Graphics

Thomas Källman (adopted from slides by M. Kierczak

10/4/2017

Graphics examples

World map with data overlayed

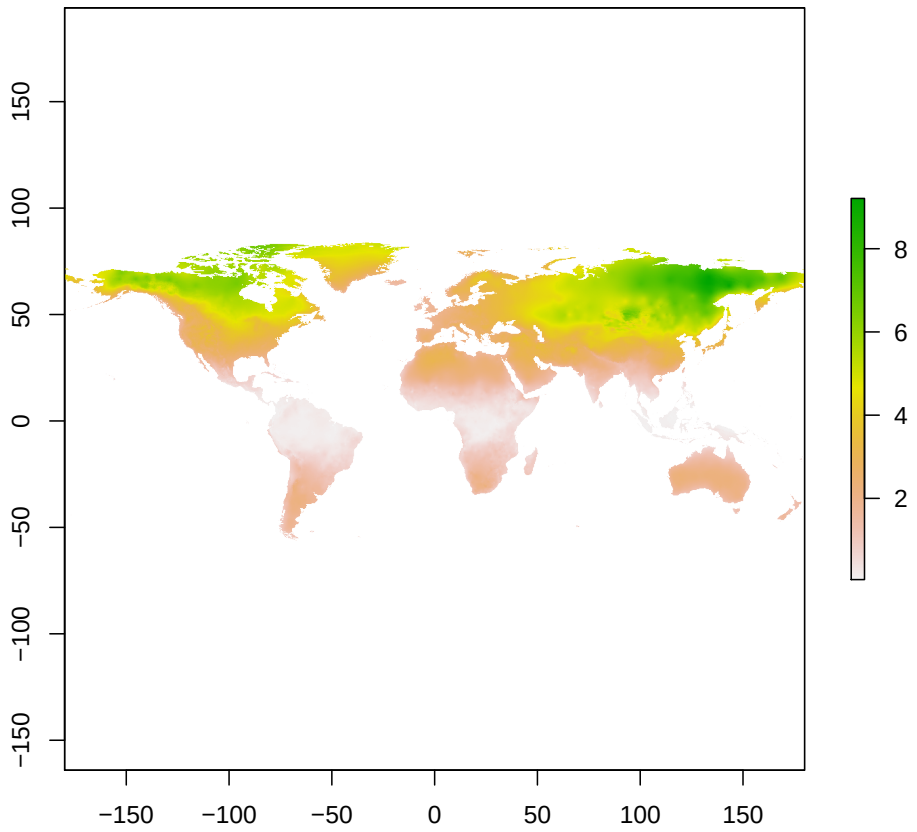


Figure 1: Climate stability (From: M. Pettersson, UU)

Voronoi tesselations on maps

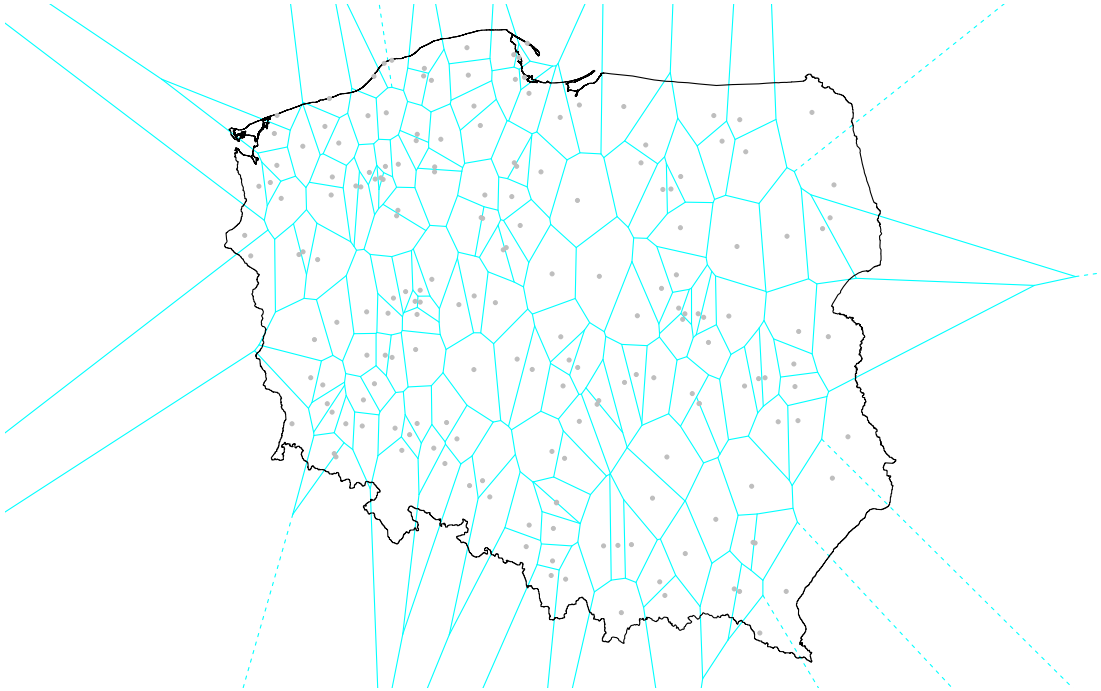


Figure 2: Airport coverage poland

Faces of asia

Figure 3: Use figures to represent data

Gapminder type of plots

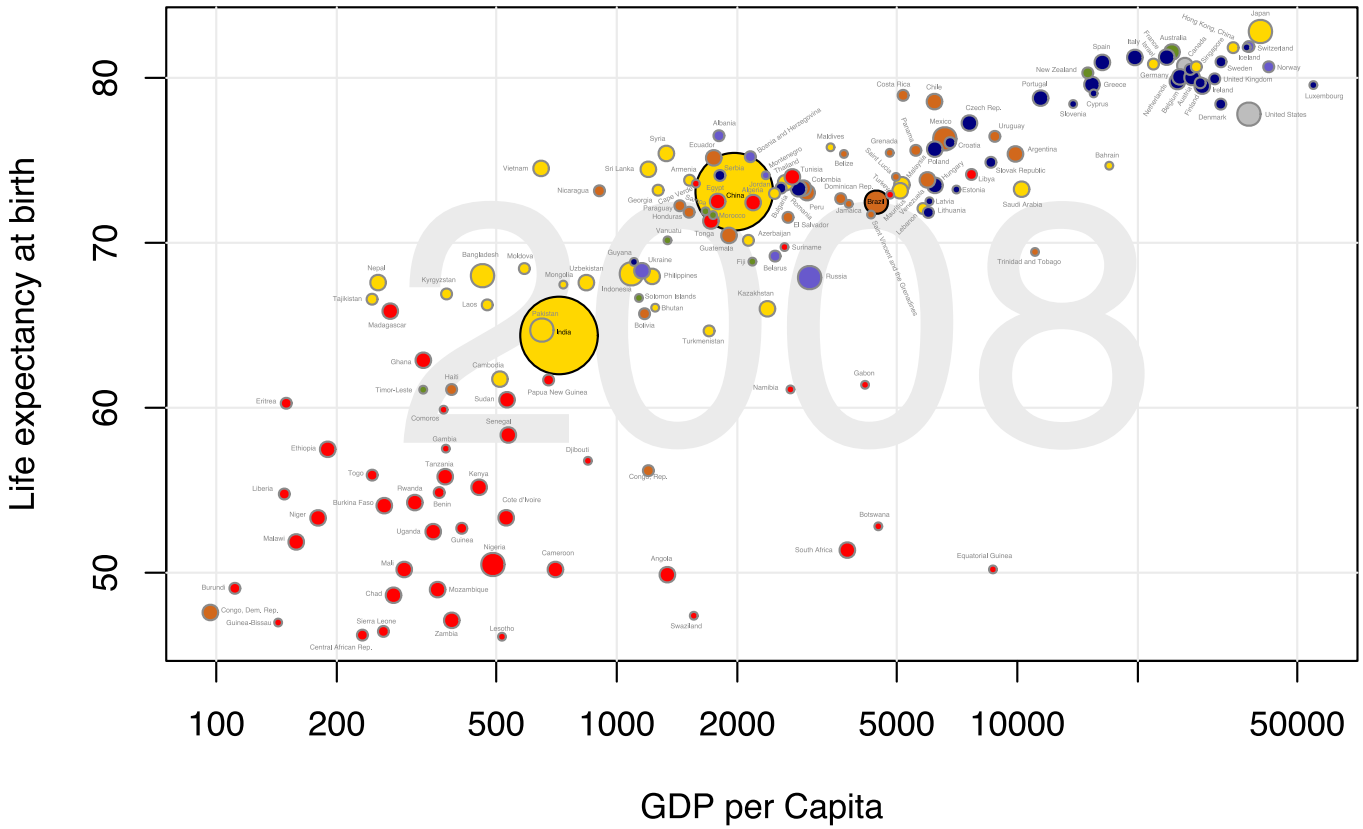


Figure 4: Economy vs expected life span

Circos plots

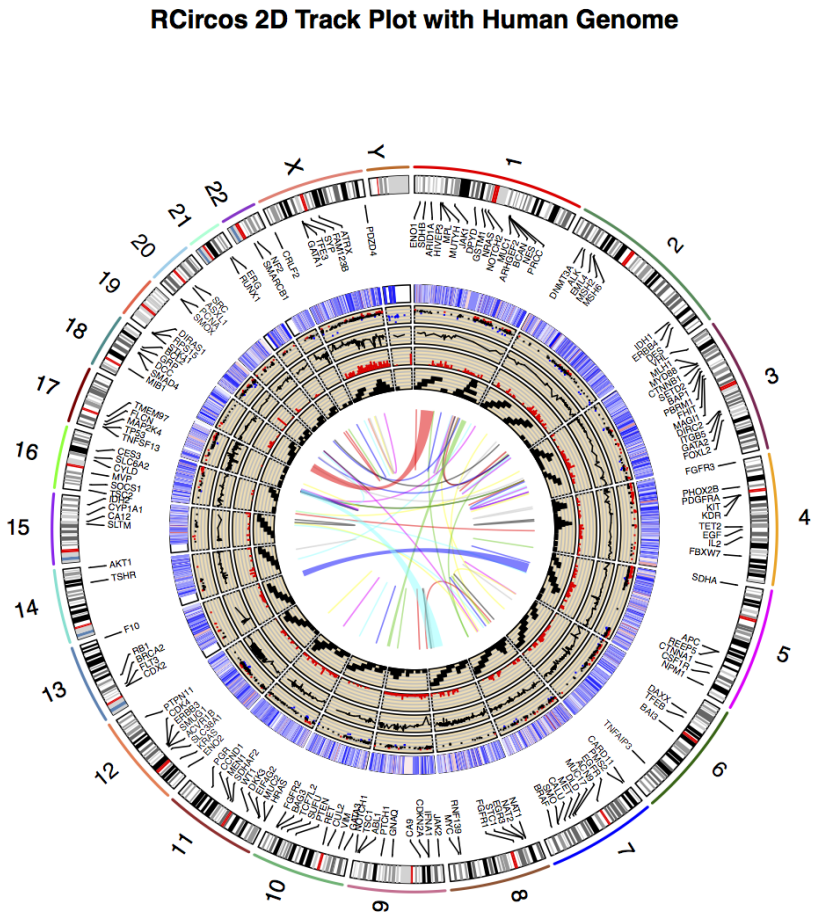


Figure 5: Human genome with metadata

Graphics

Graphical devices in R

The concept of a graphical device is crucial for understanding R graphics. A device can be a screen (default) or a file. Some R packages introduce their own devices, e.g. Cairo device.

Create plots:

- opening a graphical device (not necessary for plotting on screen)
- plotting to the graphical device
- closing the graphical device (very important!)

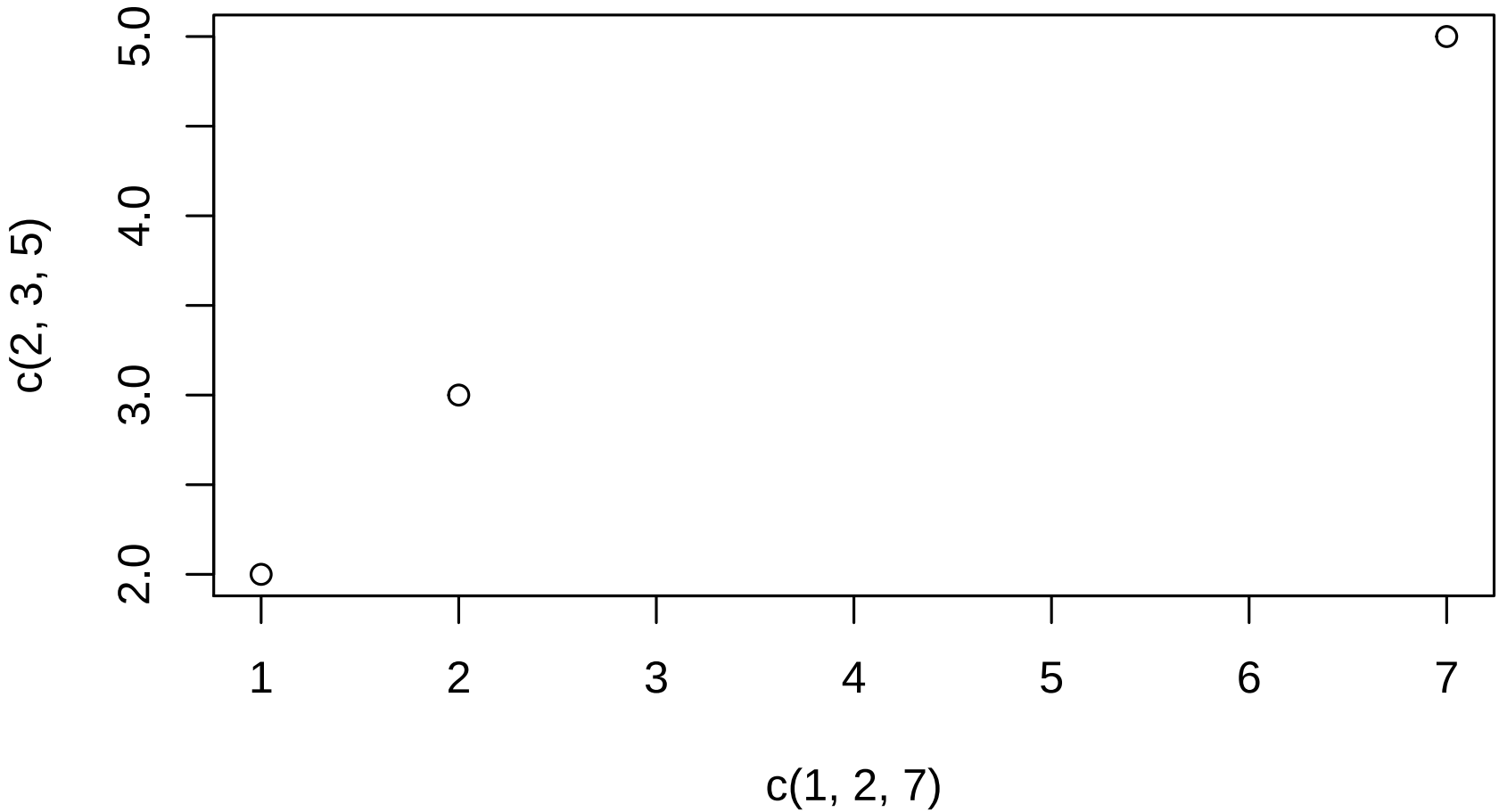
The most common devices

- screen
- bitmap/raster devices: png(), bmp(), tiff(), jpeg()
- vector devices: svg(), pdf(),
- Cairo versions of the above devices – for Windows users they offer higher quality graphics

for more information visit
`http://stat.ethz.ch/R-manual/Rdevel/`
library/grDevices/html/Devices.html

Plotting on screen

```
plot(x=c(1,2,7), y=c(2,3,5))
```



Plotting to file

```
png(filename = 'myplot.png',
    width = 320,
    height = 320,
    antialias = TRUE)
plot(x=c(1,2,7), y=c(2,3,5))
dev.off()
```

Base R graphics viewport

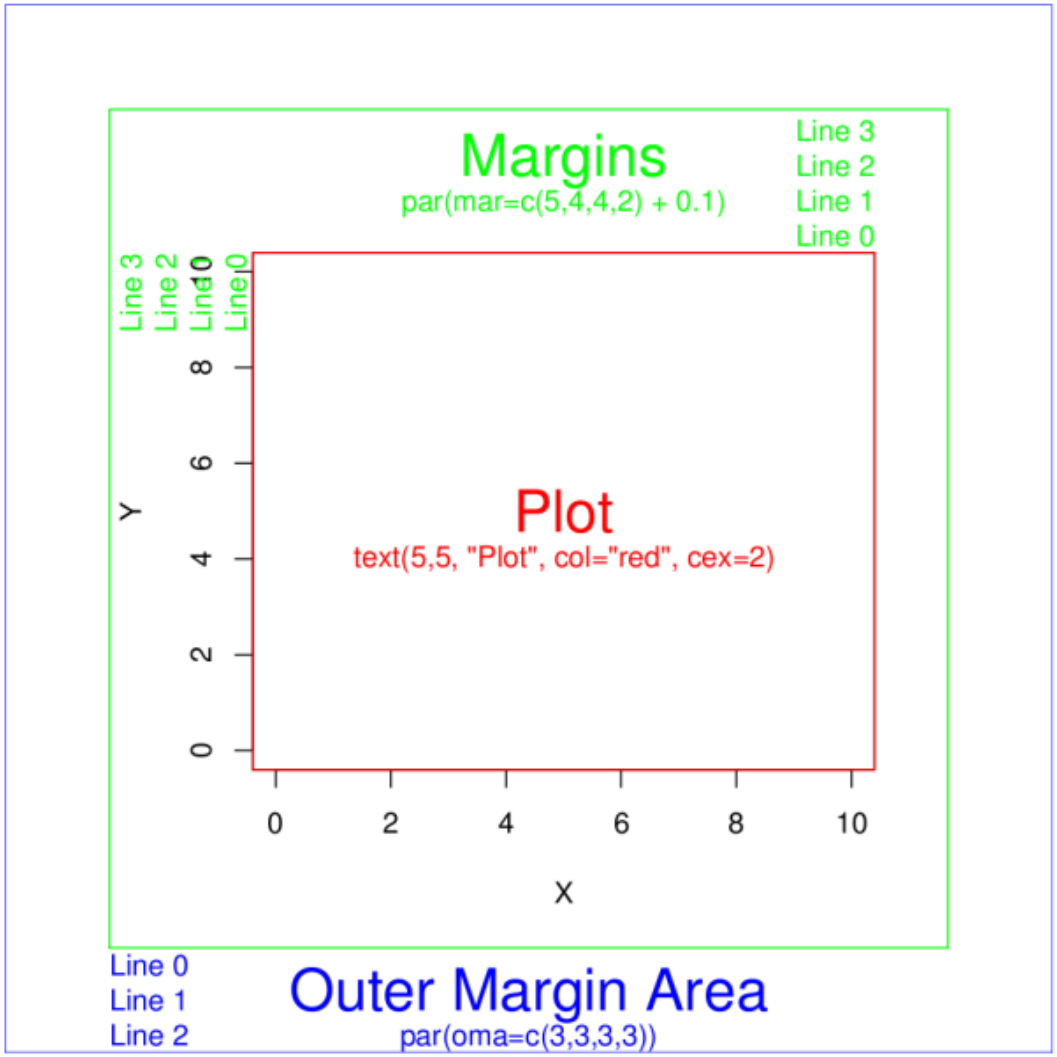


Figure 6: R plotting areas

Changing parameters

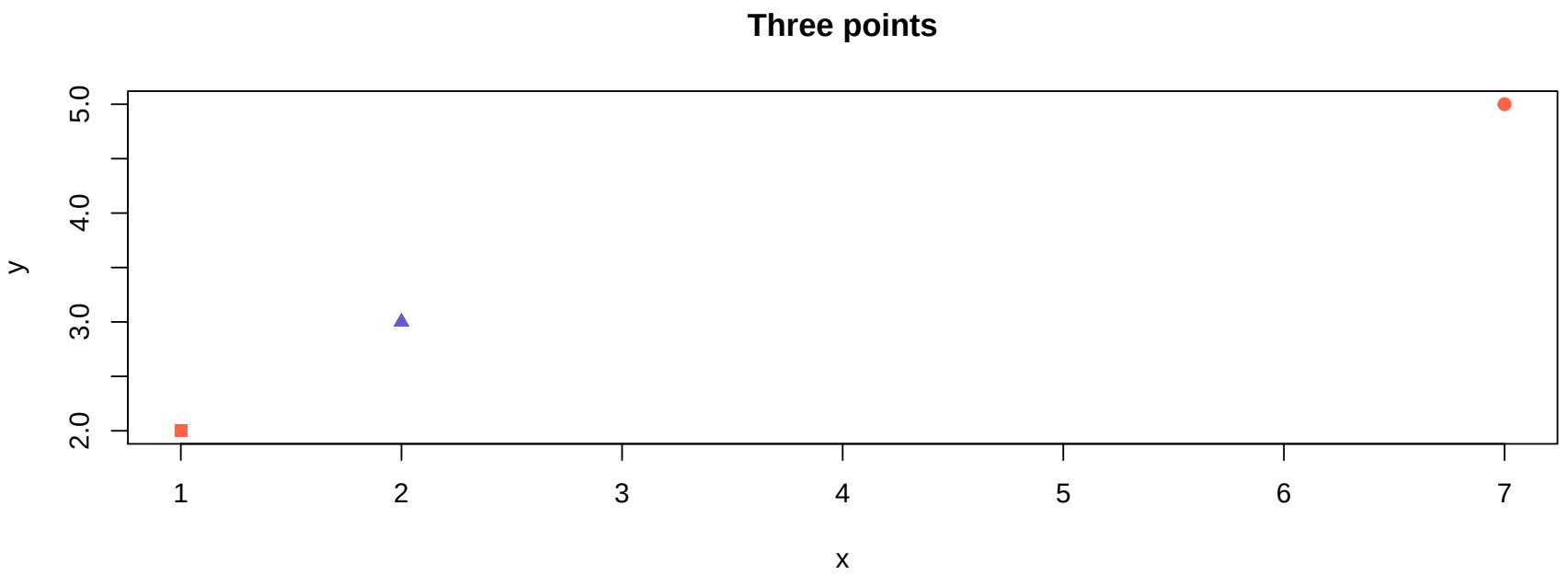
For convience we will create a data frame to hold our data

```
df <- data.frame(x = c(1,2,7),
                 y = c(2,3,5),
                 row.names = c("A","B","C"))
df
```

```
##   x y
## A 1 2
## B 2 3
## C 7 5
```

Changing parameters

```
plot(df, pch=c(15,17,19),
  col=c("tomato", "slateblue"),
  main="Three points")
```



Plotting altering defaults

There is many parameters one can set in plot().

- pch – type of the plotting symbol
- col – color of the points
- cex – scale for points
- main – main title of the plot
- sub – subtitle of the plot
- xlab – X-axis label
- ylab – Y-axis label
- las – axis labels orientation
- cex.axis – axis lables scale

Changing parameters

Graphical parameters can be set in two different ways:

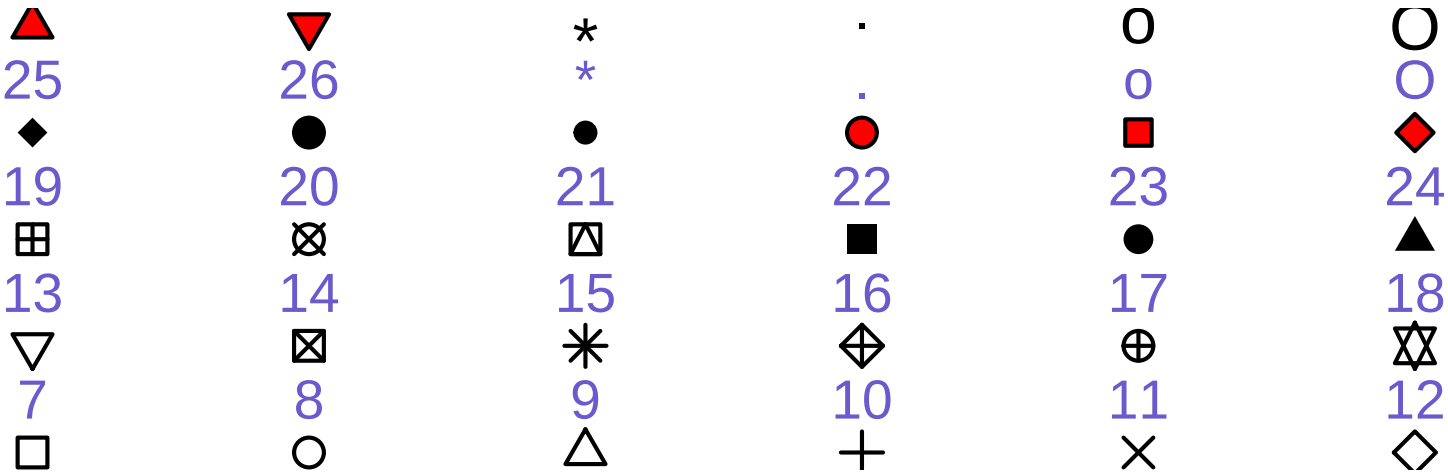
- As plotting function arguments, e.g. plot(dat, cex=0.5)
- using the function par().

```
# read current graphical parameters
par()
# first, save the current parameters so that you
# can set them back if needed
opar <- par() # should work in theory, practise varies :-(
# now, modify what you want
par(cex.axis = 0.8, bg='grey')
# do your plotting
plot(.................)
# restore the old parameters if you want
par(opar)
```

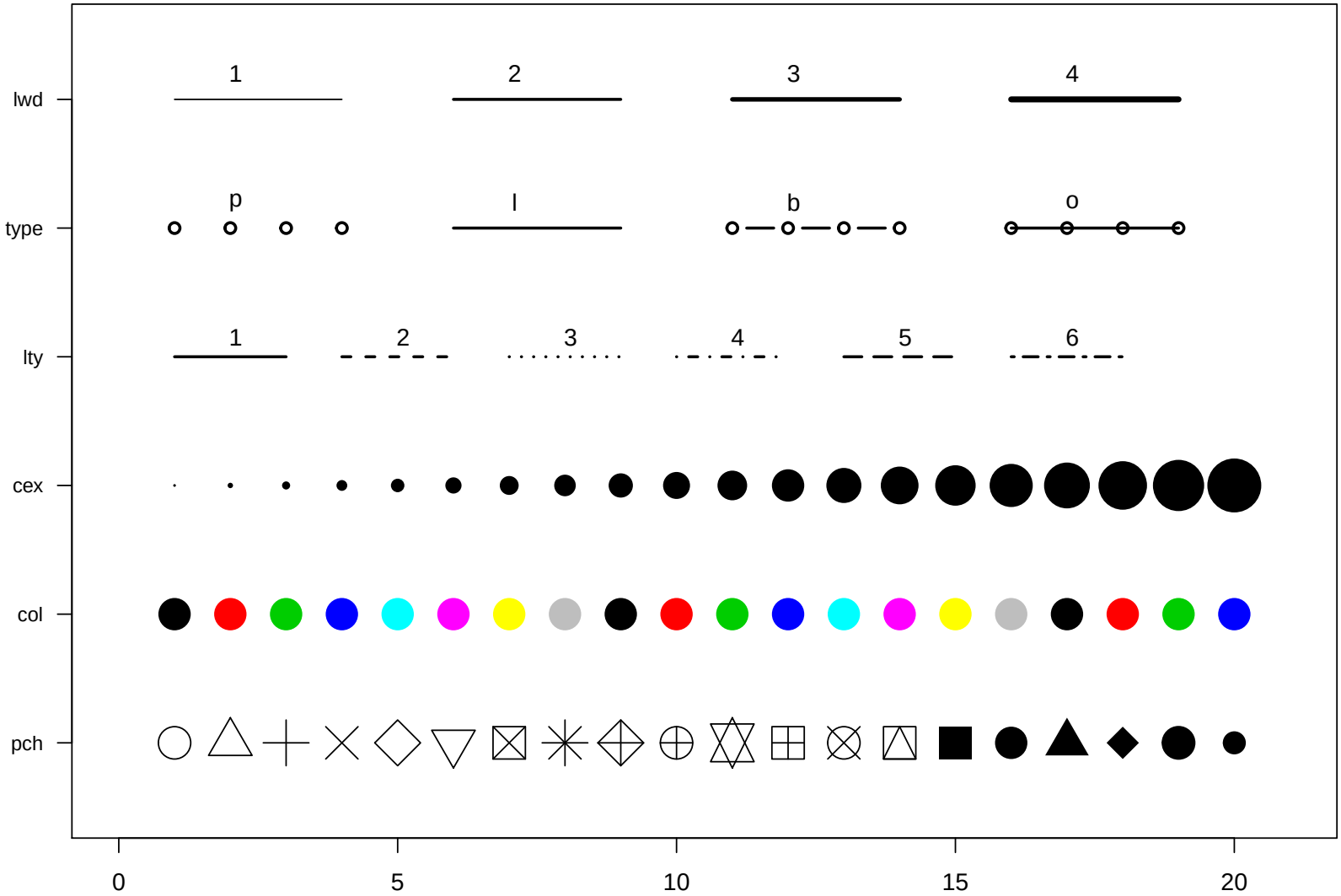
pch - example

```
# create a grid of coordinates
coords <- expand.grid(1:6,1:6)
# make a vector of numerical pch symbols
pch.num <- c(0:25)
# and a vector of character pch symbols
pch.symb <- c('*', '.', 'o', 'O', '0', '-', '+', '|', '%',
# plot numerical pch
plot(coords[1:26,1], coords[1:26,2], pch=pch.num,
bty='n', xaxt='n', yaxt='n', bg='red',
xlab='', ylab='')
# and character pch's
points(coords[27:36,1], coords[27:36,2], pch=pch.symb)
# label them
text(coords[,1], coords[,2], c(1:26, pch.symb), pos = 1,
col='slateblue', cex=.8)
```

pch - example



Visualising graph parameters

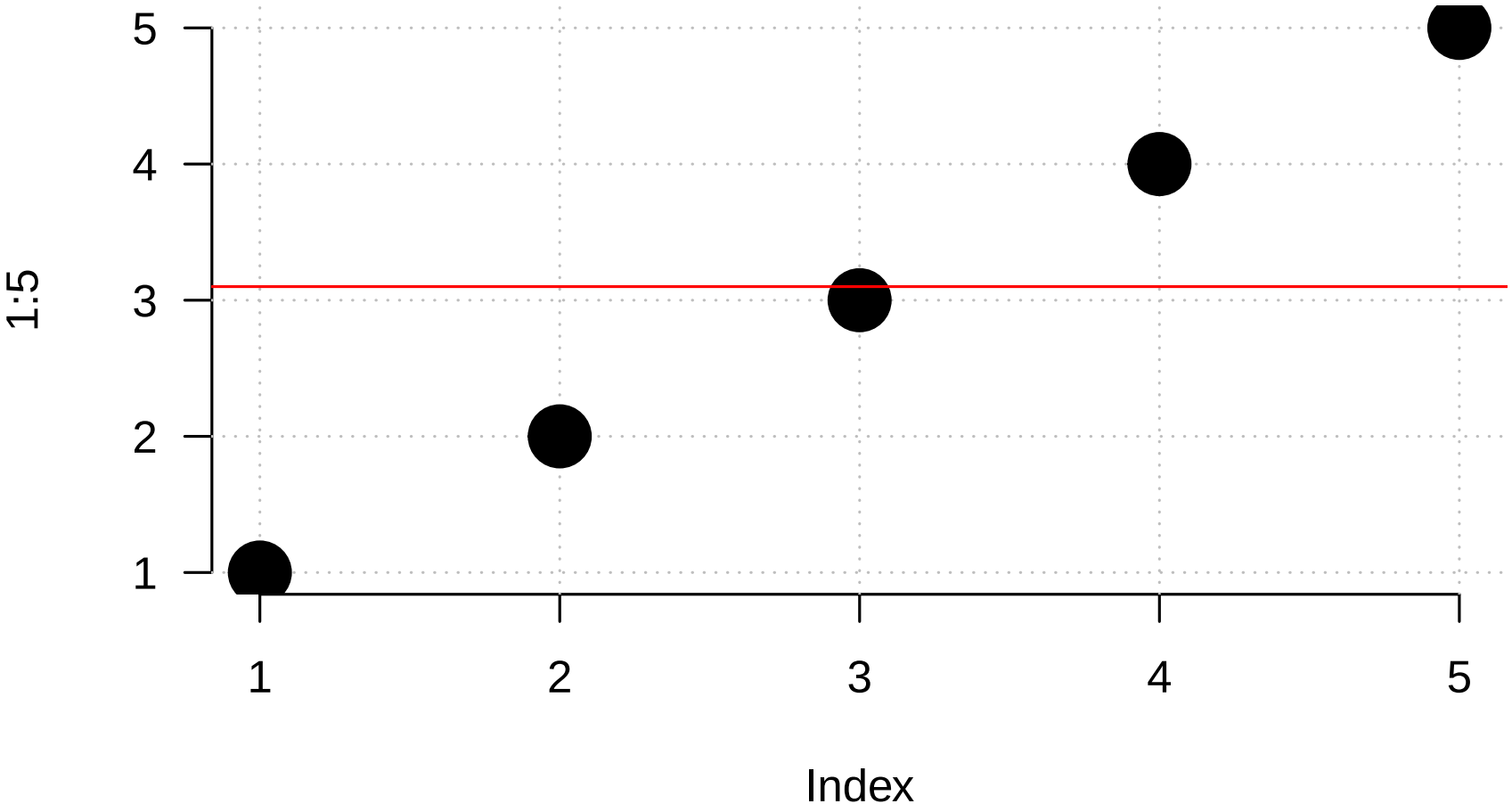


Graphic layers

Elements are added to a plot in the same order you plot them. It is like layers in a graphical program. Think about it! For instance the auxiliary grid lines should be plotted before the actual data points.

```
# make an empty plot
plot(1:5, type='n', las=1,
bty='n')
# plot grid
grid(col='grey', lty=3)
# plot the actual data
points(1:5, pch=19, cex=3)
# plot a line
abline(h = 3.1, col='red')
# you see, it overlaps the
# data point. It is better
# to plot it before points()
```

Graphic layers



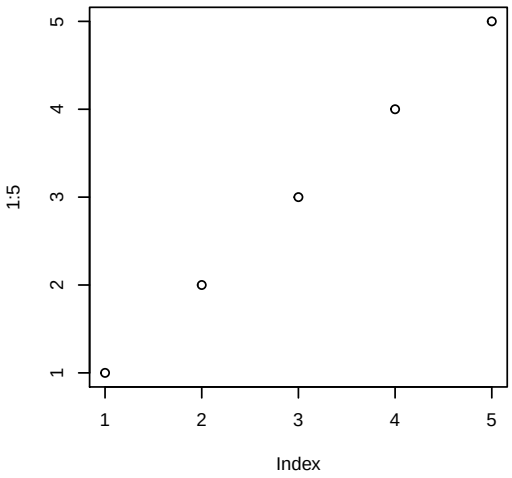
General considerations

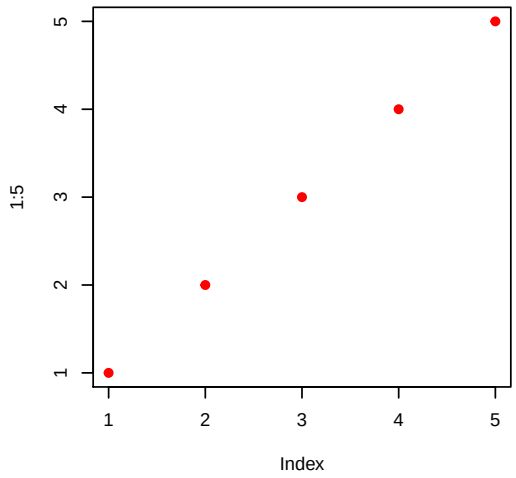
- raster or vector graphics,
- colors, e.g. color-blind people, warm vs. cool colors and optical illusions
- avoid complications, e.g. 3D plots, pie charts etc
- use black ang greys for things you do not need to emphasize, i.e. basically everything except your main result,
- avoid 3 axis

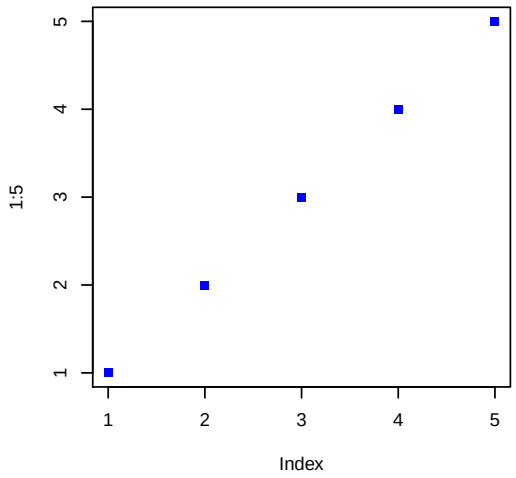
Multiple plots one figure

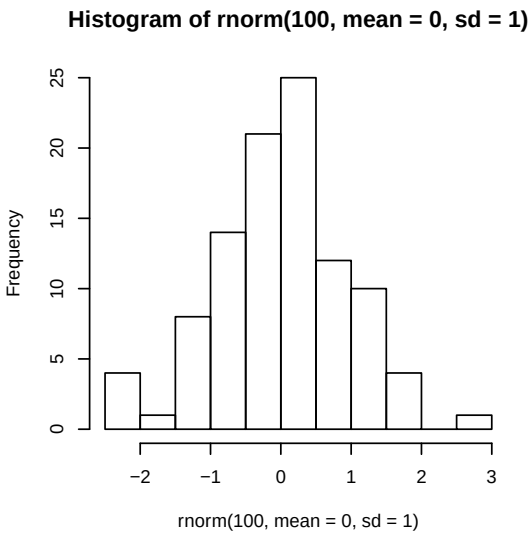
```
par(mfrow=c(2,3))
plot(1:5)
plot(1:5, pch=19, col='red')
plot(1:5, pch=15, col='blue')
hist(rnorm(100, mean = 0, sd=1))
hist(rnorm(100, mean = 7, sd=3))
hist(rnorm(100, mean = 1, sd=0.5))
par(mfrow=c(1,1))
```

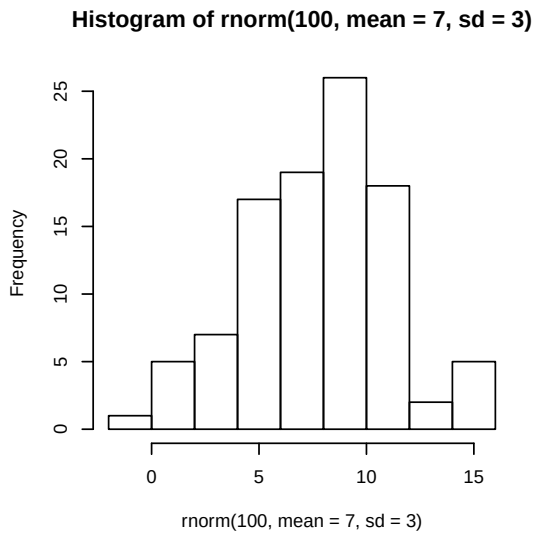
Multiple plots one figure

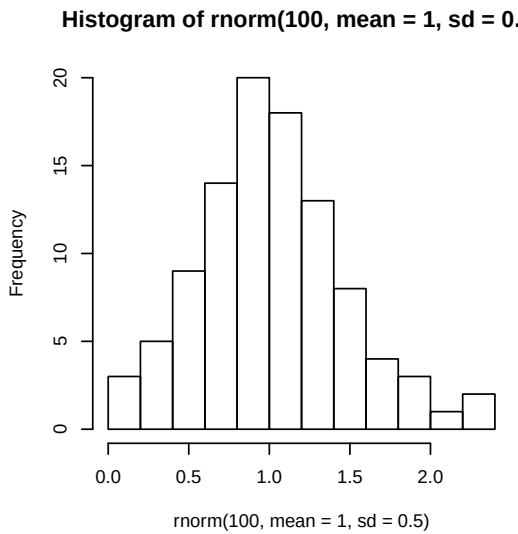










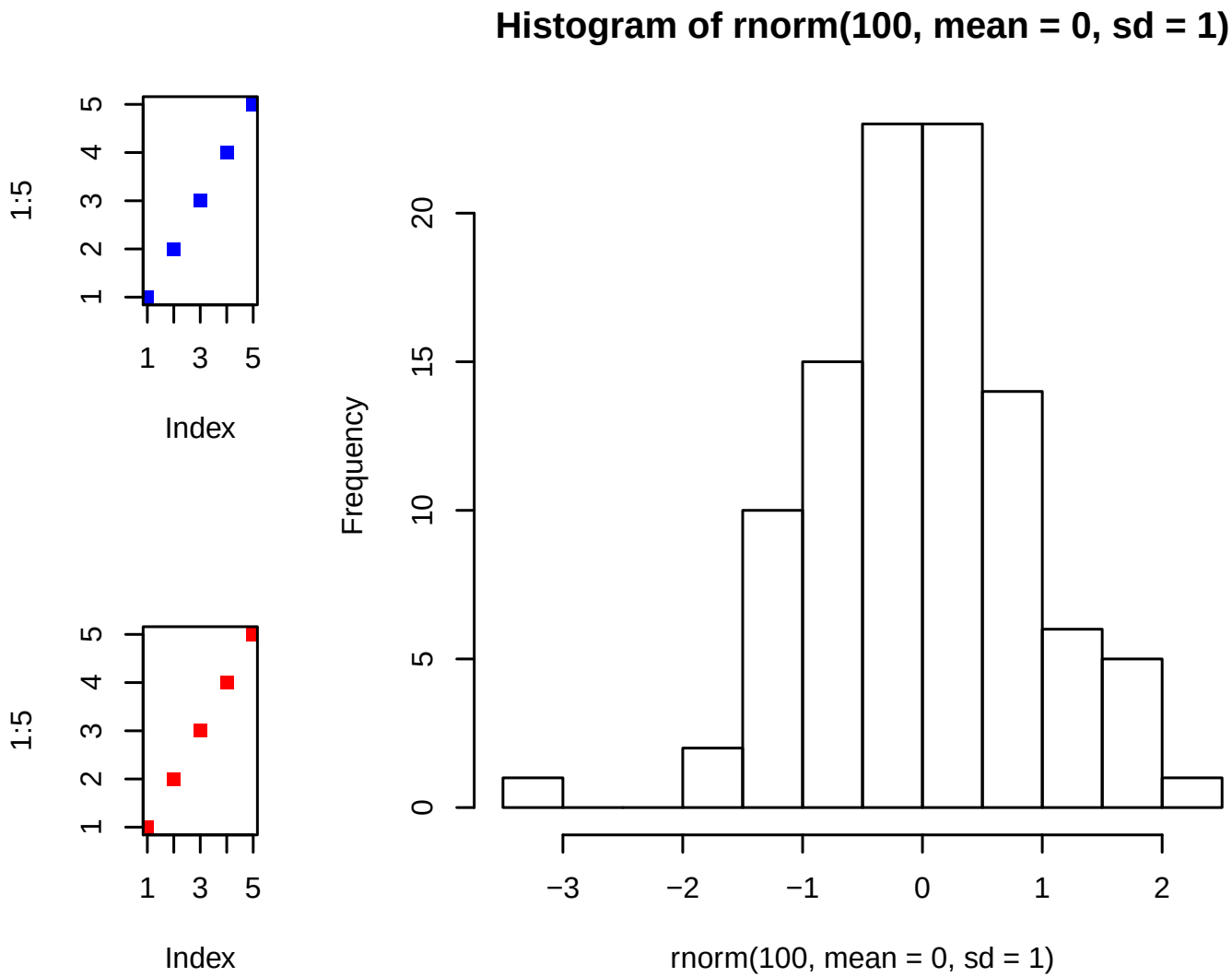


Multiple plots one figure

If more complex compositions the layout function can be used

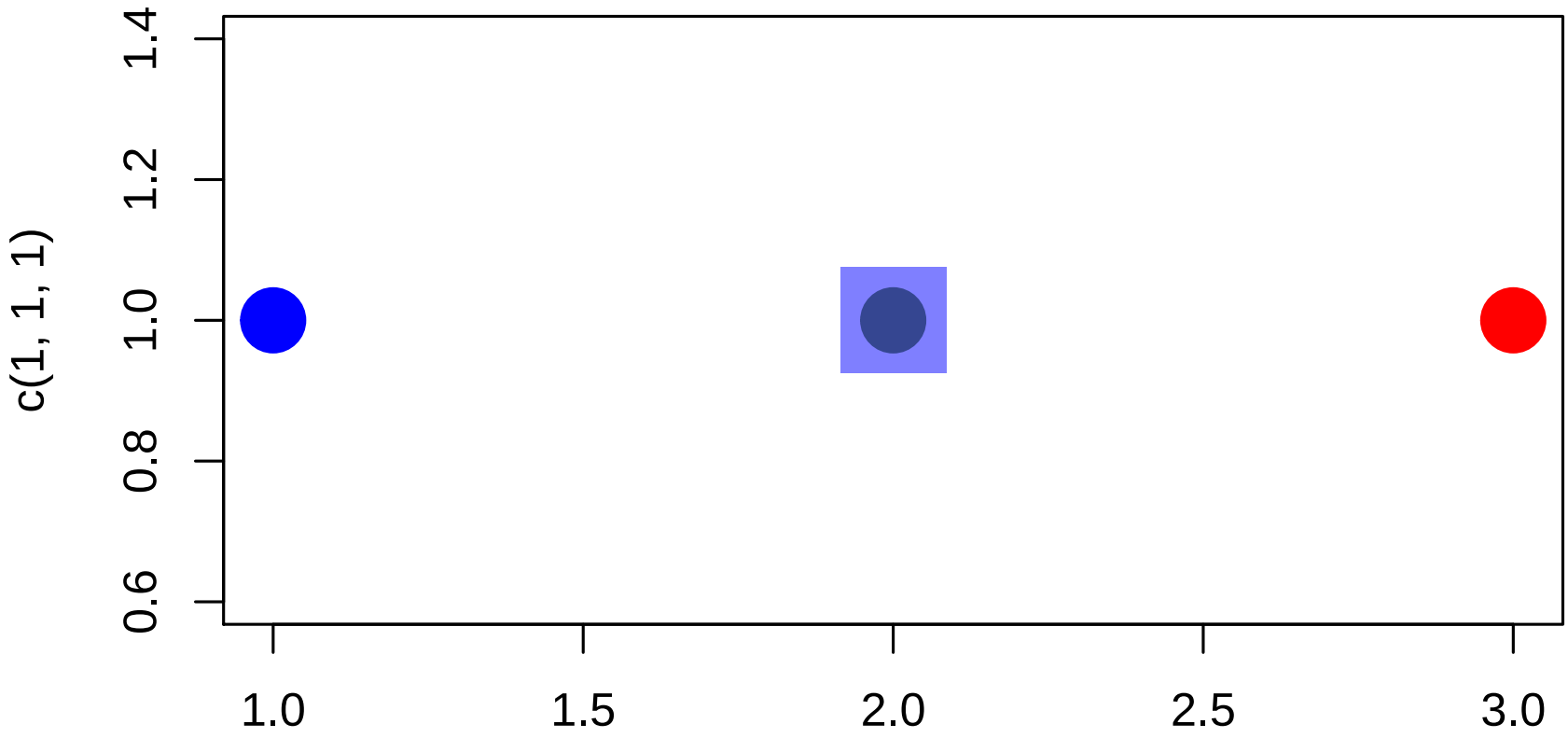
```
layout(matrix(c(1, 2, 2, 2, 3, 2, 2, 2), nrow = 2,
              ncol = 4, byrow = T))
plot(1:5, pch = 15, col = "blue")
hist(rnorm(100, mean = 0, sd = 1))
plot(1:5, pch = 15, col = "red")
```

Multiple plots one figure



Colors

```
mycol <- c(rgb(0, 0, 1), "olivedrab", "#FF0000")
plot(1:3, c(1, 1, 1), col = mycol, pch = 19, cex = 3)
points(2, 1, col = rgb(0, 0, 1, 0.5), pch = 15, cex = 5)
```



Colors

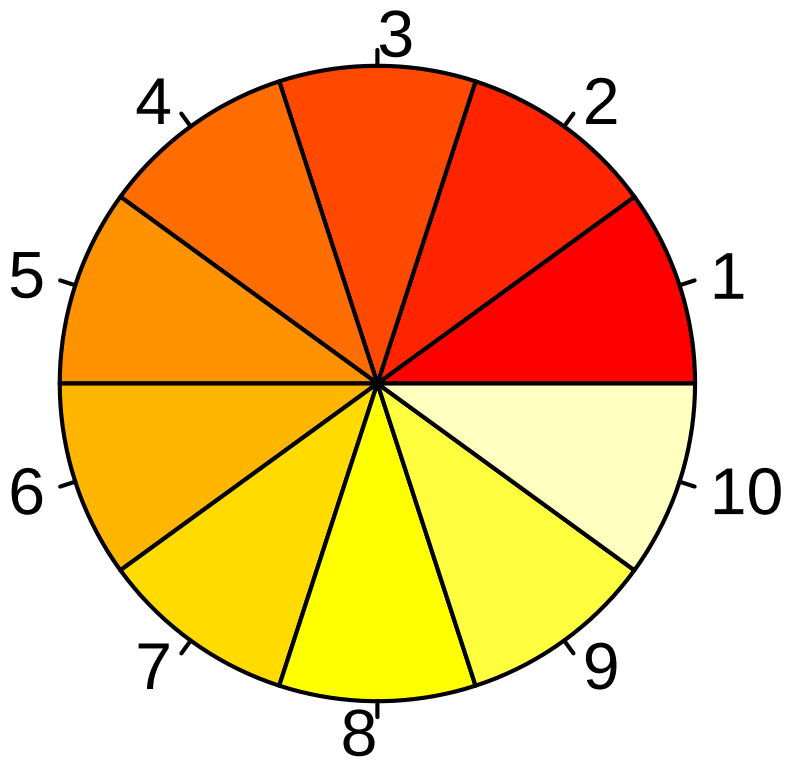
There are built-in color palettes

```
mypal <- heat.colors(10)
mypal
```

```
##  [1] "#FF0000FF" "#FF2400FF" "#FF4900FF" "#FF6D00FF" "#F
##  [6] "#FFB600FF" "#FFDB00FF" "#FFFF00FF" "#FFFF40FF" "#F
```

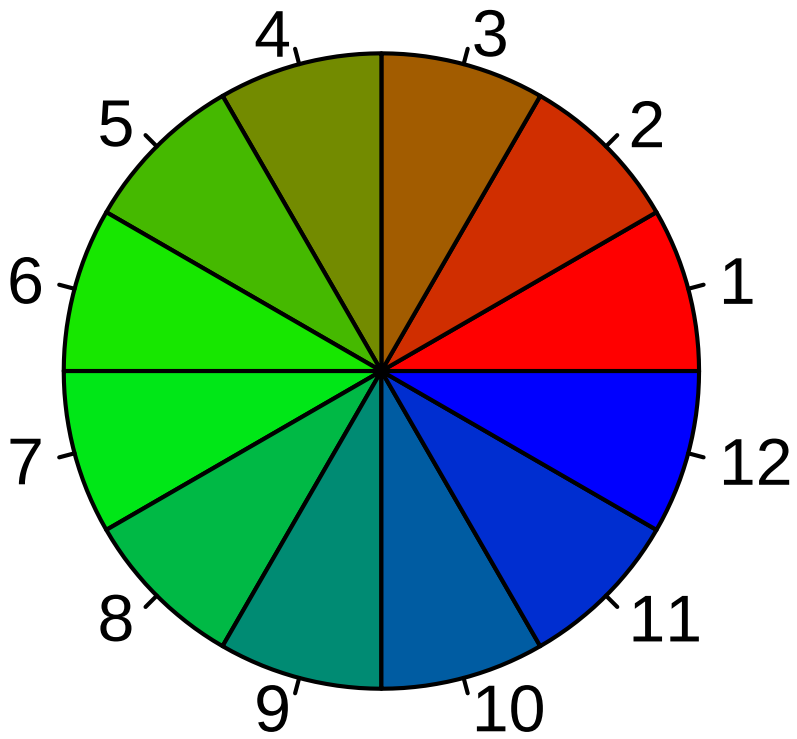
Colors

```
pie(x = rep(1, times = 10), col = mypal)
```



Colors

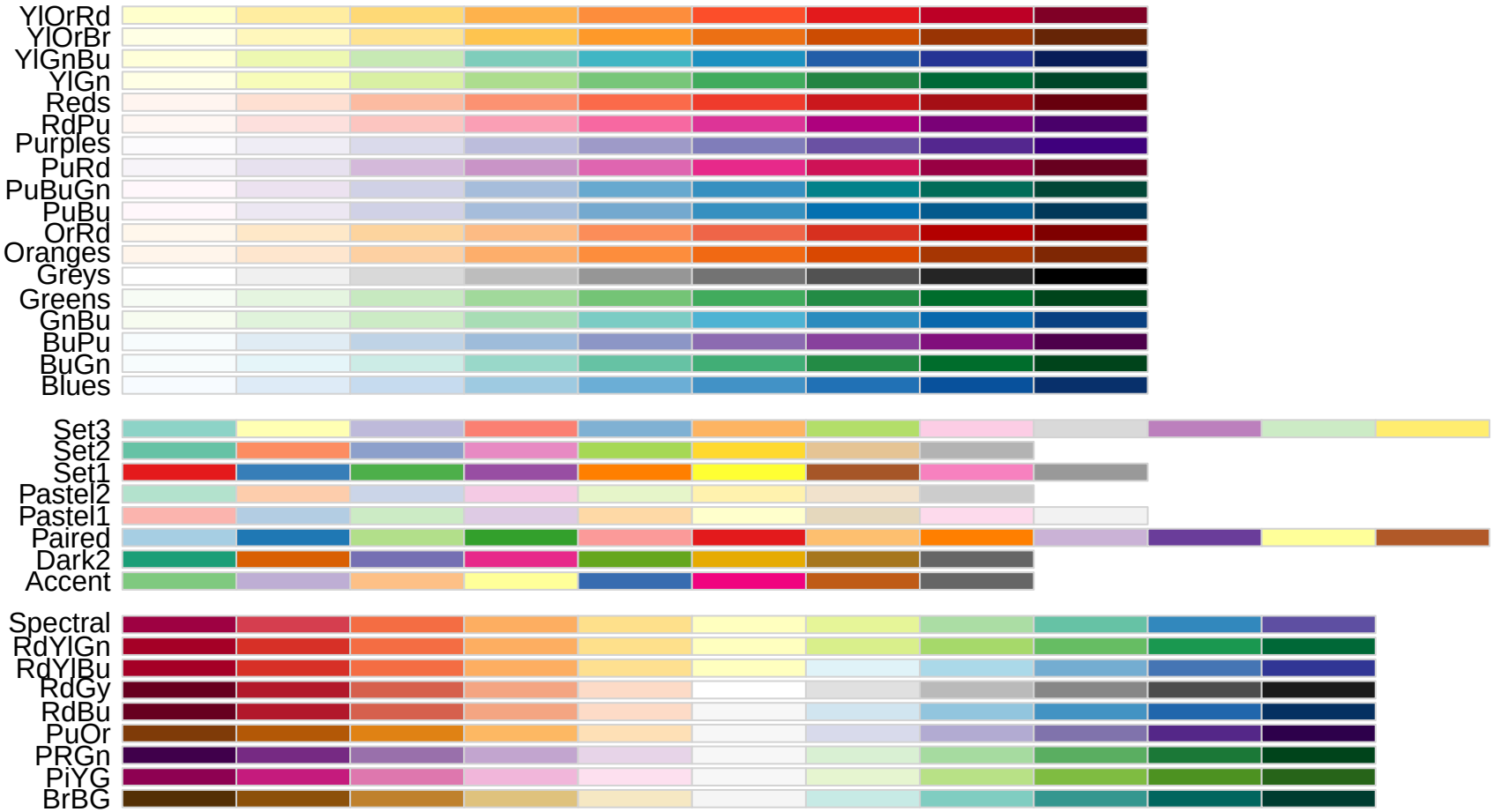
```
mypal <- colorRampPalette(c("red", "green", "blue"))
pie(x = rep(1, times = 12), col = mypal(12))
```



Note that cololRampPalette returns a function

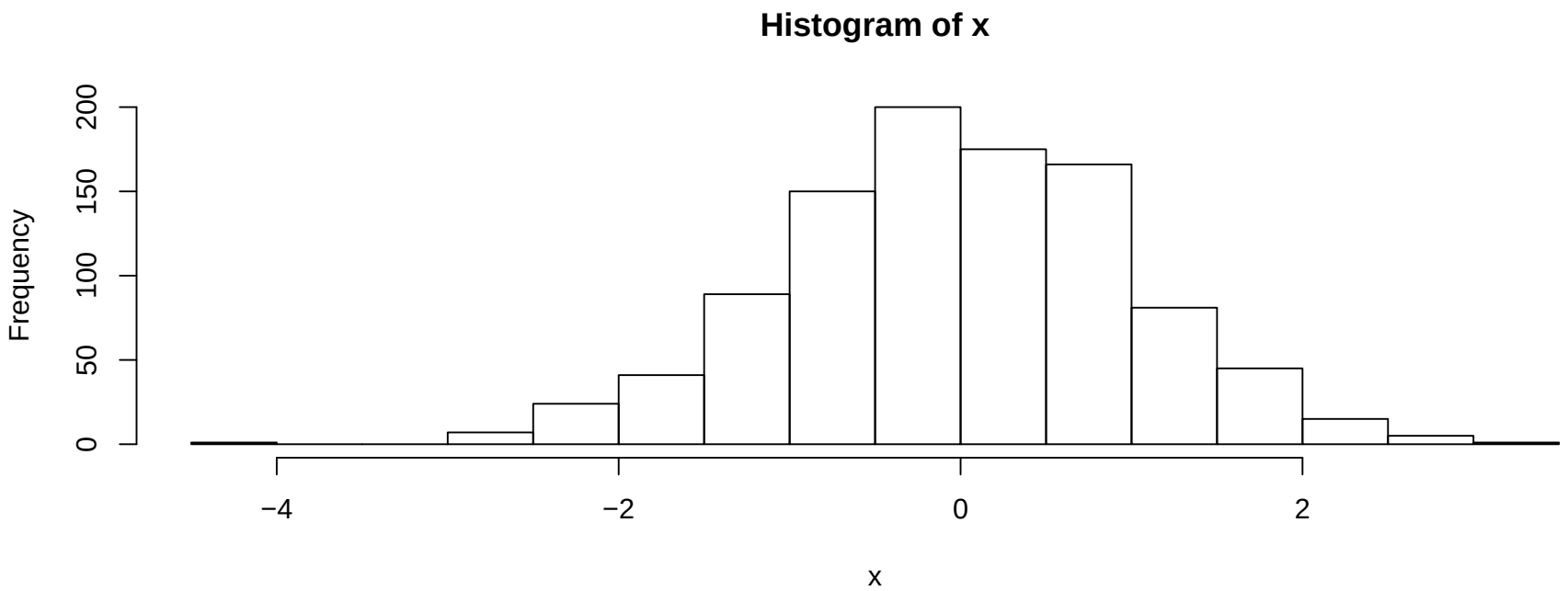
Colors

```
library(RColorBrewer)
display.brewer.all()
```



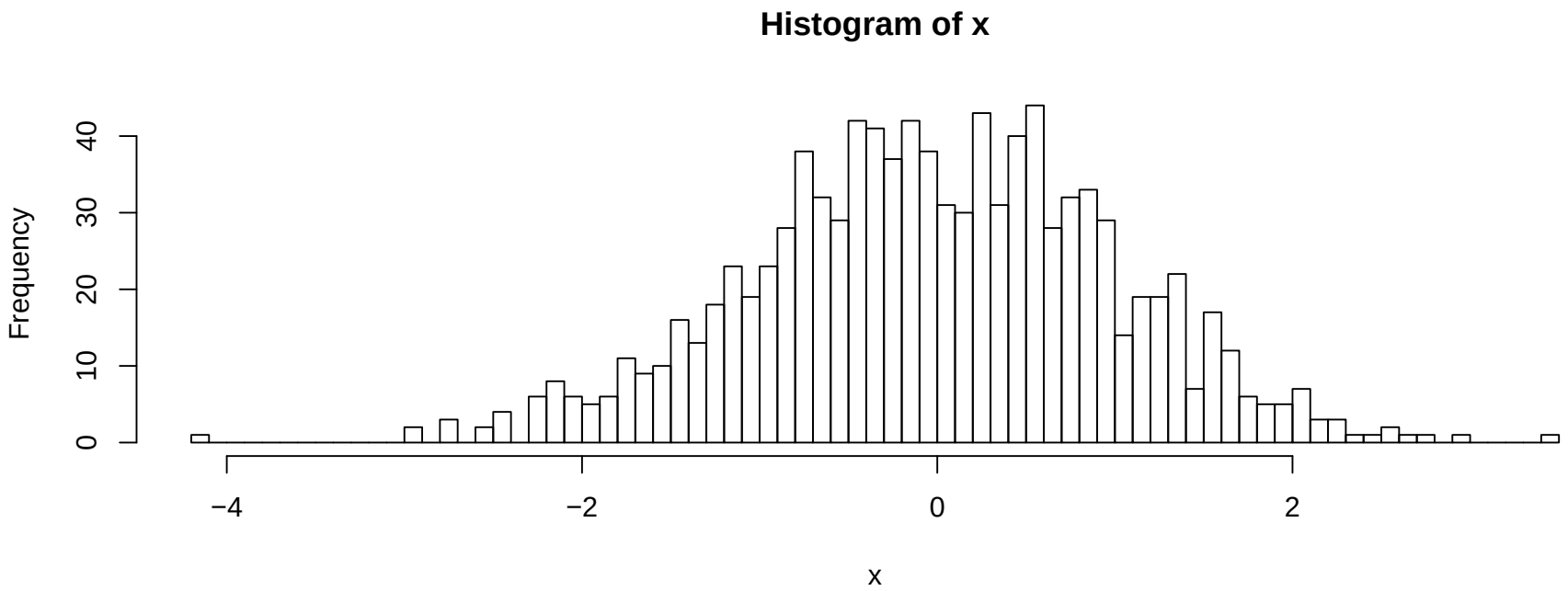
Histogram

```
x <- rnorm(1000) # generate sample from normal dist.
hist(x)
```



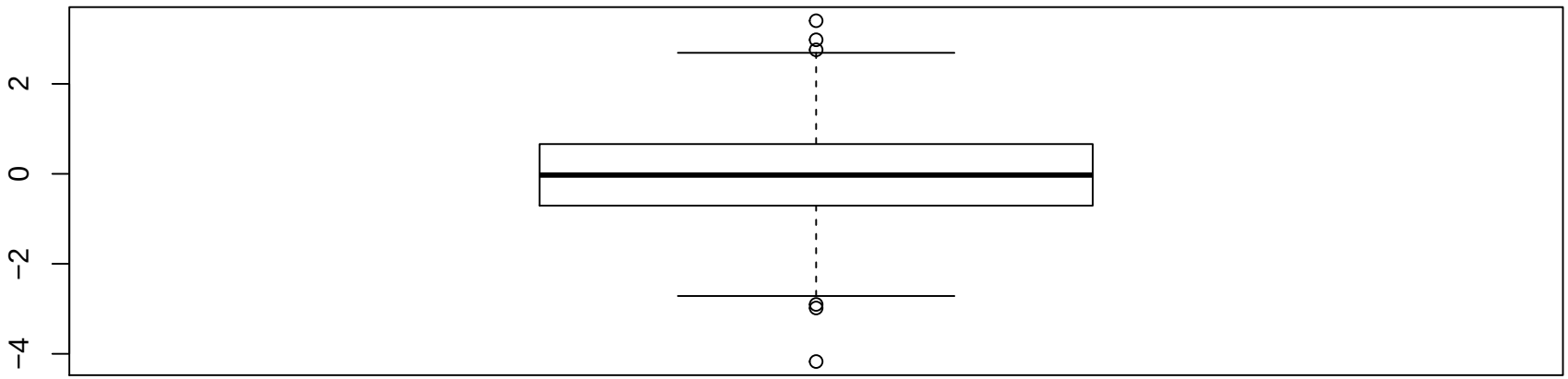
Histogram

```
hist(x, breaks = 100)
```

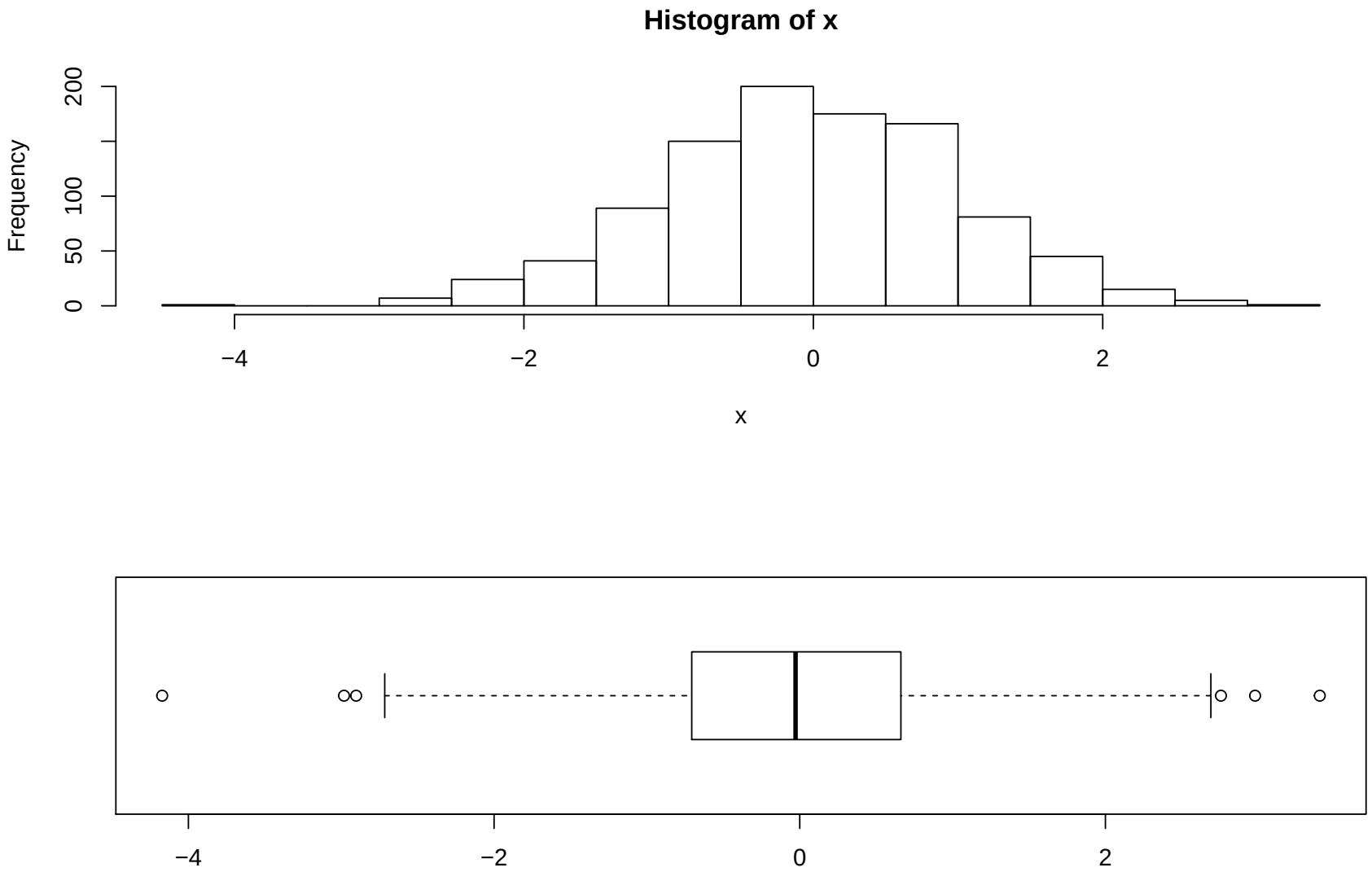


Boxplot

```
boxplot(x)
```



Hist. vs Box



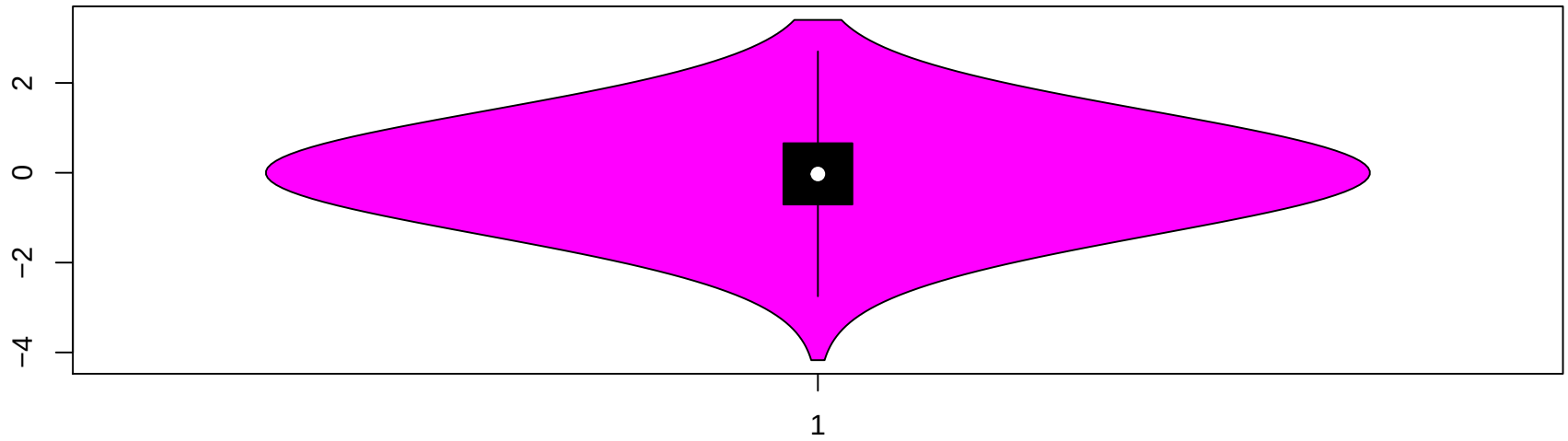
Violinplot

```
library(vioplot)
```

```
## Loading required package: sm
```

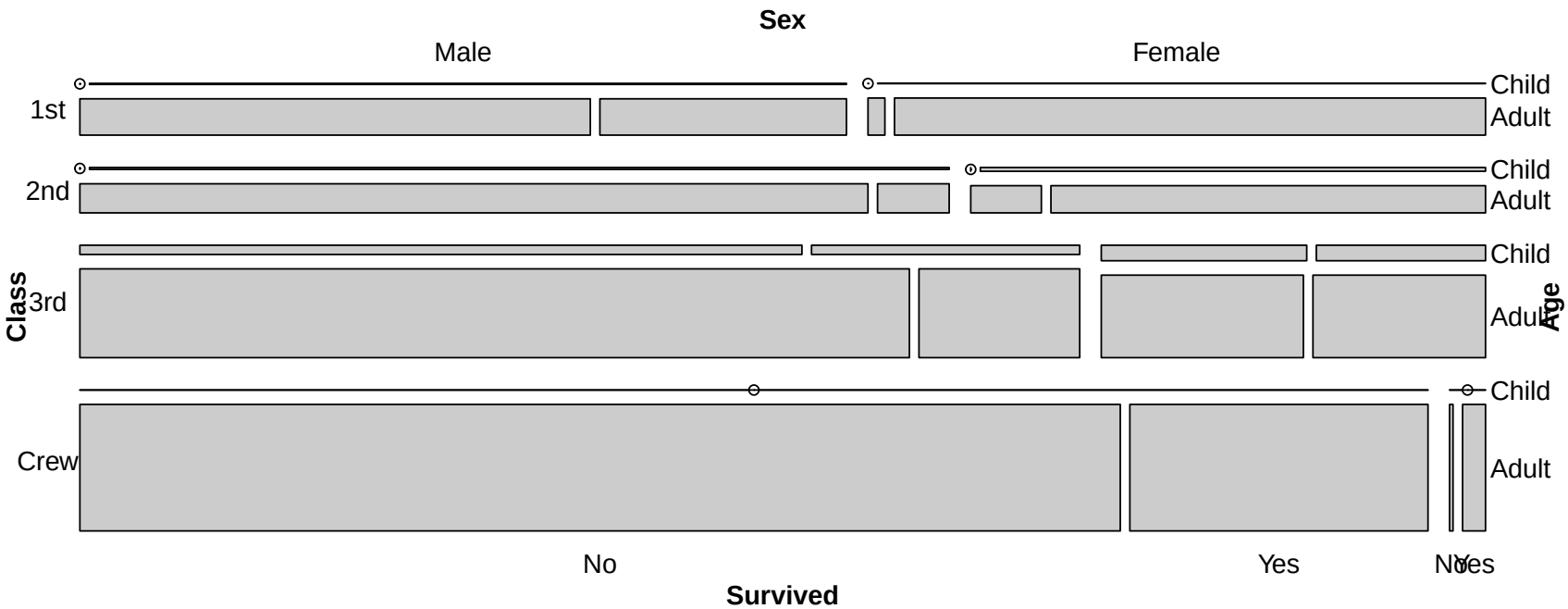
```
## Package 'sm', version 2.2-5.4: type help(sm) for summary
```

```
vioplot(x)
```



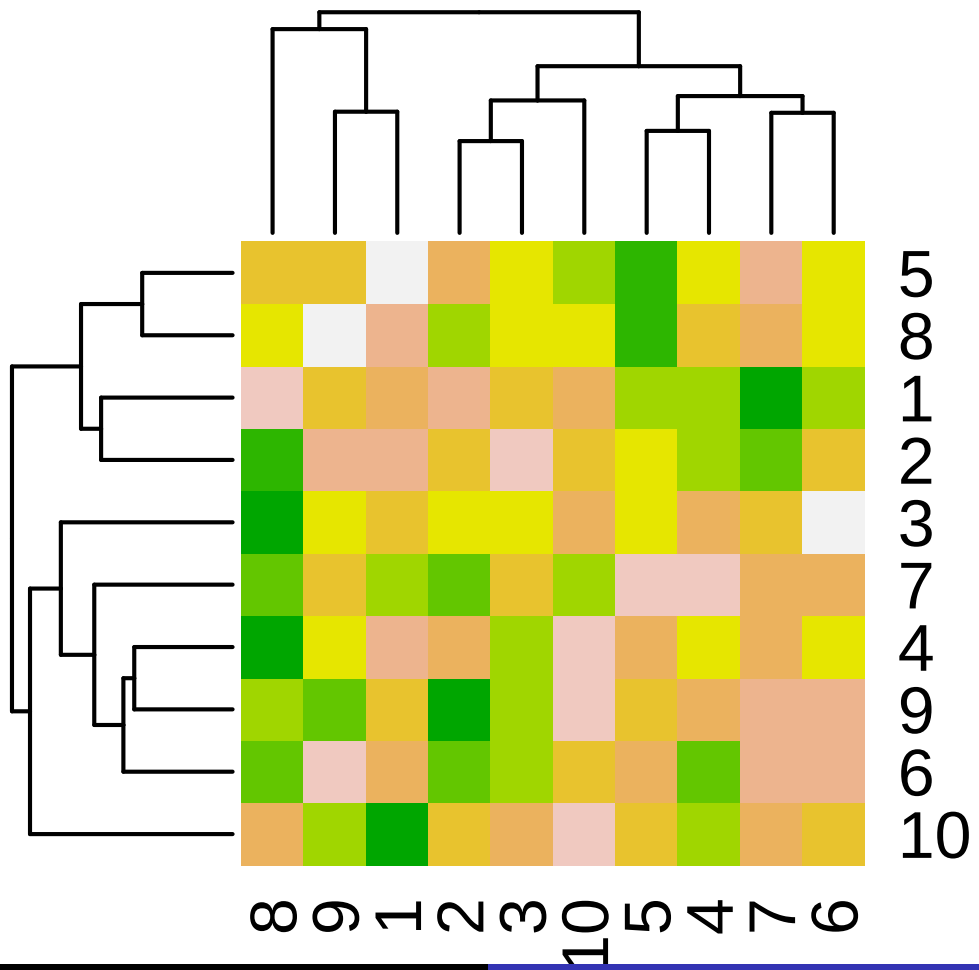
Plotting categorical data

```
library(vcd)
data(Titanic) # Load the data
mosaic(Titanic,
       labeling=labeling_border(rot_labels = c(0,0,0,0)))
```



Heatmaps

```
heatmap(matrix(rnorm(100, mean = 0, sd = 1), nrow = 10),
col=terrain.colors(10))
```



More inspiration

`http://www.r-graph-gallery.com`