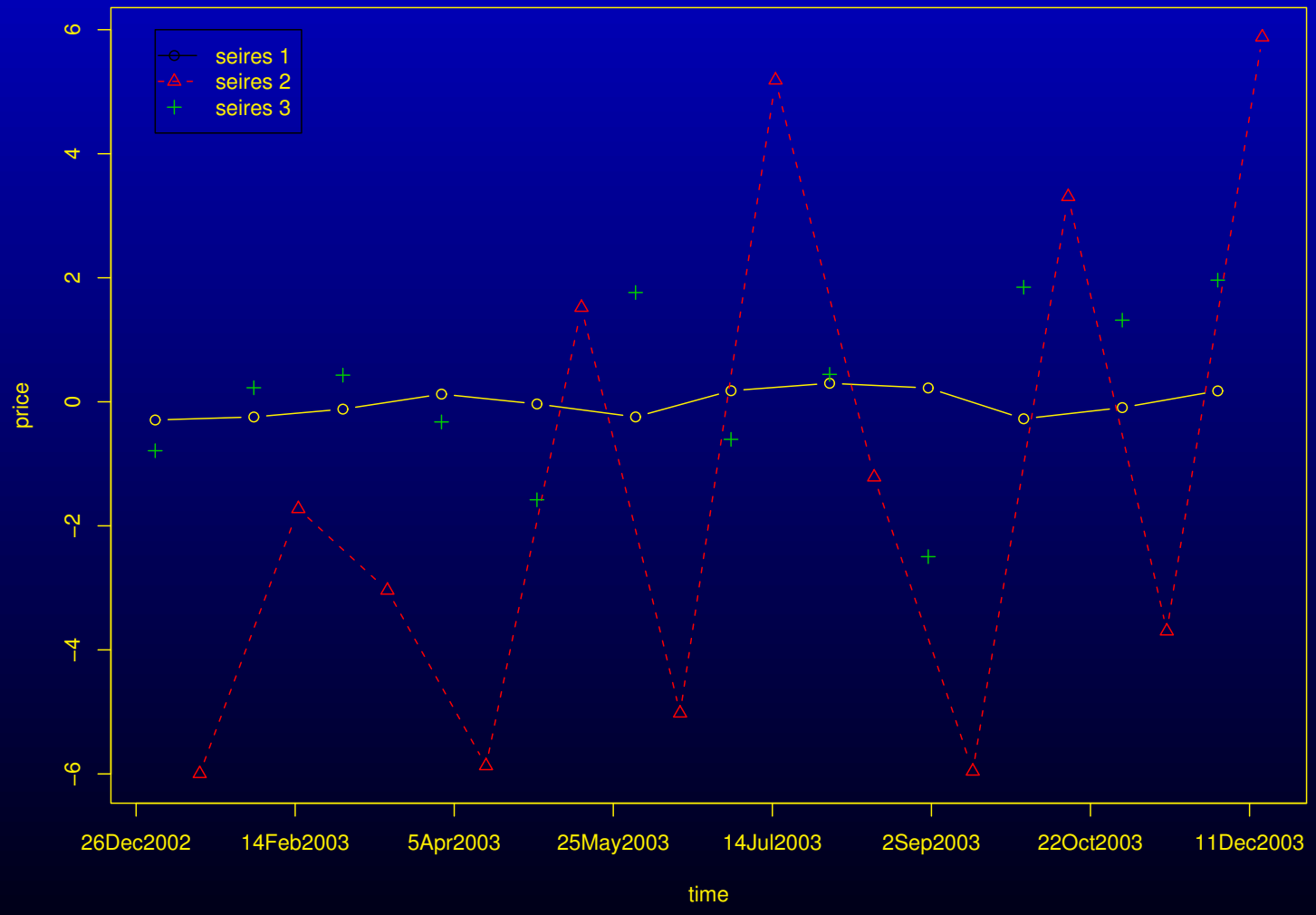
Adding points, lines (Ct'd)

```
library(survival)
myt<-as.date(paste(1:12,1, "2003", sep="/"))
myt2<-as.date(paste(1:12,15, "2003", sep="/"))
x1<-0.3*cos(2*myt+1.0)
x2<-6*cos(3*myt+9.0)
x3<-rnorm(12)
plot(myt, x1,type="b", xlab="time", ylab="price",
      xlim=range(c(myt,myt2)), ylim=range(c(x1,x2,x3)),
      pch=1, col=1,lty=1)
lines(myt2, x2, pch=2, col=2,type="b",lty=2)
points(myt, x3, pch=3, col=3)
```

Note that `myt` and `myt2` are different.

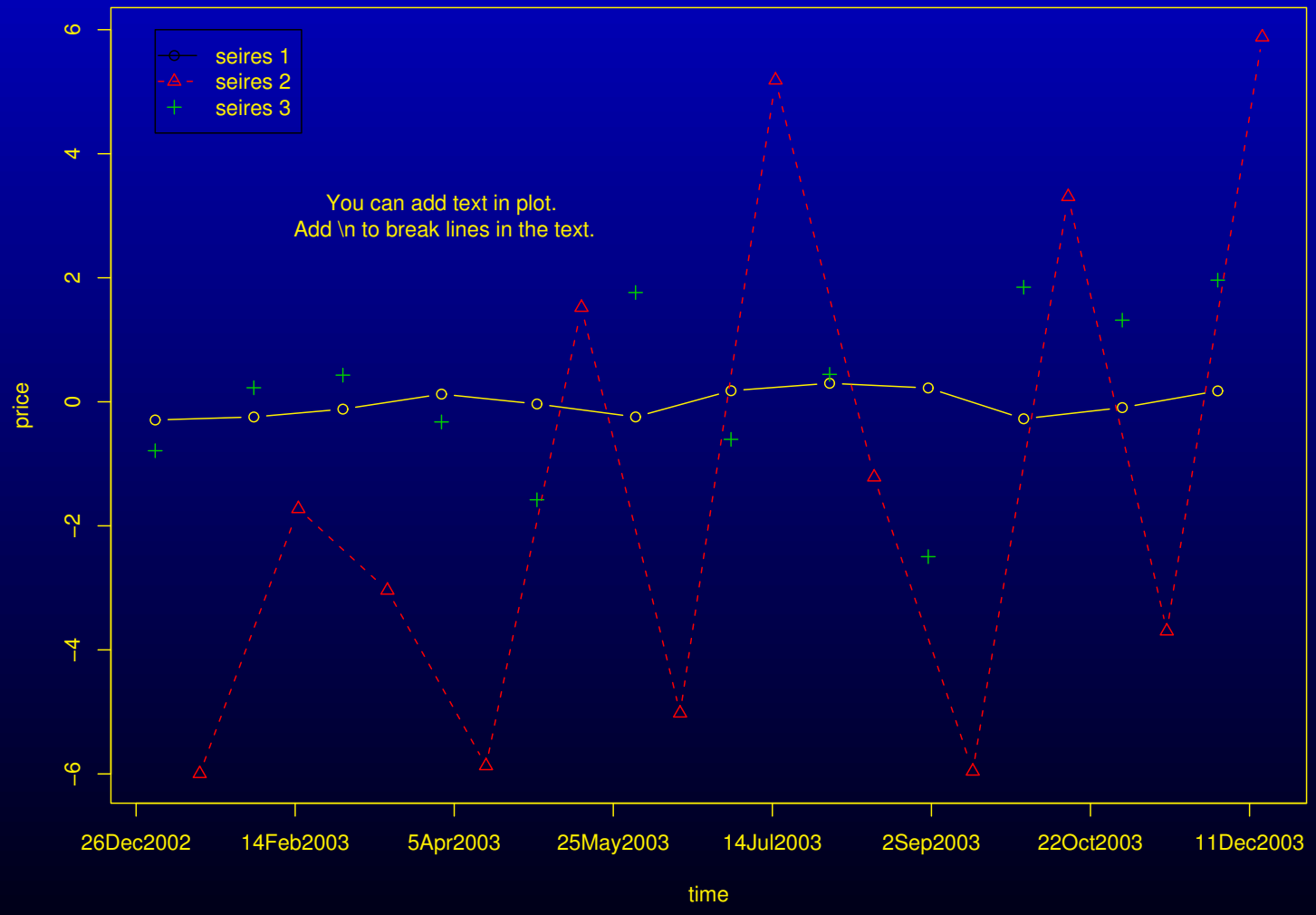
Function legend()

```
legend(x=myt[1],y=6,legend=paste("series", 1:3), pch=1:3,
       lty=c(1,2,0),col=1:3)
```

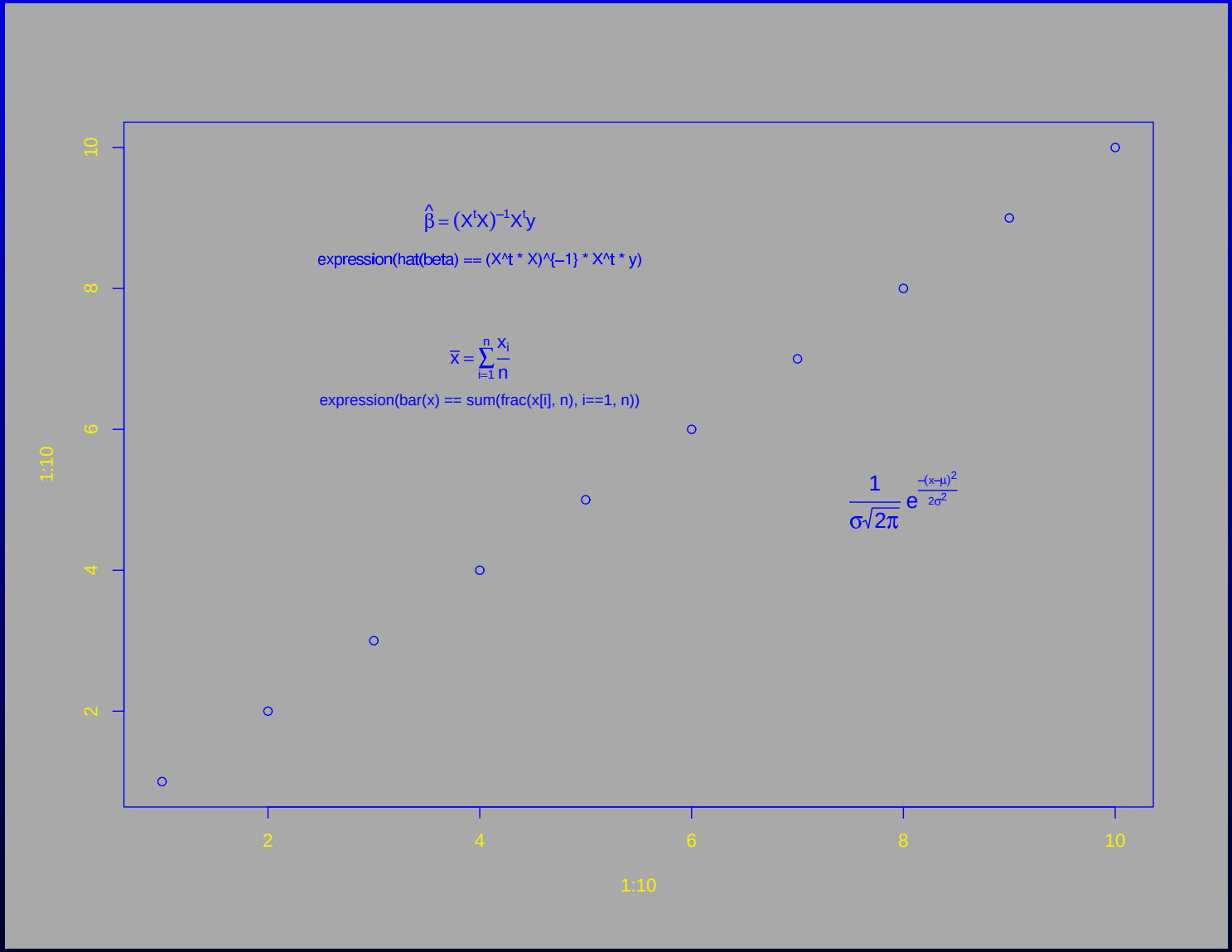


Function text()

```
text(x=myt[4], y=3,labels="You can add text in plot.\n
  Add \\n to break lines in the text.")
```



Displaying Math Symbol in R Graphics



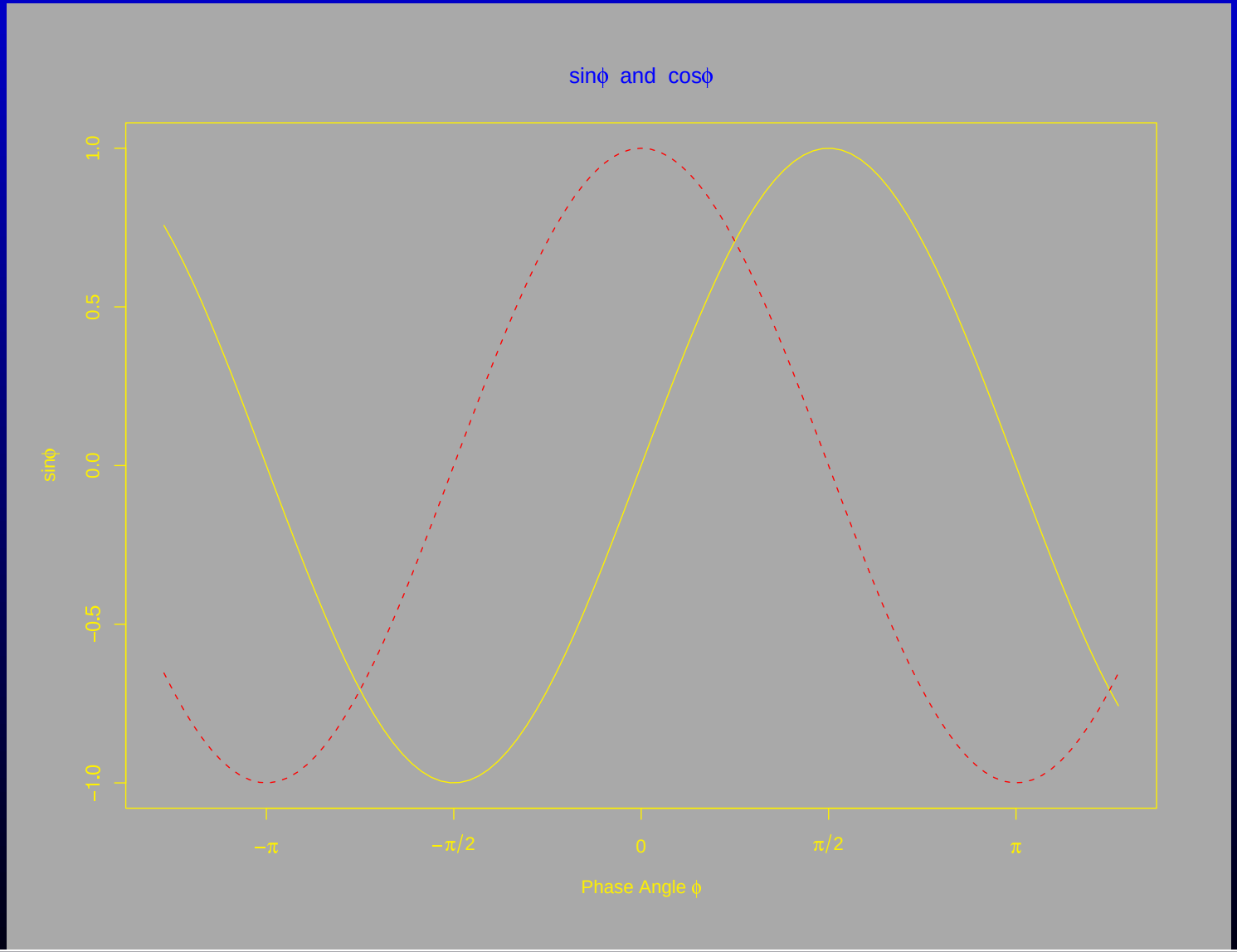
Displaying Math Symbol in R Graphics (Ct'd)

```
plot(1:10, 1:10)
text(4, 9, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
text(4, 8.4, "expression(hat(beta) == (X^t * X)^{-1} * X^t * y)",
     cex = .8)
text(4, 7, expression(bar(x) == sum(frac(x[i], n), i==1, n)))
text(4, 6.4, "expression(bar(x) == sum(frac(x[i], n), i==1, n))",
     cex = .8)
text(8, 5, expression(paste(frac(1, sigma*sqrt(2*pi)), " ",
      plain(e)^{frac(-(x-mu)^2, 2*sigma^2)})), cex= 1.2)
```

Displaying Math Symbol in R Graphics (Ct'd)

Syntax	Meaning
'x*y'	juxtapose x and y
'x/y'	x forwardslash y
'x %+-% y'	x plus or minus y
'x %/% y'	x divided by y
'x %*% y'	x times y
'x[i]'	x subscript i
'x^2'	x superscript 2
'paste(x, y, z)'	juxtapose x, y, and z
'sqrt(x)'	square root of x
'x == y'	x equals y
'plain(x)'	draw x in normal font
'bold(x)'	draw x in bold font
'italic(x)'	draw x in italic font
'hat(x)'	x with a circumflex
'bar(xy)'	xy with bar
'widehat(xy)'	xy with a wide circumflex
'alpha' - 'omega'	Greek symbols
'frac(x, y)'	x over y
'sum(x[i], i==1, n)'	sum x[i] for i equals 1 to n
'prod(plain(P)(X==x), x)'	product of P(X=x) for all values of x
'lim(f(x), x %->% 0)'	limit of f(x) as x tends to 0

Displaying Math Symbol in R Graphics (Ct'd)



Displaying Math Symbol in R Graphics (Ct'd)

```
x <- seq(-4, 4, len = 101)
y <- cbind(sin(x), cos(x))
matplot(x, y, type = "l", xaxt = "n",
        main = expression(paste(plain(sin) * phi, "  and  ",
                                plain(cos) * phi)),
        ylab = expression("sin" * phi),
        xlab = expression(paste("Phase Angle ", phi)),
        col.main = "blue")
axis(1, at = c(-pi, -pi/2, 0, pi/2, pi),
     lab = expression(-pi, -pi/2, 0, pi/2, pi))
```

xaxt is an option in the function `matplot`. It is a character which specifies the axis type. Specifying "n" causes an axis to be set up, but not plotted.

axis is an function in R which adds an axis to the current plot, allowing the specification of the side, position, labels, and other options.

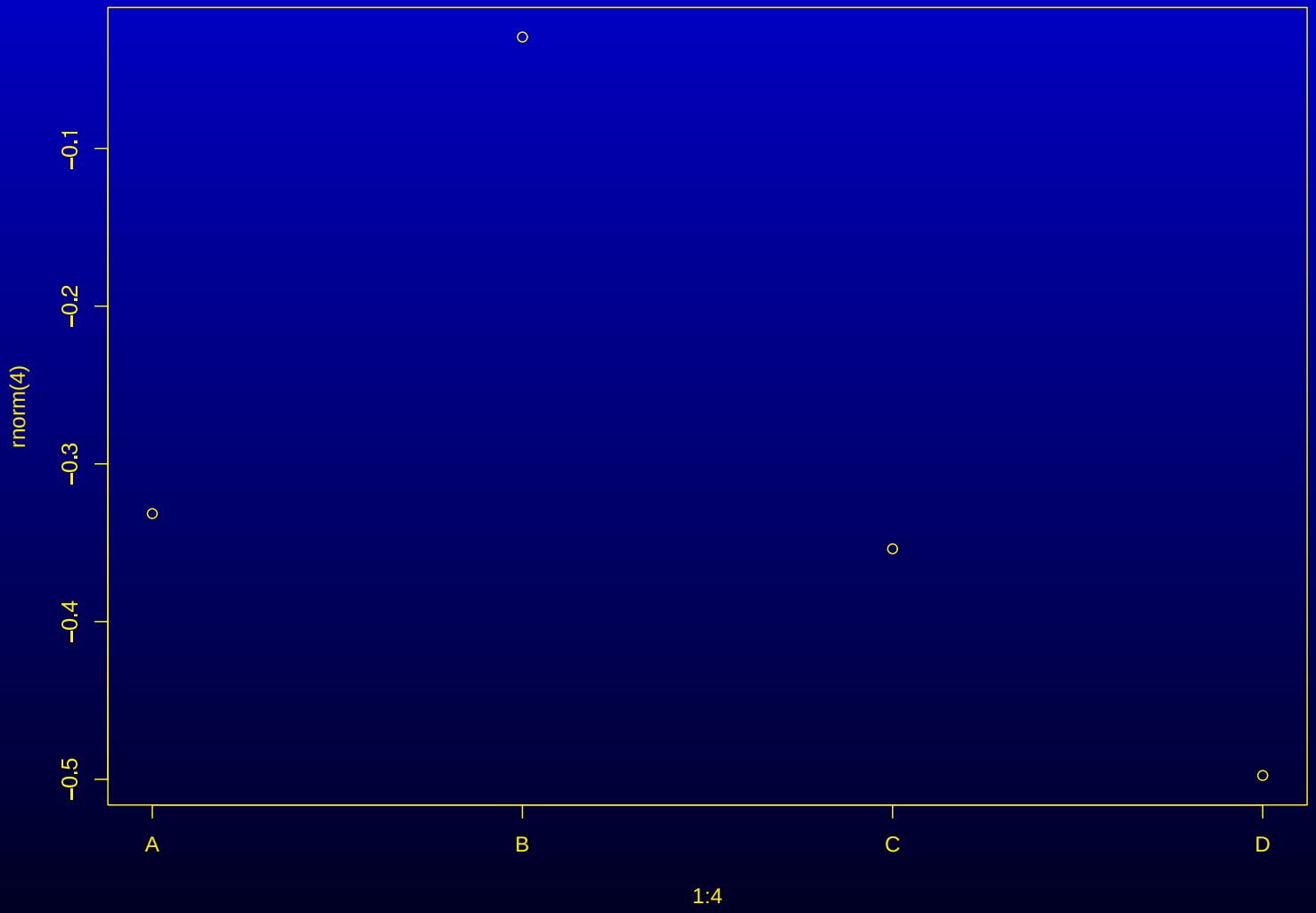
Function `axis`

```
axis(side, at, labels, ...)
```

side an integer specifying which side of the plot the axis is to be drawn on. The axis is placed as follows: 1=below, 2=left, 3=above and 4=right.

at the points at which tick-marks are to be drawn.

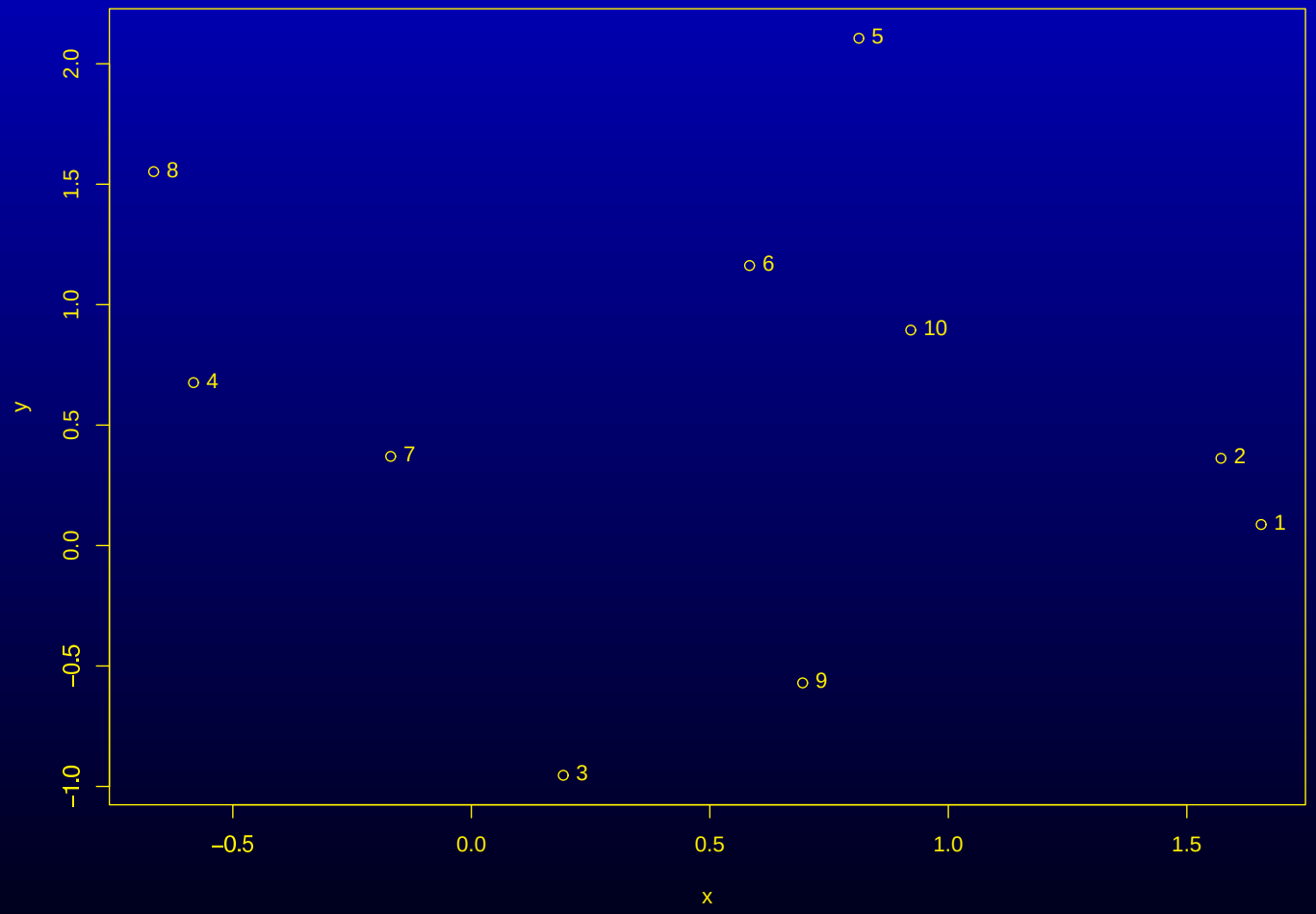
Function `axis` (Ct'd)



Function axis (Ct'd)

```
plot(1:4, rnorm(4), axes=FALSE)
#axes is a logical value indicating whether axes
# should be drawn on the plot.
axis(1, 1:4, LETTERS[1:4])
axis(2)
box() #- to make it look "as usual"
```

Labeling points



Labeling points (Ct'd)

```
library(mva)
x<-mvrnorm(10, mu=c(0,0), Sigma=diag(2))
plot(x, xlab="x", ylab="y")
text(x, pos=4)
```

The option `pos` is a position specifier for the text. If specified this overrides any 'adj' value given. Values of '1', '2', '3' and '4', respectively indicate positions below, to the left of, above and to the right of the specified coordinates.

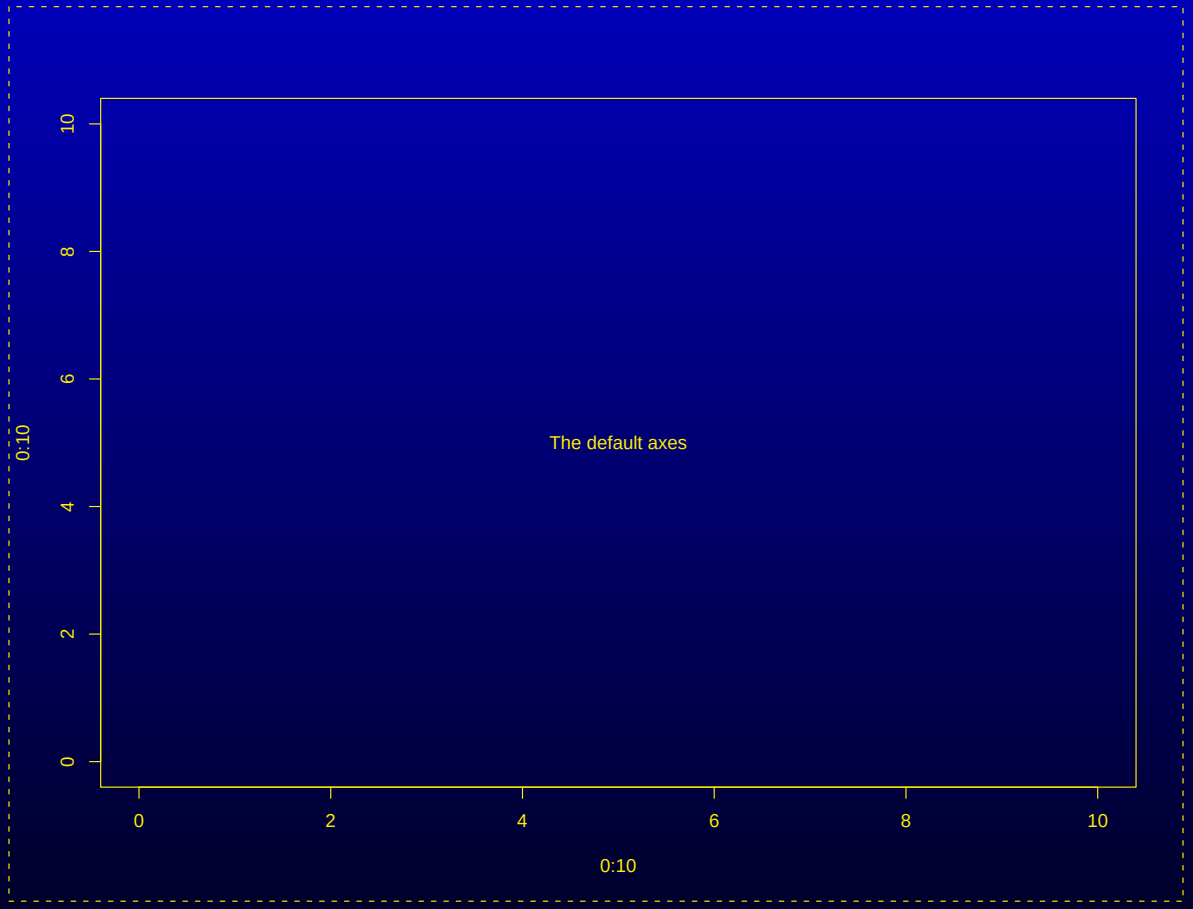
Rotating y axis label

- You can combine the use of the function `mtext` and those of the `las`, `mar` options in the function `par`.
- `mtext` writes text into one of 4 margins of a plot.
- `par` can be used to set or query graphical parameters. Parameters can be set by specifying them as arguments to 'par' in 'tag = value' form, or by passing them as a list of tagged values.

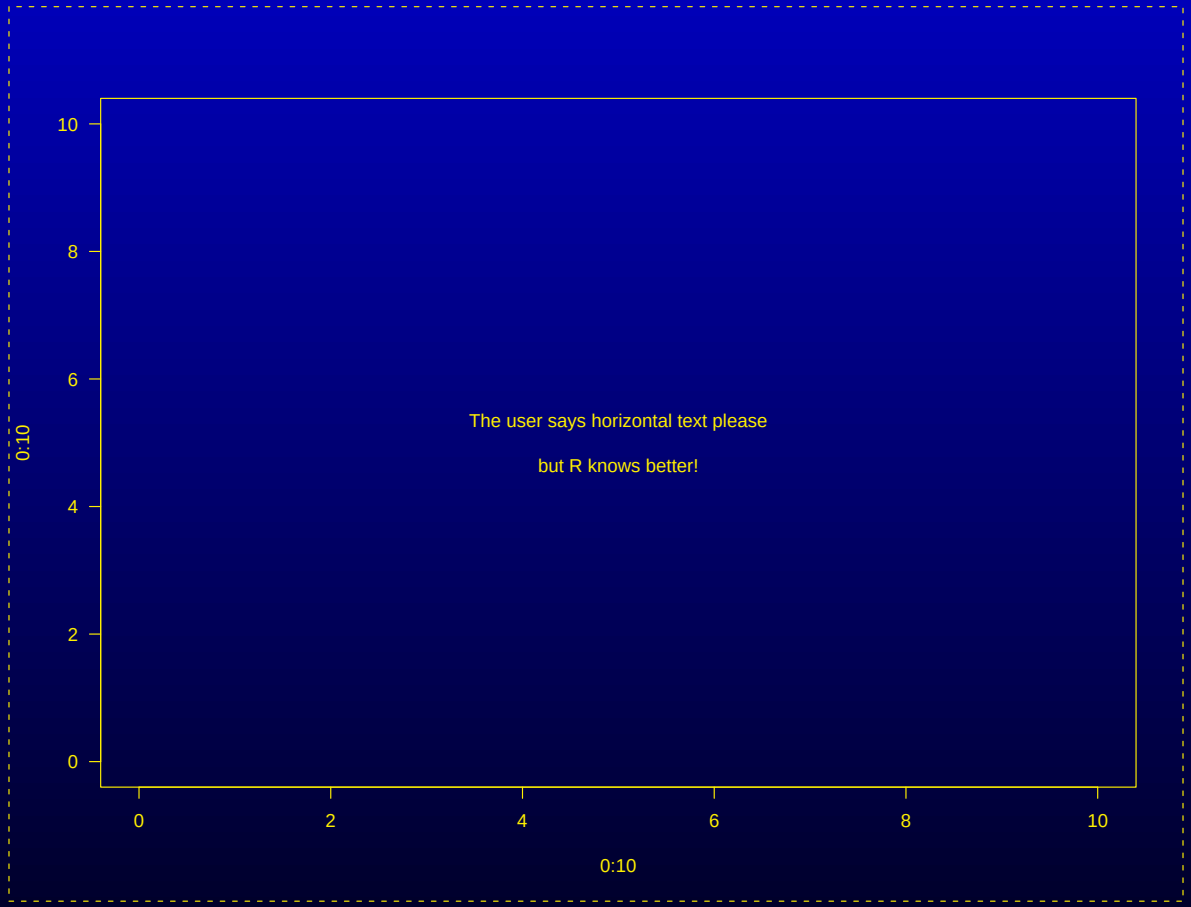
  las numeric in $\{0,1,2,3\}$; the style of axis labels.
  0: always parallel to the axis [default],
  1: always horizontal,
  2: always perpendicular to the axis,
  3: always vertical.

  mar A numerical vector of the form 'c(bottom, left, top, right)' which gives the lines of margin to be specified on the four sides of the plot. The default is 'c(5, 4, 4, 2) + 0.1'.

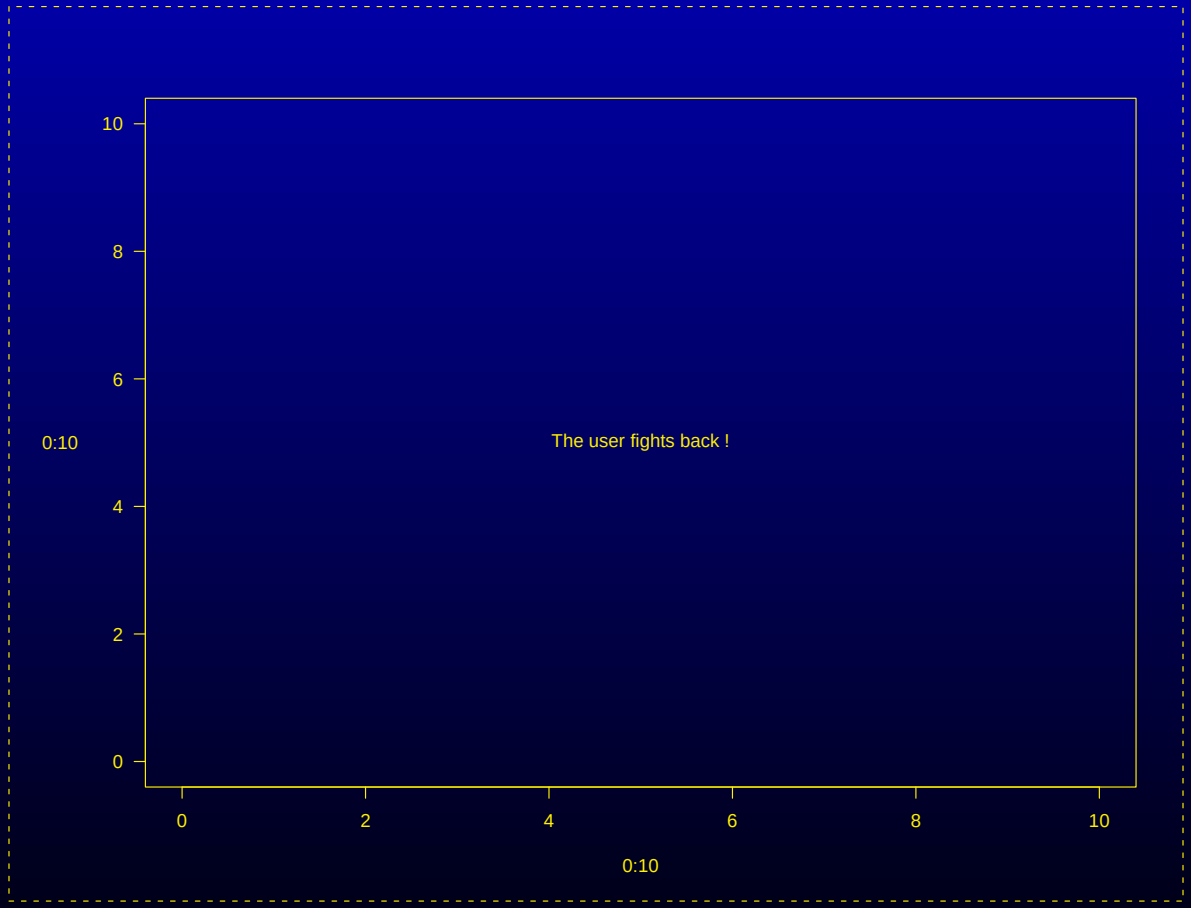
```
par(mfrow=c(1,1), mar=c(5.1, 4.1, 4.1, 2.1), las=0)
plot(0:10, 0:10, type="n")
text(5, 5, "The default axes")
box("figure", lty="dashed")
```



```
par(las=1)
plot(0:10, 0:10, type="n")
text(5,5,"The user says horizontal text please\n\nbut R knows better!")
box("figure", lty="dashed")
```

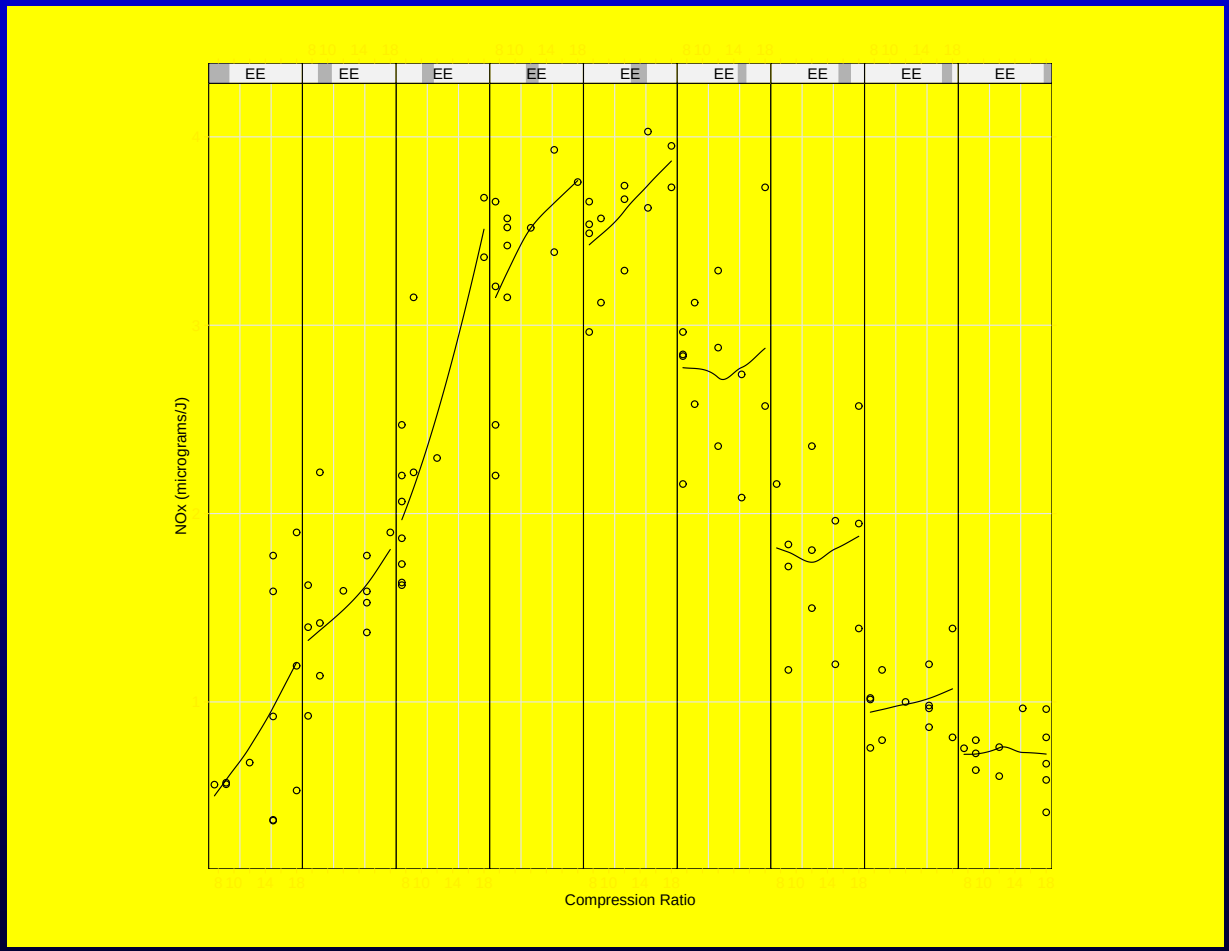


```
par(las=1, mar=c(5.1, 6.1, 4.1, 2.1))
plot(0:10, 0:10, type="n", ann=F)
mtext("0:10", side=2, line=3)
mtext("0:10", side=1, line=3)
text(5, 5, "The user fights back !")
box("figure", lty="dashed")
```



Lattice

Trellis display is a framework for the visualization of multivariable databases.



```
library(lattice)
## Examples with data from 'Visualizing Data' (Cleveland)
## (obtained from Bill Cleveland's Homepage :
## http://cm.bell-labs.com/cm/ms/departments/sia/wsc/, also
## available at statlib)
data(ethanol)
EE <- equal.count(ethanol$E, number=9, overlap=1/4)
## Constructing panel functions on the fly; prepanel
xyplot(NOx ~ C | EE, data = ethanol,
  prepanel = function(x, y) prepanel.loess(x, y, span = 1),
  xlab = "Compression Ratio", ylab = "NOx (micrograms/J)",
  panel = function(x, y) {
      panel.grid(h=-1, v= 2)
      panel.xyplot(x, y)
      panel.loess(x,y, span=1)
  },
  aspect = "xy")
```

layout Function

- `layout` divides the device up into as many rows and columns as there are in matrix 'mat', with the column-widths and the row-heights specified in the respective arguments.

- The usage:

```
layout(mat,
       widths = rep(1, dim(mat)[2]),
       heights= rep(1, dim(mat)[1]),
       respect= FALSE)
layout.show(n = 1)
```

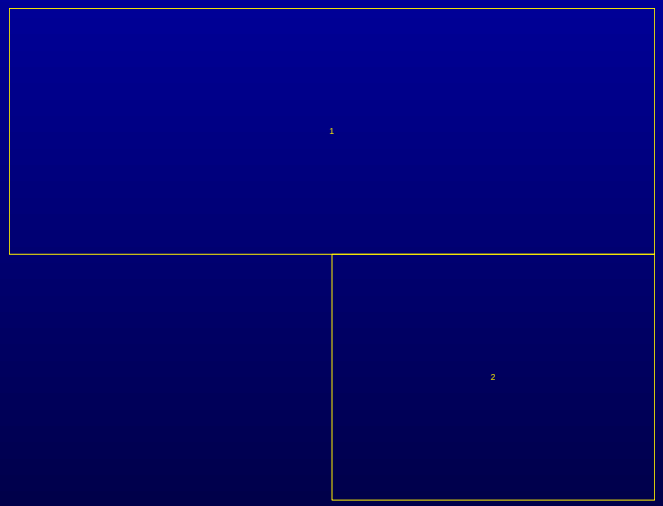
mat a matrix object specifying the location of the next N figures on the output device. Each value in the matrix must be '0' or a positive integer. If N is the largest positive integer in the matrix, then the integers 1,...,N-1 must also appear at least once in the matrix.

widths a vector of values for the widths of columns on the device.

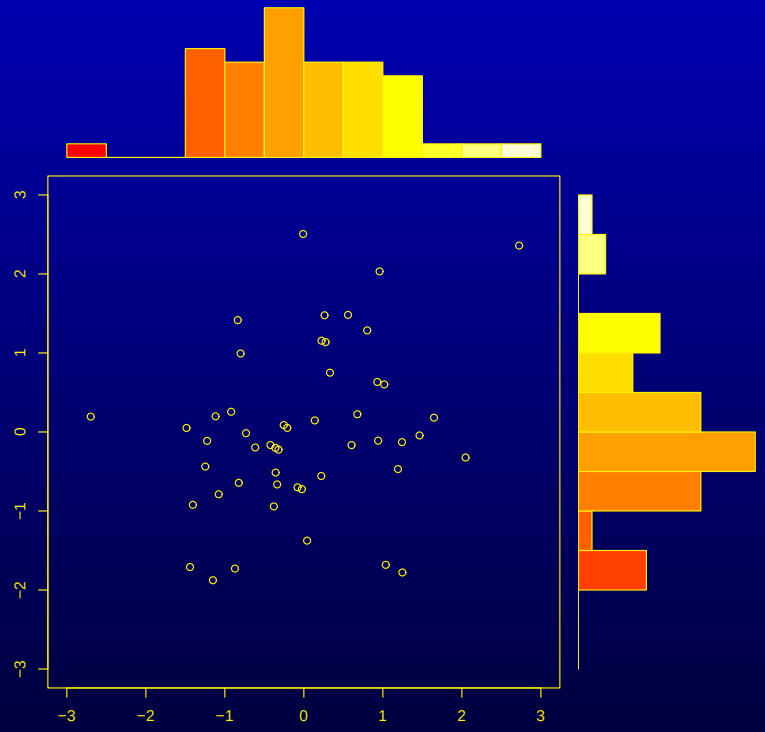
heights a vector of values for the heights of rows on the device.

layout Function (Ct'd)

```
## divide the device into two rows and two columns
## allocate figure 1 all of row 1
## allocate figure 2 the intersection of column 2 and row 2
layout(matrix(c(1,1,0,2), 2, 2, byrow = TRUE))
## show the regions that have been allocated to each plot
layout.show(2)
```



`layout` Function (Ct'd)



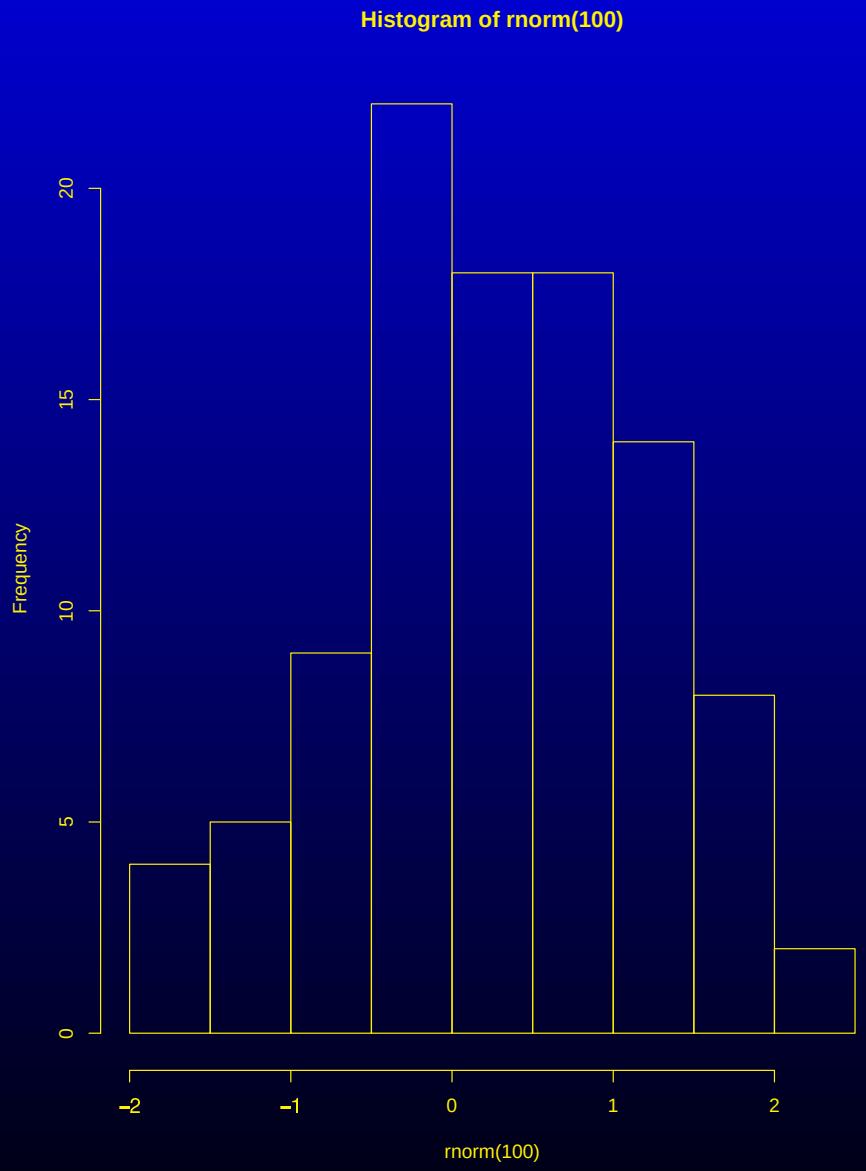
layout Function (Ct'd)

```
-- Create a scatterplot with marginal histograms -----

x <- pmin(3, pmax(-3, rnorm(50)))
y <- pmin(3, pmax(-3, rnorm(50)))
xhist <- hist(x, breaks=seq(-3,3,0.5), plot=FALSE)
yhist <- hist(y, breaks=seq(-3,3,0.5), plot=FALSE)
top <- max(c(xhist$counts, yhist$counts))
xrange <- c(-3,3)
yrange <- c(-3,3)
nf <- layout(matrix(c(2,0,1,3),2,2,byrow=TRUE), c(3,1), c(1,3), TRUE)
layout.show(nf)

par(mar=c(3,3,1,1))
plot(x, y, xlim=xrange, ylim=yrange, xlab="", ylab="")
par(mar=c(0,3,1,1))
barplot(xhist$counts, axes=FALSE, ylim=c(0, top), space=0)
par(mar=c(3,0,1,1))
barplot(yhist$counts, axes=FALSE, xlim=c(0, top), space=0, horiz=TRUE)
```

Exporting Graphics from R

A simple example:

```
postscript("hist.ps")
hist(rnorm(100))
dev.off()
```



Options of `postscript`

```
postscript(file="hist.ps", paper="letter",
           horizontal=T, bg="transparent")
hist(rnorm(100))
dev.off()
```

paper the size of paper in the printer.

horizontal The value `T` means landscape orientation.

bg the default background color to be used. If `transparent` (or an equivalent specification), no background is painted.

Other exporting commands

pdf:
```
# no "horizontal" and "paper" options
pdf(file="hist.pdf",width=6,height=6,bg="transparent")
hist(rnorm(100))
dev.off()
```

jpeg:
```
# no "horizontal" and "paper" options
jpeg(file="hist.jpg",width=6,height=6,bg="transparent")
hist(rnorm(100))
dev.off()
```

Review the second part of the workshop

How to use R to generate nice graphics?

- controlling graphics parameters
- adding legends and text to plots
- using “plotmath” to do LaTeX-like typesetting in R
- using the “lattice” package for trellis graphics
- setting “layout”
- exporting graphics from R