Instruction material

Susanne Breitner-Busch (Course A) (susanne.breitner@helmholtz-munich.de)

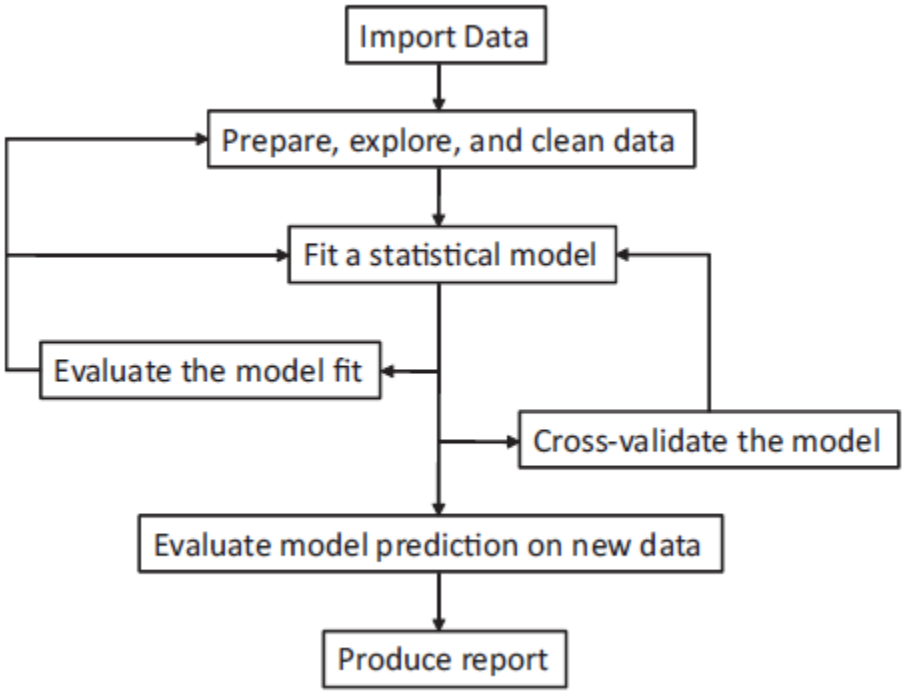
Tobias Niedermaier (Course B) (tobias.niedermaier@ibe.med.uni-muenchen.de)
Department of Medical Informatics, Biometry and Epidemiology (IBE) University of Munich (LMU)

Session 1

Why R?

- R is *open source*
- All techniques for data analyses
- State-of-the-art graphics capabilities
- A platform for programming new statistical methods or analysis pipelines (in form of R-packages)

Common steps in data analysis:



"Good programmers are made, not born." **(Gerald M. Weinberg - The Psychology of Computer Programming)**

consequence I

TRAIN,. . .

consequence II

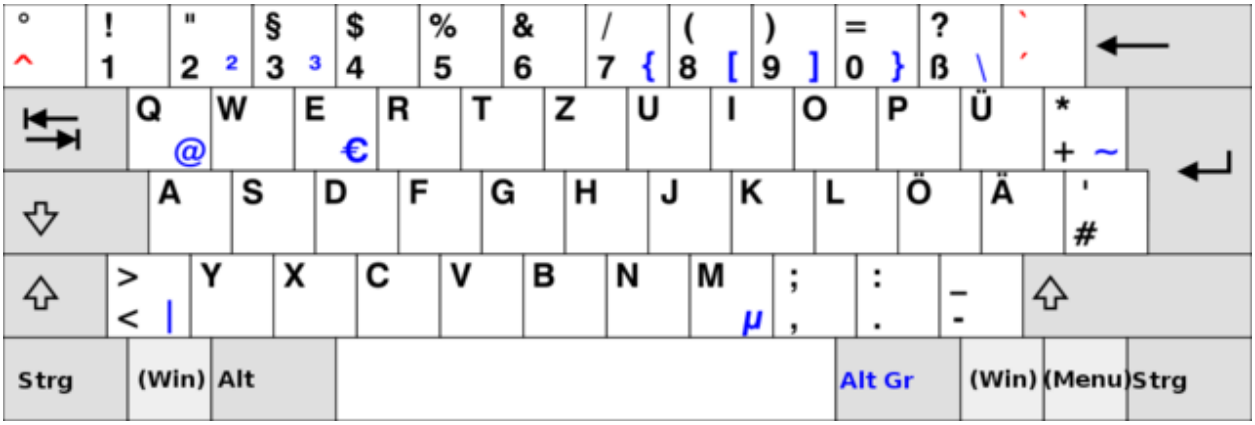
TRAIN,. . .

consequence III

AND THEN TRAIN SOME MORE!

Before we start

German/QWERTZ keyboard:



Installing R and RStudio

- R is a programming language for the purpose of doing statistical analyses;
- R is for free!
- R can be downloaded from http://www.r-project.org for different platforms;
- Follow this link with a full description of how to install R and RStudio.

  Hands-on: What is the name of the latest version of R?

Starting R

- Under windows: Rgui.exe (graphical user interface);
- under UNIX: Enter the command R in the UNIX shell;

- when R starts it provides you with a prompt >;
- the prompt is the entry point for communicating with R;
- you can type expressions at the prompt, R evaluates these expressions and returns some output

```
1+1
```

```
## [1] 2
```

- R is object oriented, i.e. we can create objects (variables) that persist during an R session

```
x <-1+1
x
```

```
## [1] 2
```

- you can finish R by using the command q().

Writing scripts

- It is often useful to write a script (program) that collects your R commands for the purpose of reproducing your data analyses;
- you can save your R commands e.g. in a text editor;
- the sympbol ## can be added before text to indicate comments

```
# y <- 1 + 1
y
```

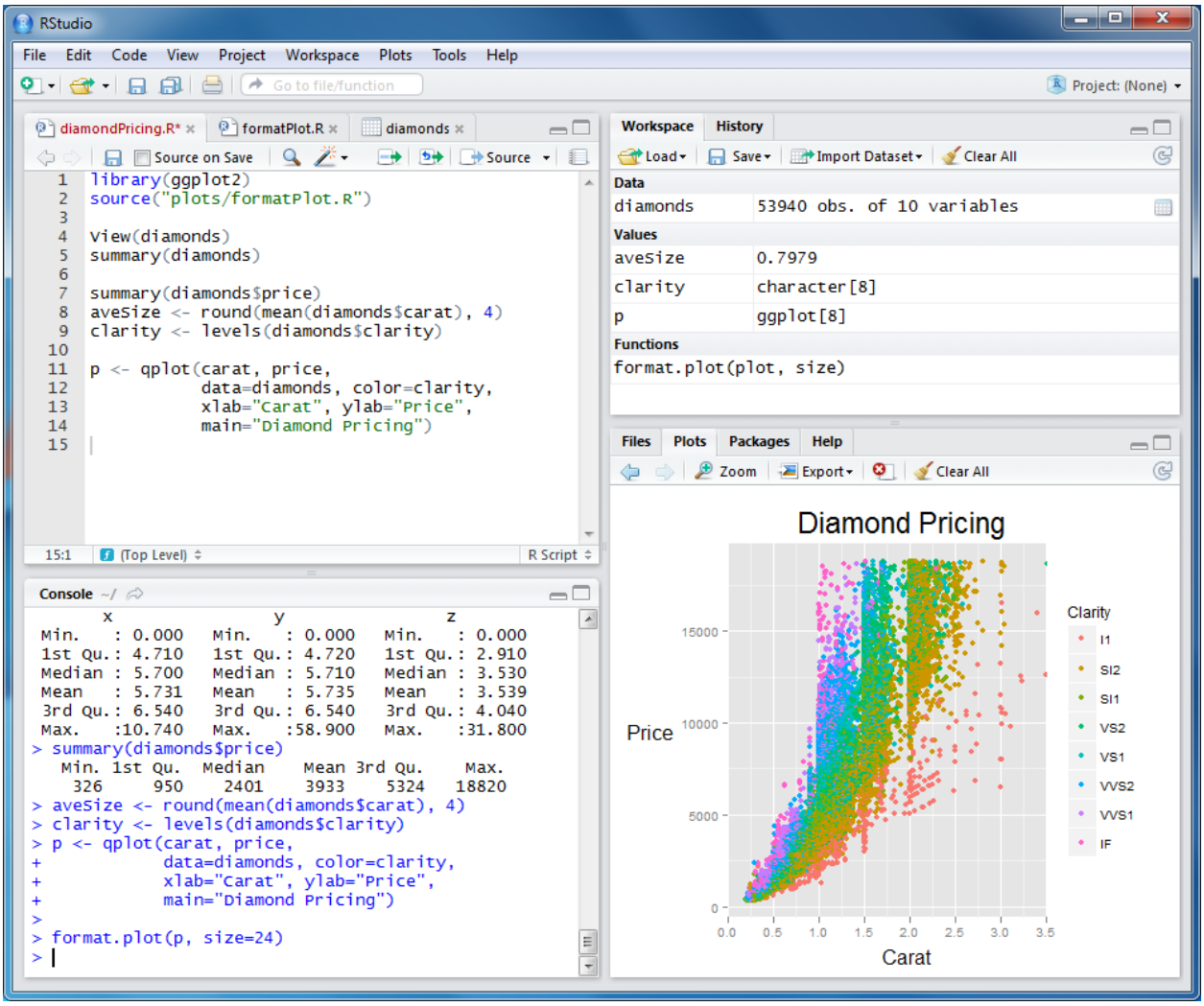
```
## Error: Objekt 'y' nicht gefunden
```

```
y <-1 + 1
y
```

```
## [1] 2
```

RStudio

- **R-Studio from http://rstudio.org/**



- Some people dislike bloated IDEs and prefer a text editor (e.g. (Neo)Vim, VScose, . . .)

  Hand-on: Create your own script in RStudio, name it and save it in your course directory.

Packages

- R has an archive called Comprehensive R Archive Network (CRAN);
- in this archive there are statistical functions for added functionality comprised in packages;
- you can install packages by using the function install.packages()

```
install.packages("ggplot2")
install.packages("knitr")
```

- installed packages and packages already installed in your “original”, downloaded R environment can be loaded by

```
library(ggplot2)
library(knitr)
```

Arithmetic I

- Operators:
- addition: +
- subtraction: -
- multiplication: * I division: /
- exponentiation: ^
- use brackets () to specify the order of operations;
- examples:

```
(1 + 1/100)^100
```

```
## [1] 2.704814
```

```
1 + 1/100^100
```

```
## [1] 1
```

Arithmetic II

- R provides a number of mathematical built-in functions and constants, e.g.:
- Sine (in radians): sin()
- Cosine (in radians): cos()
- Tangent (in radians): tan()
- Exponent: exp()
- Logarithm: log()
- Square root: sqrt()
- Factorial: factorial()
- Circle constant: pi ...
- Examples:

```
exp(1)
```

```
## [1] 2.718282
```

```
pi
```

```
## [1] 3.141593
```

```
sin(pi / 6)
```

```
## [1] 0.5
```

Variables I

- To save a number, you can store it in a variable:

```
x <- 34
```

- the assignment operator <- tells R to take the number on the right and store it into the variable x;
- variable names have to start with a letter and can be made up of letters, numbers, dot (.) and underscore ( );
- R is case sensitive, i.e. the variable x is not the same as X;
- you can display variables simply by typing the variable name on the screen.

Variables II

- Examples:

```
x <- 100
x
```

```
## [1] 100
```

```
(1 + 1 / x) ^x
```

```
## [1] 2.704814
```

```
x <- 200
(1 + 1 / x)^ x
```

```
## [1] 2.711517
```

```
y <- x ^2 + 4
y
```

```
## [1] 40004
```

Vectors I

- you are not limited to saving only a single number in a variable;
- you can create a vector, which is an ordered collection of variables;
- a basic function to create a vector is the function c() (concatenate)

```
c(2, 1, 5, 3, 8)
```

```
## [1] 2 1 5 3 8
```

- for generating regular sequences, use functions seq(from, to, by) and rep(x, times).

Vectors II

```
x <- seq(1, 14, 2)
y <- rep(3, 5)
z <- c(x,y)
x <- seq(10, 1, -1)
gender <- c("male", "female", "female", "male", "male")
names(gender) <- c("Patient1","Patient2","Patient3","Patient4","Patient5")
```

Vectors III

- The expressions seq(from, to, by = 1) and seq(from, to, by = -1) are often used so that R provides the shorthand from:to

```
1:12
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
20:10
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10
```

- to refer to the i-th element of a vector x we use x[i]:
- i is positive: x[i] is the corresponding subvector of x;
- i is negative: the corresponding values of x are omitted.

```
x <- 100:107
x[4]
```

```
## [1] 103
```

```r
x[c(1, 2, 5)]
```

```
## [1] 100 101 104
```

```r
x[-2]
```

```
## [1] 100 102 103 104 105 106 107
```

```r
x[c(-2, -3, -4)]
```

```
## [1] 100 104 105 106 107
```

Vectors IV

- The functions length(x) gives the number of elements of the vector x;
- algebraic operations on vectors act on each element separately, i.e. elementwise

```r
x <- c(2, 1, 4, 5)
y <- c(4, 5, 6, 2)
x*y
```

```
## [1]  8  5 24 10
```

```r
x + y
```

```
## [1]  6  6 10  7
```

```r
exp(x)
```

```
## [1]   7.389056   2.718282  54.598150 148.413159
```

Vectors V

- A useful set of functions taking vector arguments are sum(...), prod(...), max(...), min(...), sort(...),. . .
- Example:

```r
x <- c(1.7, 0.2, 2.3, 3.4)
x.mean <- sum(x) / length(x)
```

Functions I

- A function in R takes on one or more arguments and produces one or more outputs (return values);
- to call a built-in (or user-defined) function in R you write the name of the function followed by its argument values enclosed in brackets and separated by commas;
- for example, the seq function produces arithmetic sequences

```r
seq(from = 1, to = 12, by = 2)
```

```
## [1]  1  3  5  7  9 11
```

- some arguments are optional, and have predefined default values. For example, if we omit the argument by, then R uses by = 1

```r
seq(from = 1, to = 12)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

- every function has a default order for the arguments (see built-in help!).

```r
?seq
```

Functions II

- If you provide arguments in this order, then they do not need to be named

```r
seq(1, 12, 2)
```

```
## [1]  1  3  5  7  9 11
```

- however, you can choose to give the arguments out of order provided you give them names in the format `argument_name = expression`

```r
seq(to=12,by=2,from=1)
```

```
## [1]  1  3  5  7  9 11
```

- each argument value is given by an expression which can be a constant, variable or another function call

```r
x <- 9
seq(1,x,x/3)
```

```
## [1] 1 4 7
```

Logical expressions I

- A logical expression is formed using the comparison operators <, >, <=, >=, ==, ! = and the logical operators & (and), | (or) and ! (not);
- the value of a logical expression is either TRUE or FALSE

```
x <- 24
x > 100
```

```
## [1] FALSE
```

```
x < 100
```

```
## [1] TRUE
```

```
x != 100
```

```
## [1] TRUE
```

```
x == 100
```

```
## [1] FALSE
```

- logical expressions are useful for selecting a subvector using the indexing operation x[subset].
- WARNING: Dont use it for finding a long decimal number

```
sqrt(7)
```

```
## [1] 2.645751
```

```
2.645751 == sqrt(7)
```

```
## [1] FALSE
```

- Why?

Logical expressions II

- Examples:

```
x <- c(1:9, 4, 5)
x > 3
```

```
##  [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
x[x > 3]
```

```
## [1] 4 5 6 7 8 9 4 5
```

- if you wish to know the index positions of TRUE elements of a logical vector x, then use which(x)

```r
which(x == 5)
```

```
## [1]  5 11
```

```r
which(x < 5)
```

```
## [1]  1  2  3  4 10
```

Data Frames

- Data frames are the typical data structure in R. Sloppily said (for now :)), data frames are rectangular tables with columns representing different variables. An example data frame painters is available in the library MASS:

```r
library(MASS)
painters
```

```
##                 Composition Drawing Colour Expression School
## Da Udine                 10       8     16          3      A
## Da Vinci                 15      16      4         14      A
## Del Piombo                8      13     16          7      A
## Del Sarto                12      16      9          8      A
## Fr. Penni                 0      15      8          0      A
## Guilio Romano            15      16      4         14      A
## Michelangelo              8      17      4          8      A
## Perino del Vaga          15      16      7          6      A
## Perugino                  4      12     10          4      A
## Raphael                  17      18     12         18      A
## F. Zucarro               10      13      8          8      B
## Fr. Salviata             13      15      8          8      B
## Parmigiano               10      15      6          6      B
## Primaticcio              15      14      7         10      B
## T. Zucarro               13      14     10          9      B
## Volterra                 12      15      5          8      B
## Barocci                  14      15      6         10      C
## Cortona                  16      14     12          6      C
## Josepin                  10      10      6          2      C
## L. Jordaens              13      12      9          6      C
## Testa                    11      15      0          6      C
## Vanius                   15      15     12         13      C
## Bassano                   6       8     17          0      D
## Bellini                   4       6     14          0      D
## Giorgione                 8       9     18          4      D
## Murillo                   6       8     15          4      D
## Palma Giovane            12       9     14          6      D
## Palma Vecchio             5       6     16          0      D
## Pordenone                 8      14     17          5      D
## Tintoretto               15      14     16          4      D
## Titian                   12      15     18          6      D
## Veronese                 15      10     16          3      D
## Albani                   14      14     10          6      E
```

```
## Caravaggio                6       6     16          0      E
## Corregio                 13      13     15         12      E
## Domenichino              15      17      9         17      E
## Guercino                 18      10     10          4      E
## Lanfranco                14      13     10          5      E
## The Carraci              15      17     13         13      E
## Durer                     8      10     10          8      F
## Holbein                   9      10     16         13      F
## Pourbus                   4      15      6          6      F
## Van Leyden                8       6      6          4      F
## Diepenbeck               11      10     14          6      G
## J. Jordaens              10       8     16          6      G
## Otho Venius              13      14     10         10      G
## Rembrandt                15       6     17         12      G
## Rubens                   18      13     17         17      G
## Teniers                  15      12     13          6      G
## Van Dyck                 15      10     17         13      G
## Bourdon                  10       8      8          4      H
## Le Brun                  16      16      8         16      H
## Le Suer                  15      15      4         15      H
## Poussin                  15      17      6         15      H
```

The output is very long. Use head() to show only the first elements.

```
head(painters)
```

```
##               Composition Drawing Colour Expression School
## Da Udine               10       8     16          3      A
## Da Vinci               15      16      4         14      A
## Del Piombo              8      13     16          7      A
## Del Sarto              12      16      9          8      A
## Fr. Penni               0      15      8          0      A
## Guilio Romano          15      16      4         14      A
```

- Here, the names of the painters serve as row identifications, i.e., every row is assigned to the name of the corresponding painter. However, these names are not variables of the data set! You can assess the names with the function rownames(painters);
- the data set contains five variables: names(painters) (also: colnames(painters));
- use $ to extract single variables: painters$Composition.
- Subsets of a data frame can be obtained with subset() or with the second equivalent command:

```
subset(painters, School=="F")
```

```
##            Composition Drawing Colour Expression School
## Durer                8      10     10          8      F
## Holbein              9      10     16         13      F
## Pourbus              4      15      6          6      F
## Van Leyden           8       6      6          4      F
```

```
painters[painters$School=="F",]
```

```
##            Composition Drawing Colour Expression School
## Durer                8      10     10          8      F
## Holbein              9      10     16         13      F
## Pourbus              4      15      6          6      F
## Van Leyden           8       6      6          4      F
```

- uninteresting columns can be eliminated:

```
subset(painters, School=="F", select = c(-3,-5))
```

```
##            Composition Drawing Expression
## Durer                8      10          8
## Holbein              9      10         13
## Pourbus              4      15          6
## Van Leyden           8       6          4
```

Help

- We cannot cover all features of R in our R exercises and even all features mentioned in our exercises are not covered in full detail;
- to find out more about an R command or function, e.g. about the function seq(), type:

```
help(seq)
#or just
?seq
```

- if you do not know the exact name of a function, you can try

```
?help.search("sequence")
#or just
??sequence
```

- for a useful HTML help interface, type :

```
help.start()
```

- Google, ChatGPT etc. are your friends!
- There's always a person who had the same problem before you.

References

- The R Foundation for Statistical Computing, R-version 4.5.1 (Statistical Software). Available at http://www.r-project.org.
- Jones O, Maillardet R, Robinson A. (2009) Introduction to Scientific Programming and Simulation Using R. CRC Press, Boca Raton.
- Robert I. Kabacoff (2011) R in Action Data Analysis and Graphics with R.ISBN 9781935182399

Session 2

Data Import

- In an R session, you can import a data file . . .
- `read.table()` is a standard function for importing a text file
- most important arguments:
- header with values TRUE or FALSE: Should the first row be interpreted as variable names of the data set?
- sep: Which character separates different values on each line?
- row.names (TRUE or FALSE): Should the first column be interpreted as row (sample) names?
- Other functions: `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()`

Data Export

- or export a data file:
- with the function `write.table()` you can save your data frame in a text (or Excel . . .) file;
- you can also save objects you created in R (e.g. the results of your analyses) in an .Rdata file and load them every time you start your R session:

```
x <- c(1,5, 2, 3, 6)
save(x, file="x.Rdata")
load("x.Rdata")
```

Object types in R

- We already met vectors and data frames. Now we see factors:
- a factor is a vector whose elements are not interpreted as numbers, but as categories;
- e.g., you would save gender, race, tumor stage . . . in a factor

```
x <- factor(c(1, 1, 2, 1, 1, 2, 2, 2, 1, 1, 2))
```

- usually, researchers code categorical variables as numbers, e.g. male=0, female=1. This makes it important to specify the object type, because R functions handle different types of objects in different ways:

```
y <- c(1, 1, 2, 1, 1, 2, 2, 2, 1, 1, 2)
summary(x)
```

```
## 1 2
## 6 5
```

```r
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.455   2.000   2.000
```

- to see if a variable is a factor:

```r
is.factor(x)
```

```
## [1] TRUE
```

Missing data

- In R, a missing value is coded by NA;
- if there is a missing value in a vector, some functions return a value NA when calculating with it:

```r
x <- c(2, 1, 6, NA, 1)
sum(x)
```

```
## [1] NA
```

- you can omit this by setting the argument na.rm to the value TRUE (ignore NA's):

```r
sum(x, na.rm=T)
```

```
## [1] 10
```

Summary Statistics

- Location: `mean()`, `median()`, `quantile()`;
- `summary()` computes the basic statistics (min, 1^{st} quartile,mean, median, 3^{rd} quartile, max);
- spread: var, sd, range, IQR (interquartile range)

```r
x <- c(2, 1, 6, 3, 1, 5, 1, 2)
median(x)
```

```
## [1] 2
```

```r
quantile(x, 0.5)
```

```
## 50%
##   2
```

```r
var(x)
```

```
## [1] 3.696429
```

```r
IQR(x)
```

```
## [1] 2.5
```

```r
x <- c(2, 1, 6, 3, 1, 5, 1, NA)
mean(x, na.rm=T)
```

```
## [1] 2.714286
```

```r
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.000   2.000   2.714   4.000   6.000       1
```

Table()

- when applied to a factor, `summary()` gives the sample size;
- the same output can be obtained with command `table()`;
- this command is useful especially for cross tabulation.
- Examples:

```r
sex <- c("male", "male", "female", "male", "female")
smoke <- c("yes", "no", "yes", "no", "no")
table(sex, smoke)
```

```
##         smoke
## sex      no yes
##   female  1   1
##   male    2   1
```

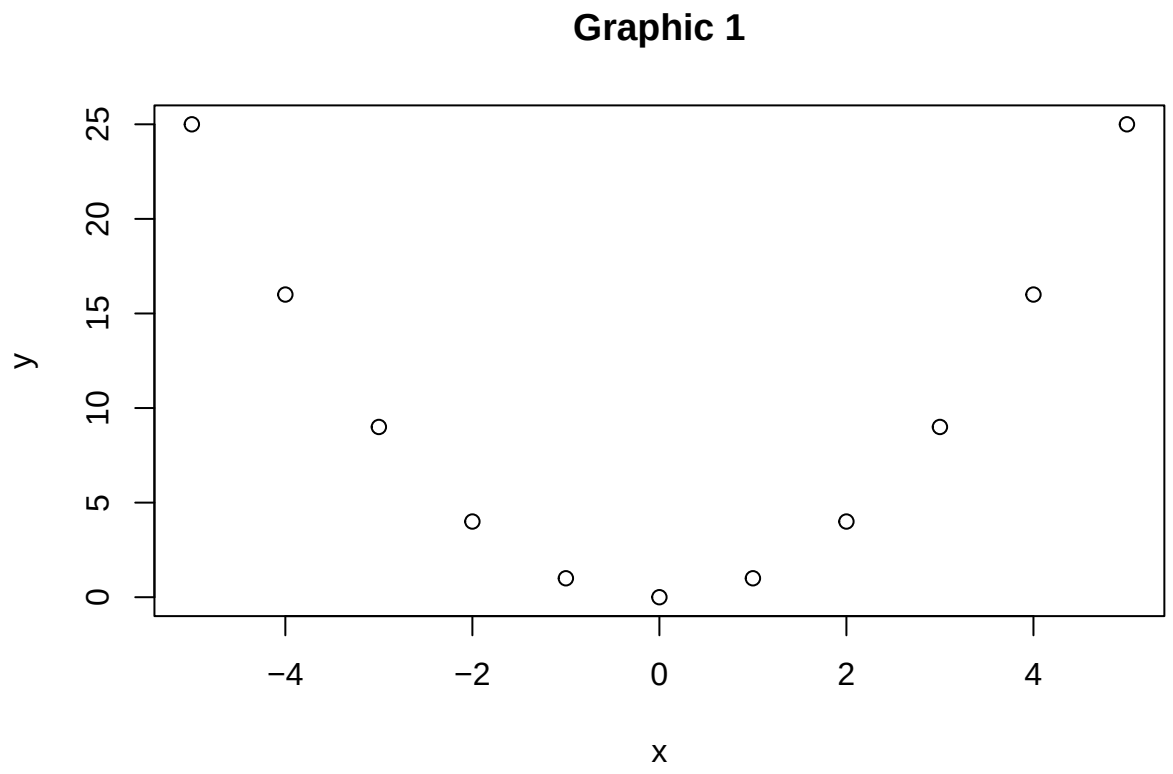
Graphics

- There are several kind of graphics available in R:
- `plot()`; `hist()`; `boxplot()`;
- the title is set by argument `main = 'Main title'`;
- the axes names by argument:
- `xlab = 'x'` for the x-axis; `ylab = 'y'` for the y-axis.
- color can be set by argument col (e.g., `col='red'`, `col='grey'`, . . .).

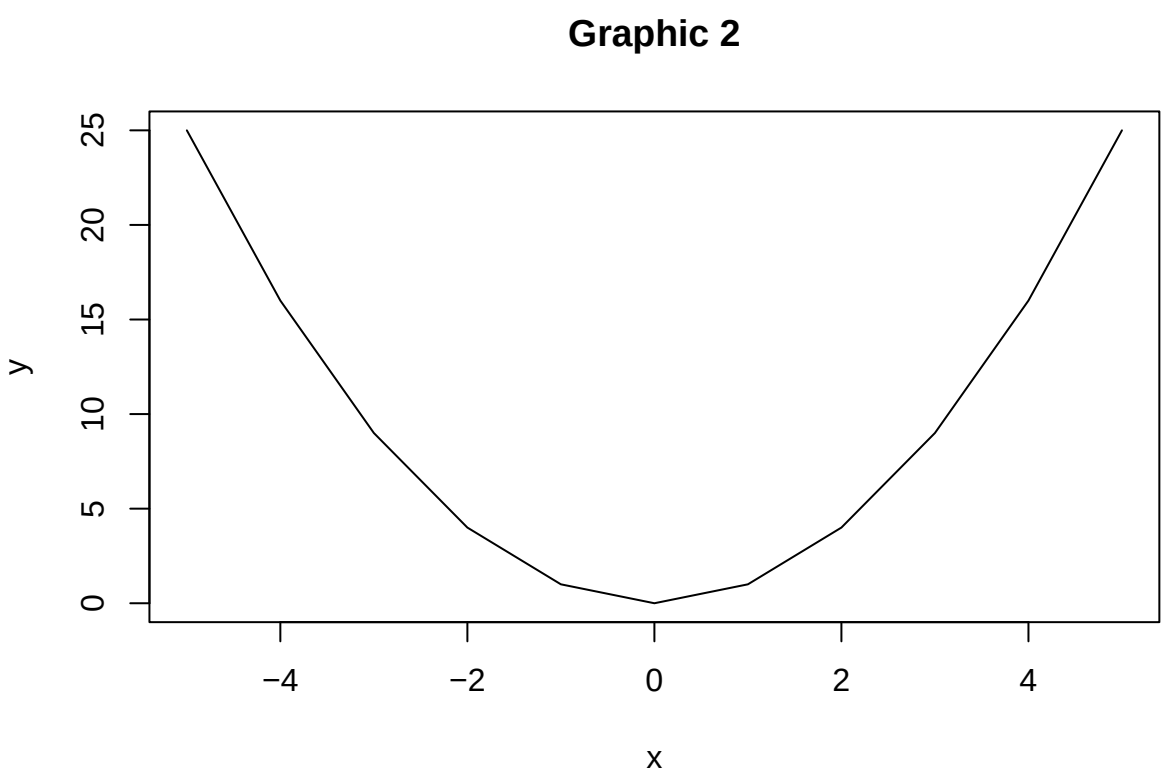
Plot

- It is the more general kind;
- needs the value for the x and the y;
- different types can be chosen (via argument `type`):
- 'p' for points (defaults); 'l' for lines;
- see help for more details.

```
x <- -5:5
y <- (-5:5)^2
plot(x, y, main="Graphic 1")
```



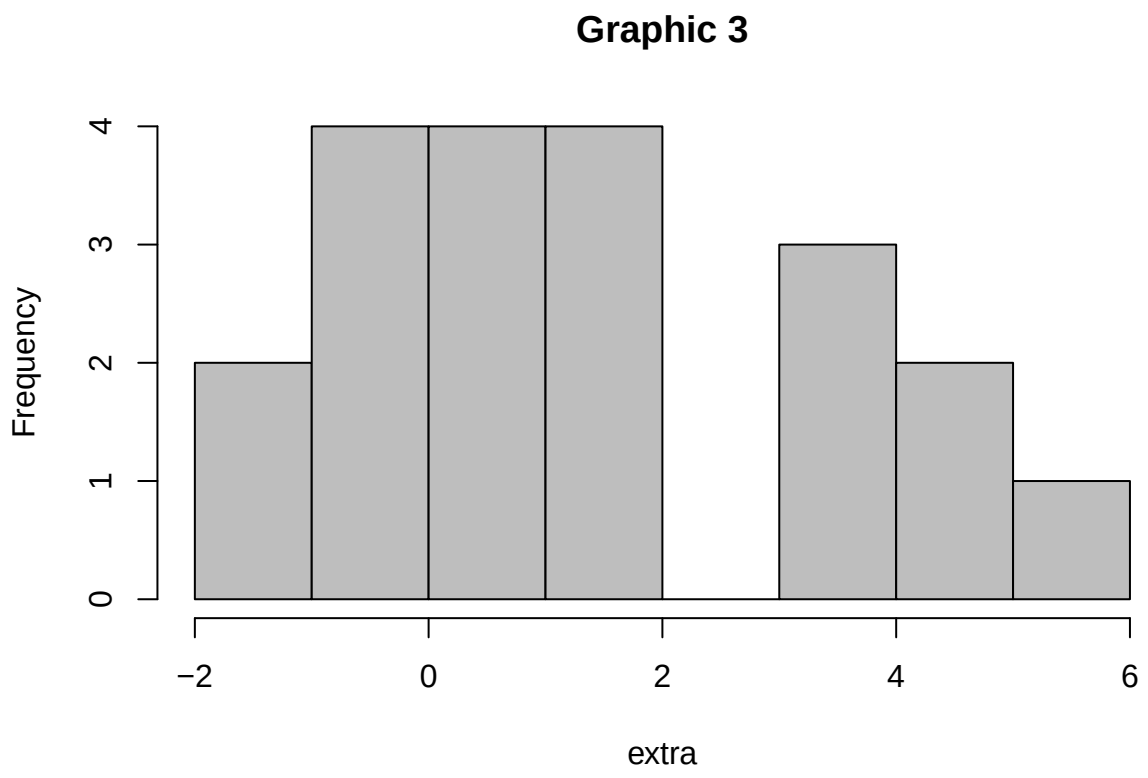

```
plot(y ~ x, main="Graphic 2", type="l")
```



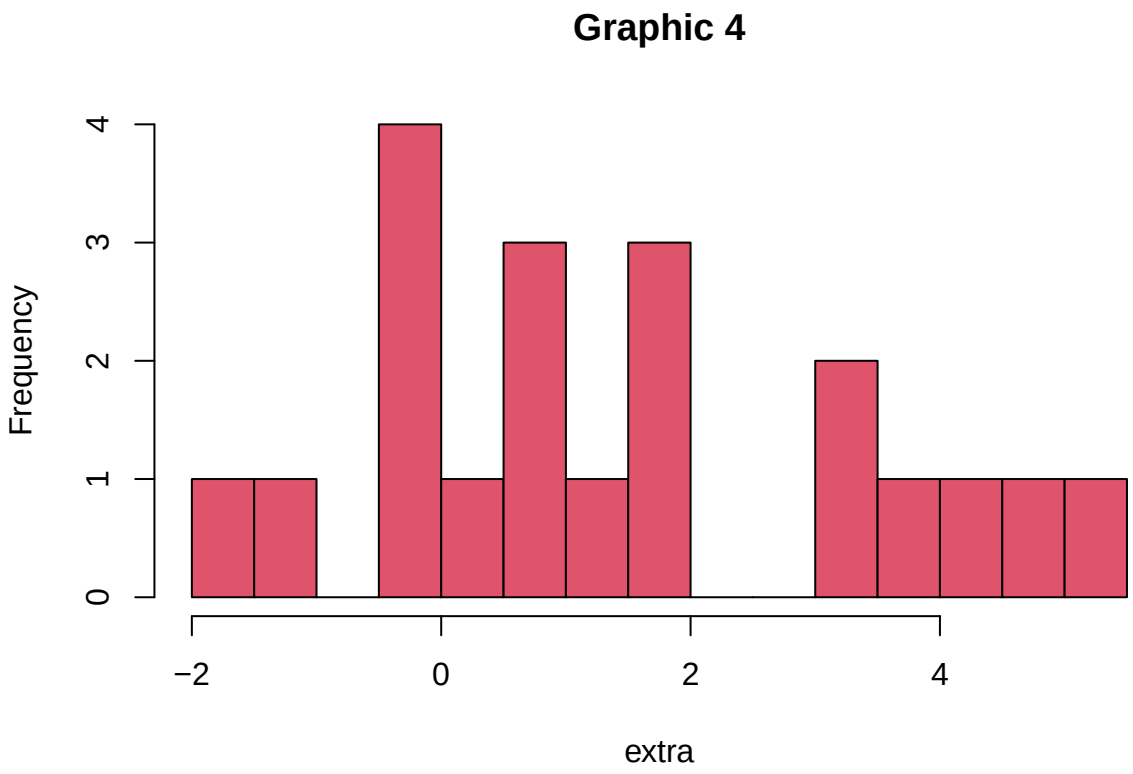
Histogram

- Command `hist()`;
- use the argument breaks to have more or less bars;
- set option freq = TRUE for frequencies, freq = FALSE for densities;

```
attach(sleep)
hist(extra, main="Graphic 3", col="grey")
```




```
hist(extra, main="Graphic 4", breaks=20, col=2)
```



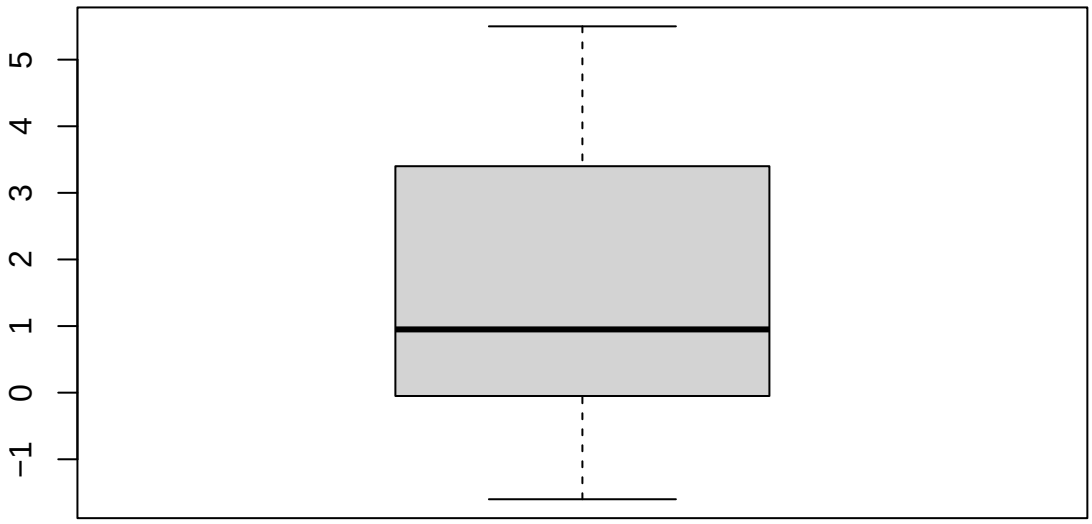
Boxplot

- command `boxplot()`;
- easy to plot with different groups;
- median, 1st and 3rd quartile are plotted:

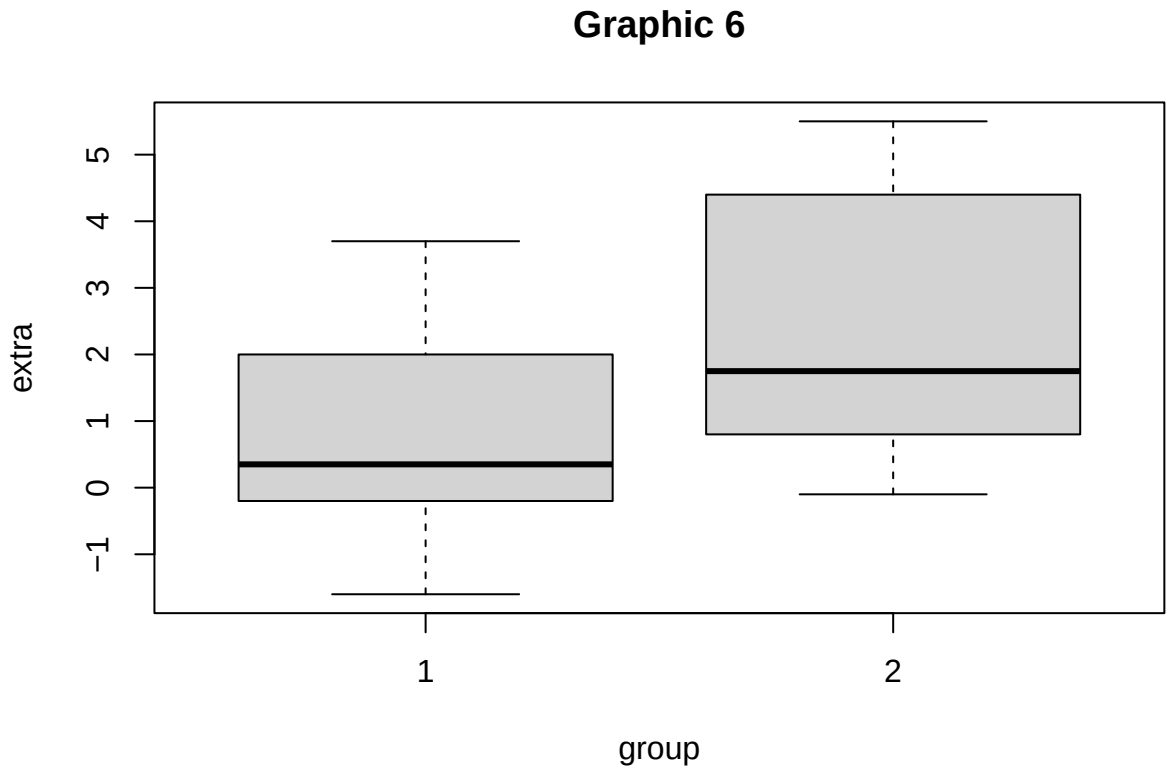
```
attach(sleep)
```

```
## Die folgenden Objekte sind maskiert von sleep (pos = 3):
##
##     extra, group, ID
```

```
boxplot(extra, main="Graphic 5")
```




```r
boxplot(extra ~ group, main="Graphic 6", ylab="extra", xlab="group")
```



Session 3

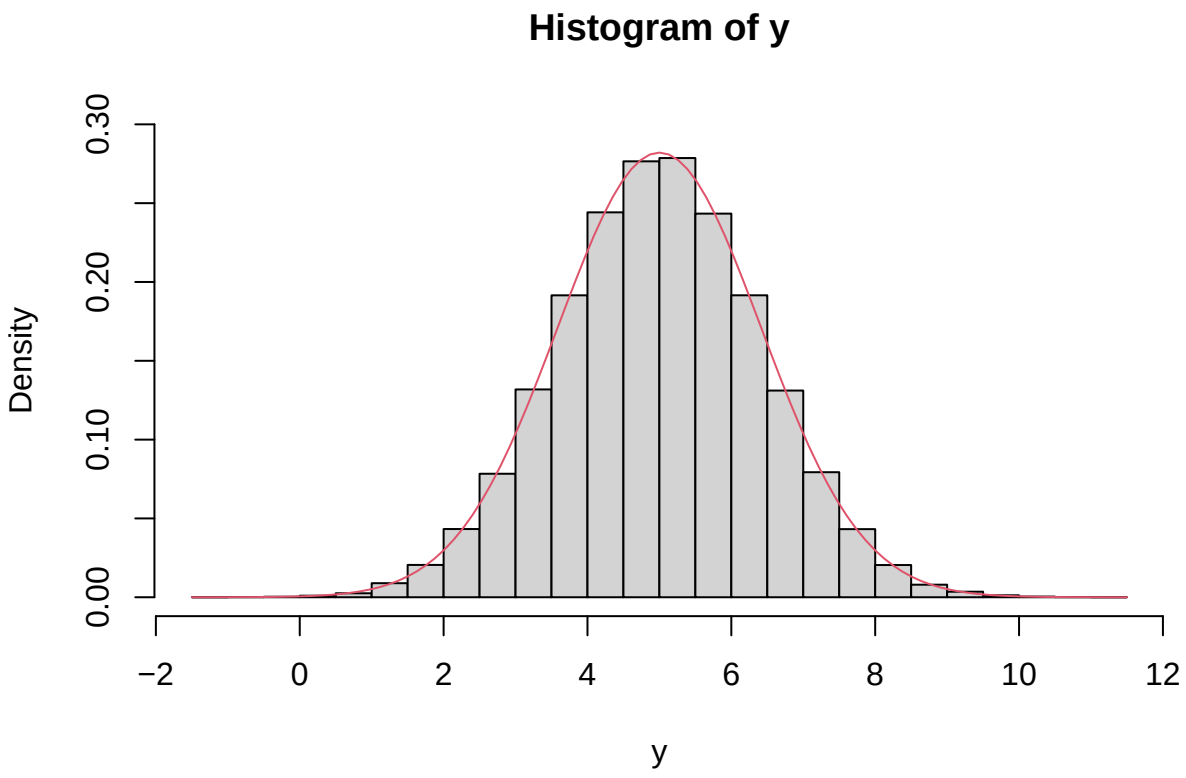
NORMAL DISTRIBUTION

- $Y \sim N(\mu, \sigma^2)$
- generate a sample from a Gaussian distribution: rnorm(n,mean = 0, sd = 1)

n is the sample size; mean is the mean (μ); sd is the square root of the variance (σ).

Example:

```
y <- rnorm(n=100000,mean=5, sd=sqrt(2))
hist(y, freq=F, ylim=c(0, 0.3))
curve(dnorm(x, mean=5, sd=sqrt(2)), col=2, add=T)
```

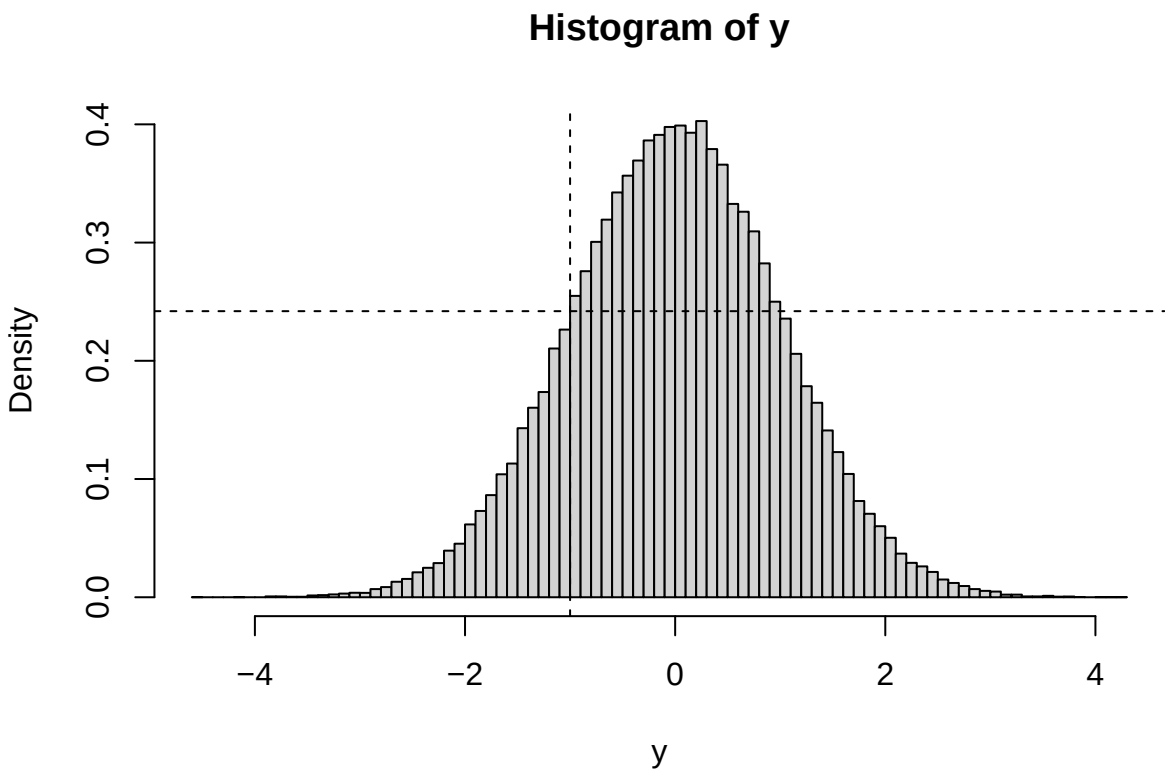


- $Y \sim N(\mu, \sigma^2)$;
- density function: dnorm(x, mean = 0, sd = 1)

```
y <- rnorm(n=100000, mean=0, sd=1)
hist(y, freq=F, ylim=c(0, 0.4), breaks=100)
dnorm(-1)
```

```
## [1] 0.2419707
```

```
hist(y, freq=F, ylim=c(0, 0.4), breaks=100)
abline(v=-1, lty=2)
abline(h=dnorm(-1), lty=2)
```

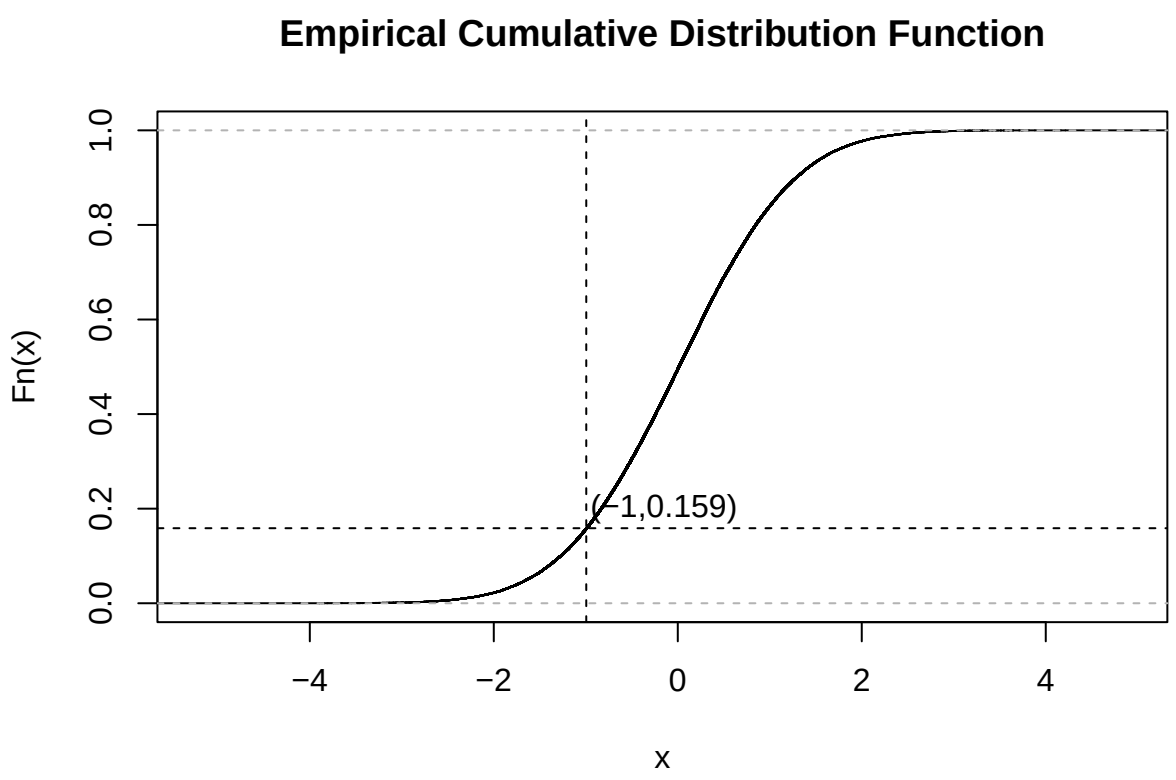


- distribution function: pnorm(q, mean = 0, sd = 1)

```
plot(ecdf(y), main="Empirical Cumulative Distribution Function")
pnorm(-1)
```

```
## [1] 0.1586553
```

```
#library(tigerstats)
#pnormGC(-1, region="below", graph=T)
plot(ecdf(y), main="Empirical Cumulative Distribution Function")
abline(v=quantile(ecdf(y),0.158655254), lty=2)
abline(h=pnorm(-1), lty=2)
text(x=-0.15,y=0.2, labels = "(-1,0.159)")
```



- quantile function: qnorm(p, mean = 0, sd = 1)

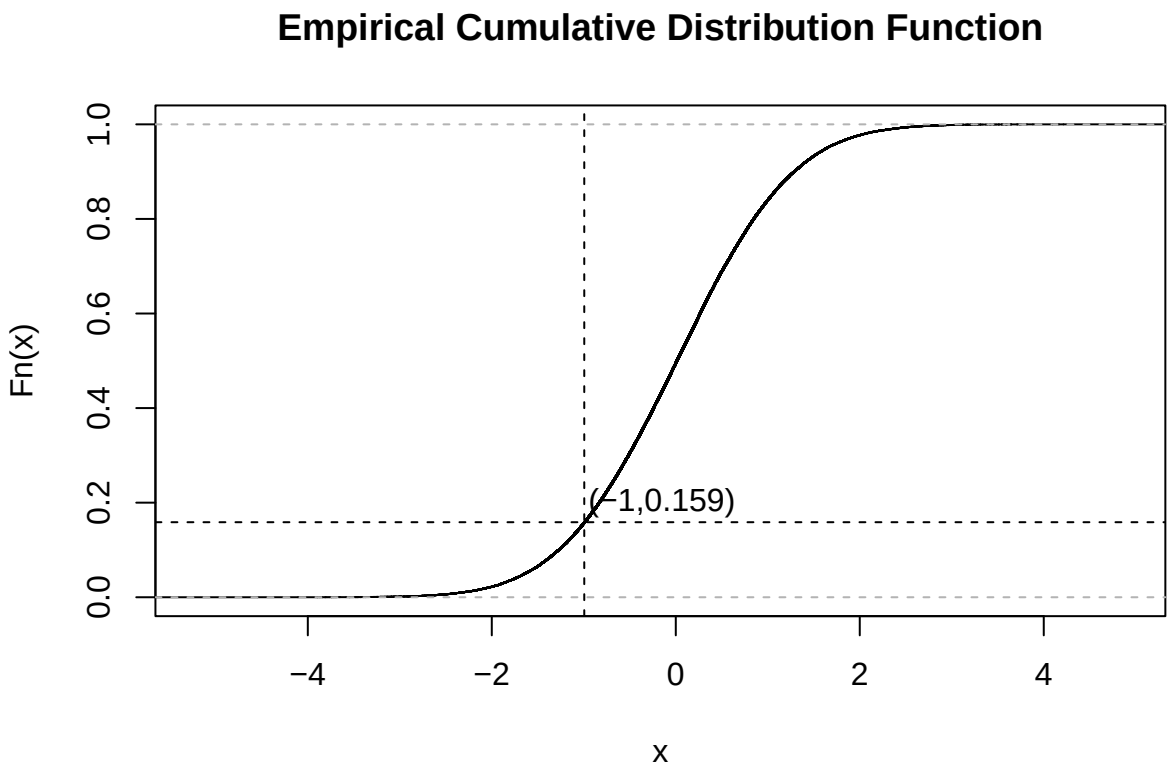
```
qnorm(0.158655254)
```

```
## [1] -1
```

```
quantile(ecdf(y), 0.158655254)
```

```
##  15.86553%
## -0.9949038
```

```
plot(ecdf(y), main="Empirical Cumulative Distribution Function")
abline(v=quantile(ecdf(y), 0.158655254), lty=2)
abline(h=pnorm(-1), lty=2)
text(x=-0.15, y=0.2, labels = "(-1,0.159)")
```



EXAMPLES

Consider $X \sim N(0,1)$. It is very easy to compute the following probabilities with R:

- $P(X \leq 0.89)$

```
pnorm(1.64) #try with 1.96 and 2.57. What do you notice?
```

```
## [1] 0.9494974
```

- $P(X \geq 1.21)$

```
1-pnorm(1.21)
```

```
## [1] 0.1131394
```

- $P(-1.02 \leq X \leq 1.98)$

```
pnorm(1.98)-pnorm(-1.02)
```

```
## [1] 0.822284
```

- $P(\mid X \mid \leq 0.92)$

```r
pnorm(0.92)-pnorm(-0.92)
```

```
## [1] 0.6424272
```

- $P(|\ X\ | \geq 1.11)$

```r
(1-pnorm(1.11))+pnorm(-1.11)
```

```
## [1] 0.266999
```

OTHER DISTRIBUTIONS

We can use other distributions like the Normal distribution, for example - binomial: rbinom, dbinom, pbinom, qbinom

- uniform: runif, dunif, punif, qunif
- Student's t: rt, dt, pt, qt
- ...

Remember to use the help!!!

Functions cont'd

We can build our own functions in R to run specific tasks.

Here is a rather first example of a function

```r
add_two <- function(x, y){
  z <- x + y
  return(z)
}
```

The new function `add_two` takes two arguments (`x` and `y`) and returns the sum.

A more elaborate function which we can use afterwards: The following self-built function aims to derive the mode of a distribution from its density function.

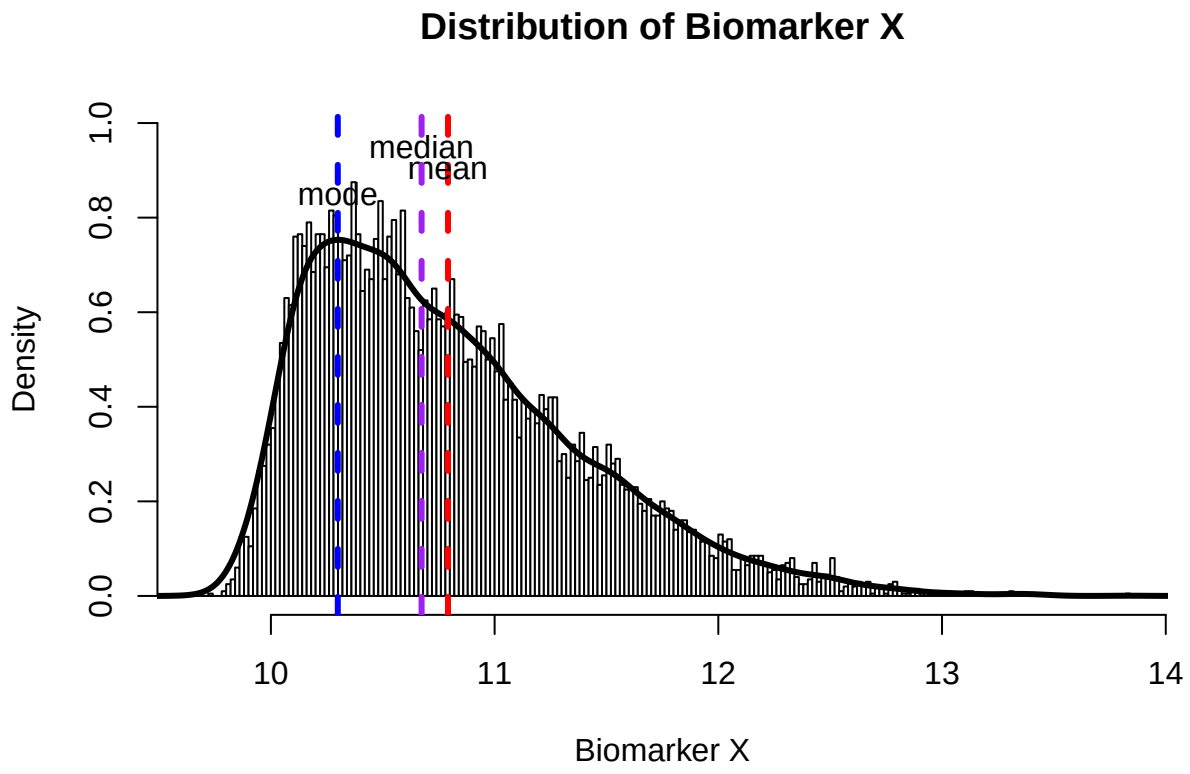
```r
Mode <- function(x) {
 return(density(x)$x[which(density(x)$y==max(density(x)$y))])
}
```

Here assign a function with the name `Mode`. It takes one argument: `x`. Inside the function we use various subfunction to fulfill a specific task (calculate the mode of `x`).

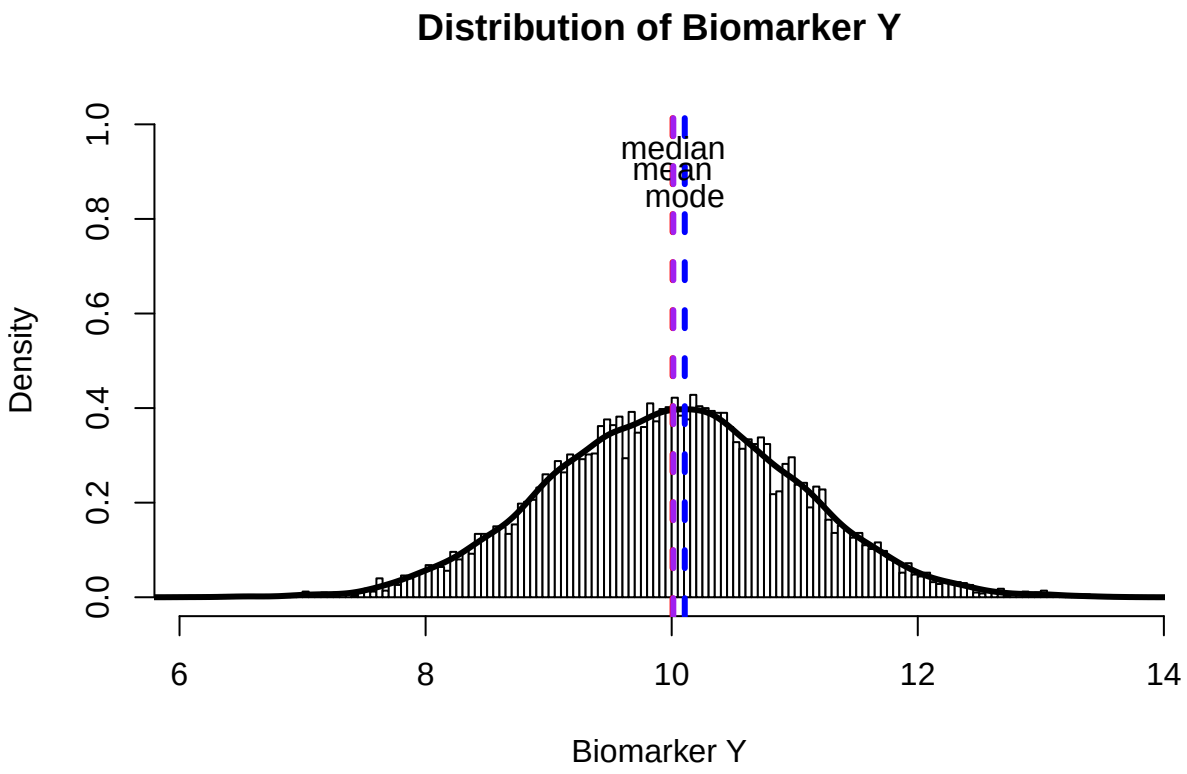
Hands-on:

Load the workspace "biomarker.RData". This workspace includes simulated data of 3 (imaginary) biomarkers X, Y, and Z that were randomly collected in the population. One biomarker is standard normal distributed. The other two biomarkers have skewed distribution. Find out which one has which distribution. How many values were collected for each biomarker? Hint: use the function hist(), length(). You can also try to visualize the mean, median and mode of each distribution on the histogram.

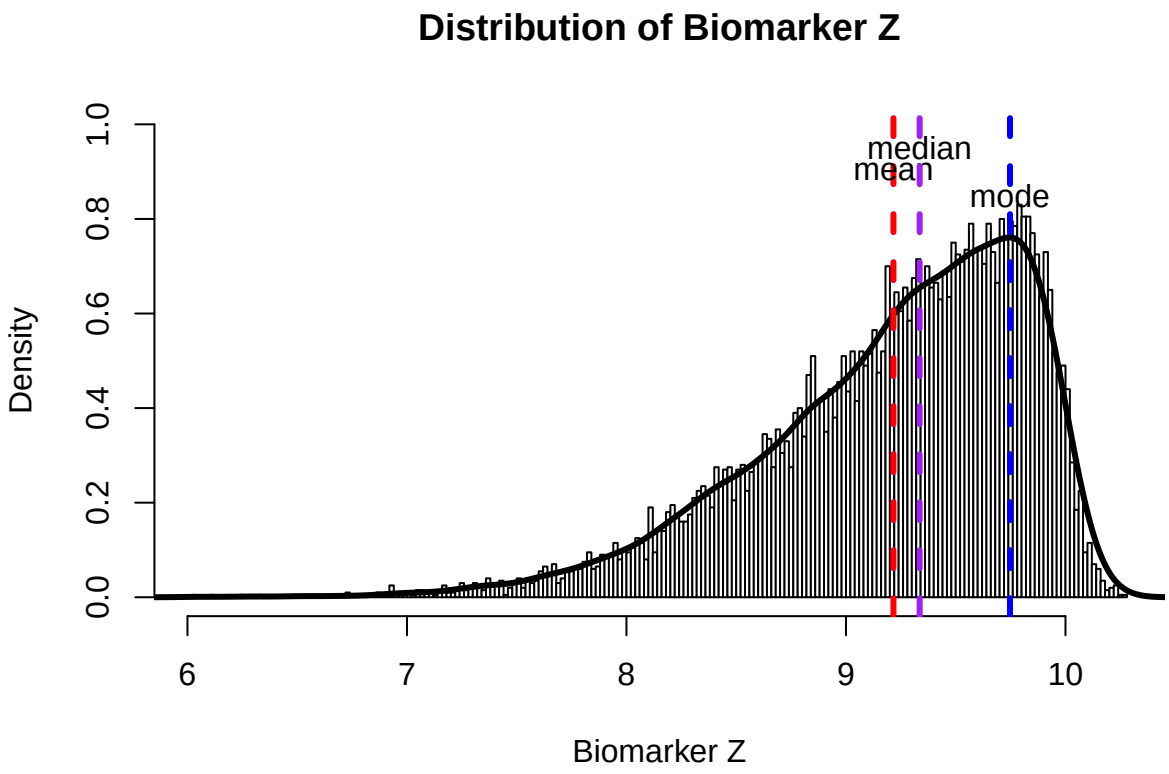
```
load("biomarker.RData")

myhist <- function (X, label ="") {
  hist(X,freq=F,main=paste0("Distribution of ", label), xlab=label, breaks=200, col="grey100", ylim=c(0
  lines(density(X), col="black", lwd=3)
  abline(v=Mode(X), col="blue", lwd=3, lty=2)
  text(x=Mode(X), y=0.85,"mode")
  abline(v=mean(X), col="red",lwd=3, lty=2)
  text(x=mean(X), y=0.9,"mean")
  abline(v=median(X), col="purple", lwd=3, lty=2)
  text(x=median(X), y=0.95, "median")
}

myhist(X, "Biomarker X")
```




```
myhist(Y, "Biomarker Y")
```



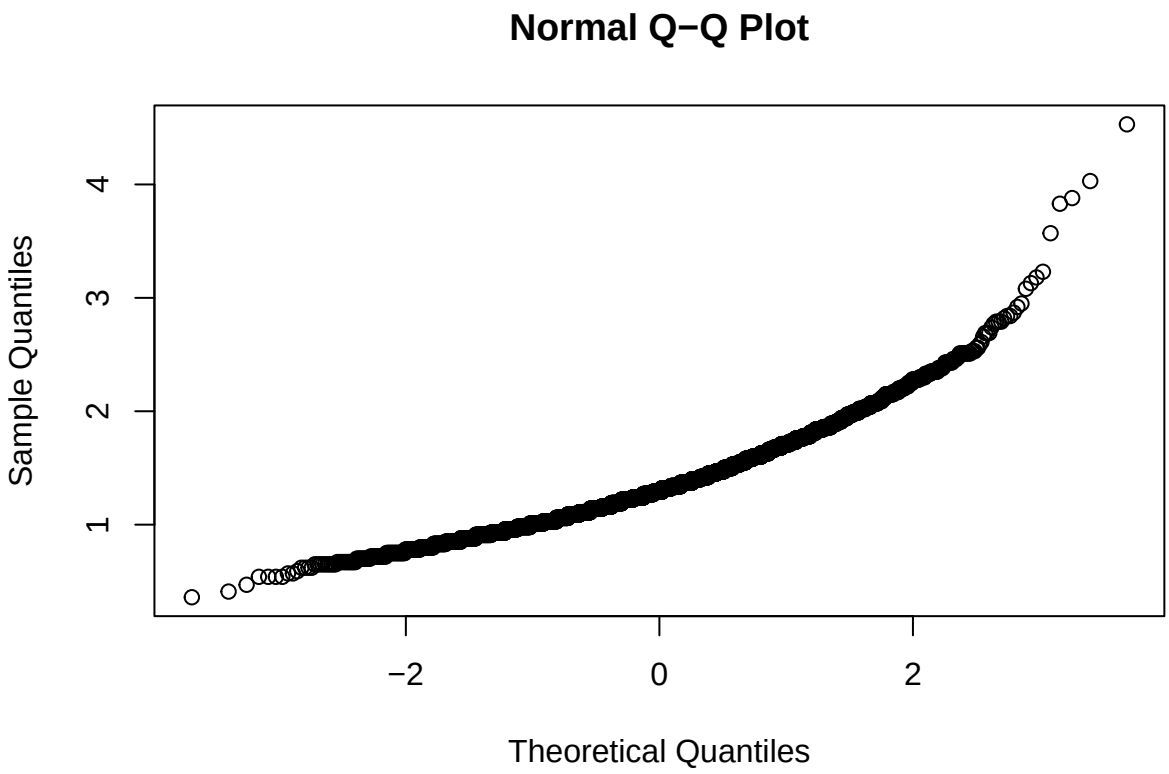

```
myhist(Z, "Biomarker Z")
```



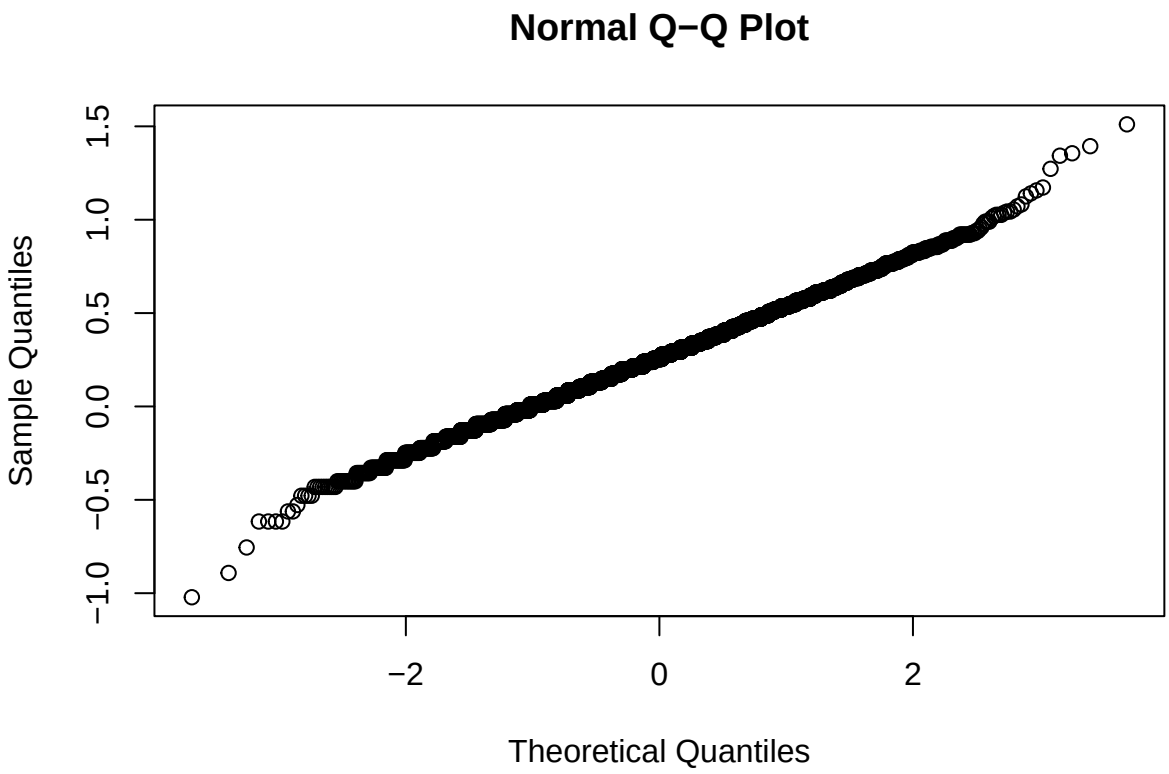
QQ-PLOT and QQ-NORM

- qqplot compares the quantiles of two distributions;
- qqnorm compares the quantiles of a distribution with the quantiles of a standard normal distribution;
- qqline draws the line of a normal quantile-quantile plot; Example:

```
load("myNhanes.RData")
attach(mydata)
qqnorm(hdl)
```




```
qqnorm(log(hdl))
```



Session 4

CONFIDENCE INTERVALS

- it is an interval estimate;
- we construct an interval following a procedure that, if applied in a large number of replications of the experiment, gives intervals which contain the true value $1 - \alpha$ of the time;
- $1 - \alpha$ is called confidence level;
- usual values are 0.90, 0.95 and 0.99.

EXAMPLES of CONFIDENCE INTERVALS

- true mean, gaussian distribution, known variance:

```
x <- rnorm(100,mean=1,sd=2)
mean.x <- mean(x)
low <- mean.x - qnorm(0.975)*2/sqrt(100)
up <- mean.x + qnorm(0.975)*2/sqrt(100)
c(low,up)
```

```
## [1] 0.629951 1.413937
```

- true mean, gaussian distribution, unknown variance:

```
sd.x <- sd(x)
low <- mean.x - qt(0.975, df=99)*sd.x/sqrt(100)
up <- mean.x + qt(0.975, df=99)*sd.x/sqrt(100)
c(low,up)
```

```
## [1] 0.5768293 1.4670582
```

Session 5

STUDENT'S ONE-SAMPLE TEST

- $H_0 : \mu = \mu_0$
- $H_1 : \mu \neq \mu_0$

`t.test(x, mu=`μ_0`, alternative='two-sided')`

`t.test(x, mu=`μ_0`, alternative='greater')`

`t.test(x, mu=`μ_0`, alternative='less')`

- default alternative is 'two-sided'

```
x <- rnorm(100, mean=3, sd=1)
t.test(x, mu=1)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 17.207, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 1
## 95 percent confidence interval:
##  2.63852 3.06567
## sample estimates:
## mean of x
##  2.852095
```

Further examples:

```
t.test(x, mu=1, alternative="greater")
```

```
##
##  One Sample t-test
##
## data:  x
## t = 17.207, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 1
## 95 percent confidence interval:
```

```
##  2.673376      Inf
## sample estimates:
## mean of x
##  2.852095
```

```r
t.test(x, mu=1, alternative="less")
```

```
##
##  One Sample t-test
##
## data:  x
## t = 17.207, df = 99, p-value = 1
## alternative hypothesis: true mean is less than 1
## 95 percent confidence interval:
##      -Inf 3.030814
## sample estimates:
## mean of x
##  2.852095
```

TWO-SAMPLE T-TEST (UNEQUAL VARIANCE)

- $Var[X] \neq Var[Y]$
- $H_0 : \mu_X = \mu_Y$

```
t.test(x, y, alternative=c('two.sided', 'less','greater'))
```

Example:

```r
x <- rnorm(100, mean=4, sd=5)
y <- rnorm(100, mean=2, sd=2)
t.test(x, y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 3.4092, df = 126.8, p-value = 0.0008739
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.7403491 2.7888004
## sample estimates:
## mean of x mean of y
##  3.764732  2.000157
```

TWO-SAMPLE T-TEST (EQUAL VARIANCE)

- $Var[X] = Var[Y]$
- $H_0 : \mu_X = \mu_Y$

```
t.test(x, y, var.equal=T,alternative=c('two.sided', 'less', 'greater'))
```

Example:

```
x <- rnorm(100, mean=4, sd=1)
y <- rnorm(100, mean=2, sd=1)
t.test(x, y, var.equal=T)
```

```
##
##  Two Sample t-test
##
## data:  x and y
## t = 13.918, df = 198, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.617393 2.151392
## sample estimates:
## mean of x mean of y
##  3.988982  2.104589
```

F-TEST FOR COMPARING VARIANCES

- $H_0 : \sigma_X^2 = \sigma_Y^2$
- $H_1 : \sigma_X^2 \neq \sigma_Y^2$

```
var.test(x, y, ratio = 1,alternative=c('two.sided', 'less', 'greater'),conf.level = 0.95)
```

Example:

```
x <- rnorm(80, mean=4, sd=5)
y <- rnorm(20, mean=2, sd=2)
var.test(x, y)
```

```
##
##  F test to compare two variances
##
## data:  x and y
## F = 4.0362, num df = 79, denom df = 19, p-value = 0.001228
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.803451 7.692916
## sample estimates:
## ratio of variances
##           4.036189
```

PROPORTION TEST 1

- $H_0 : p = p_0$
- $H_1 : p \neq p_0$

```
prop.test(x, n, p)
```

- x: number of successes
- n: total number of trials
- p: proportion to be tested.
- Example:

```
#Read data table saved in this directory
load(file="myNhanes.RData")
prop.test(sum(mydata$male), length(mydata$male), p=0.5)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  sum(mydata$male) out of length(mydata$male), null probability 0.5
## X-squared = 0.2738, df = 1, p-value = 0.6008
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.4898438 0.5177503
## sample estimates:
##      p
## 0.5038
```

PROPORTION TEST 2

- $H_0 : p_1 = p_2$
- $H_1 : p_1 \neq p_2$

```
prop.test(table(x, y))
```

- x: first (categorical) variable
- y second (binary) variable
- Example:

```
# prop.test(table(educ, male))
tbl <- table(mydata$educ, mydata$male)
prop.test(tbl)
```

```
##
##  5-sample test for equality of proportions without continuity correction
##
## data:  tbl
## X-squared = 13.481, df = 4, p-value = 0.00915
## alternative hypothesis: two.sided
## sample estimates:
##    prop 1    prop 2    prop 3    prop 4    prop 5
## 0.5218295 0.5082707 0.5292929 0.4605078 0.4881356
```

Session 6

Introduction

- in the previous lectures, we have seen how to perform a t-test and a F-test:

```
t.test(height[male==T],height[male==F])

var.test(height[male==T],height[male==F])
```

- and a Z-test, testing the equality of proportions:

```
table(male,heartdis_ever,dnn=c("Male","Heart disease"))
prop.table(table(male,!heartdis_ever,dnn=c("Male","Heart disease")),1)
prop.test(table(male,!heartdis_ever))
```

Today we see how to perform in R the other tests which you have learnt in the theoretical lectures (Chi-square test, Fisher test, McNemar test, Sign test, Wilcoxon test, Mann-Whitney test).

Chi-square test

- $H_0 : \pi_1 = \pi_2 = \pi$

```
chisq.test(x, y = NULL, correct = TRUE, p =rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

- you can pass to the function:

a contingency table; directly two categorical vectors.

Examples:

```
x <- c("M","F","M","F","M","M","F","F","M")
y <- c("Y","N","N","N","Y","Y","N","Y","N")
chisq.test(x,y)
```

```
## Warning in chisq.test(x, y): Chi-Quadrat-Approximation kann inkorrekt sein

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  x and y
## X-squared = 0.14062, df = 1, p-value = 0.7077
```

. . . or, with a contingency table,

```
tab <- table(x,y)
chisq.test(tab)
```

```
## Warning in chisq.test(tab): Chi-Quadrat-Approximation kann inkorrekt sein

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab
## X-squared = 0.14062, df = 1, p-value = 0.7077
```

Fisher test

- exact test

```
fisher.test(x, y = NULL, workspace = 200000,
    hybrid = FALSE, control = list(), or = 1,
    alternative = "two.sided", conf.int = TRUE,
    conf.level = 0.95, simulate.p.value = FALSE, B =
    2000)
```

- you can pass to the function two factor vectors

Examples:

```
fisher.test(x,y)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  x and y
## p-value = 0.5238
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##    0.1497998 312.5621395
## sample estimates:
## odds ratio
##   3.764195
```

. . . or a contingency table

```
tab <- table(x,y)
fisher.test(tab)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tab
## p-value = 0.5238
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##    0.1497998 312.5621395
## sample estimates:
## odds ratio
##   3.764195
```

McNemar test

- symmetry of rows and columns;

```
mcnemar.test(x, y = NULL, correct = TRUE)
```

- you can pass to the function two factor vectors

Examples:

```
mcnemar.test(x,y)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  x and y
## McNemar's chi-squared = 0, df = 1, p-value = 1
```

. . . or a contingency table

```
tab <- table(x,y)
mcnemar.test(tab)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  tab
## McNemar's chi-squared = 0, df = 1, p-value = 1
```

Non-parametric Tests:

Sign test

- $H_0 : median = md_0$
- we need the package BSDA

```
SIGN.test(x, y = NULL, md = 0, alternative = "two.sided", conf.level = 0.95)
```

Example:

```
library(BSDA)
```

```
## Lade nötiges Paket: lattice

##
## Attache Paket: 'BSDA'

## Das folgende Objekt ist maskiert 'package:datasets':
##
##     Orange
```

```r
x <- rpois(100,5)
SIGN.test(x, md=5)
```

```
##
##  One-sample Sign-Test
##
## data:  x
## s = 35, p-value = 0.2664
## alternative hypothesis: true median is not equal to 5
## 95 percent confidence interval:
##  4 5
## sample estimates:
## median of x
##           5
##
## Achieved and Interpolated Confidence Intervals:
##
##                   Conf.Level L.E.pt U.E.pt
## Lower Achieved CI     0.9431      4      5
## Interpolated CI       0.9500      4      5
## Upper Achieved CI     0.9648      4      5
```

Wilcoxon test

- $H_0 : median = md_0$
- test based on ranks

```r
wilcox.test(x, y = NULL, alternative = c("two.sided", "less", "greater"), mu = 0,
            paired = FALSE, exact = NULL, correct = TRUE, conf.int = FALSE,
            conf.level = 0.95, ...)
```

Example:

```r
x <- rpois(100,5)
wilcox.test(x, mu=5)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  x
## V = 1524.5, p-value = 0.3151
## alternative hypothesis: true location is not equal to 5
```

Recall: Sign test vs. Wilcoxon test vs. t-test

Sign test:

- Applicable for ordinal or higher scale level

- No distributional assumptions
- Low power

Wilcoxon test:

- Applicable for discrete or continuous quantitative data
- Symmetrical distribution required
- Intermediate power

T-test:

- Normal distribution required
- Highest power

Compare all tests on the same data (we assume Normal distributed data!)

```
set.seed(123)
x <- rnorm(20,0.5)
c(
  signTest = BSDA::SIGN.test(x, md = 0)$p.value,
  wilcoxonTest = wilcox.test(x,mu=0)$p.value,
  tTest = t.test(x, mu = 0)$p.value
)
```

```
##     signTest wilcoxonTest        tTest
##   0.04138947   0.01068878   0.00822065
```

Mann-Whitney U test

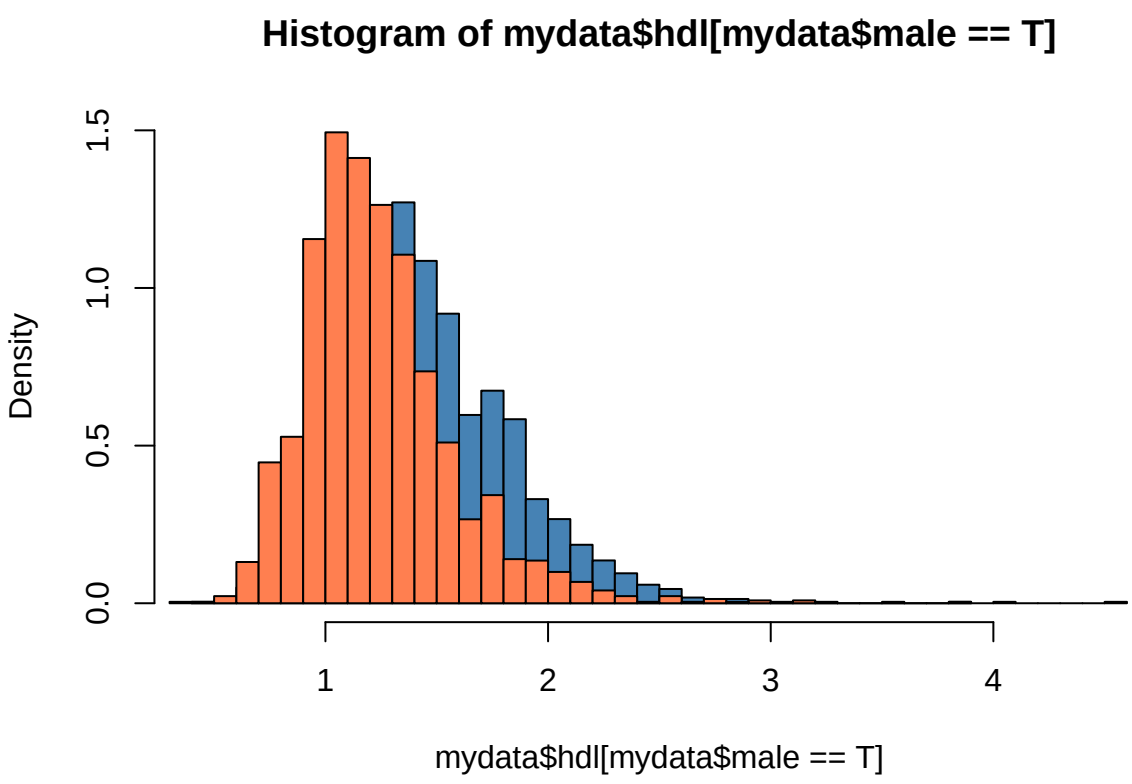
- $H_0 : median_1 = median_2$
- non-parametric test for independent samples
- package coin (Why extra package and not using `wilcox.test`? -> https://stats.stackexchange.com/questions/31417/what-is-the-difference-between-wilcox-test-and-coinwilcox-test-in-r)

```
wilcox_test(formula, data, subset = NULL, weights= NULL, ...)
```

- needs a formula, e.g. y~x.

Example hdl in NHANES data (not normal):

```
hist(mydata$hdl[mydata$male == T], breaks = 40, col = "steelblue", freq = F, ylim=c(0,1.5))
hist(mydata$hdl[mydata$male == F], breaks = 40, col = "coral", add = T, freq = F)
```




```
coin::wilcox_test(hdl ~ as.factor(male), data = mydata)
```

```
##
##  Asymptotic Wilcoxon-Mann-Whitney Test
##
## data:  hdl by as.factor(male) (FALSE, TRUE)
## Z = -21.838, p-value < 2.2e-16
## alternative hypothesis: true mu is not equal to 0
```

```
wilcox.test(hdl ~ as.factor(male), data = mydata)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  hdl by as.factor(male)
## W = 1520720, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Session 7

```
#Read data table saved in this directory
load(file="myNhanes.RData")
attach(mydata)
```

Correlation

```
cor(X,Y)

cor(x, y = NULL, use = "everything", method = c("pearson", "kendall", "spearman"))
```

- correlation between x and y
- with incomplete observations, we may want to set use =“complete.obs”
- based on the argument method, the function cor() computes
  - Pearson’s r (default)
  - Kendall’s τ (rank based)
  - Spearman’s ρ (rank based)

Example:

```
x <- rnorm(10)
y <- rnorm(10)
cor(x,y)
```

```
## [1] -0.4372649
```

Correlation test

- test for correlation between paired samples

```
cor.test(x, y, alternative = c("two.sided", "less", "greater"), method = c("pearson", "kendall", "spearm
```

- H_1 is defined through the argument alternative
- the argument `method` sets the correlation coefficient

Example:

```
cor.test(x,y)
```

```
##
##  Pearson's product-moment correlation
##
## data:  x and y
## t = -1.3752, df = 8, p-value = 0.2063
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8365717  0.2654408
## sample estimates:
##        cor
## -0.4372649
```

Linear model

$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + ... + \epsilon$,with $\epsilon \sim N(0, \sigma^2)$

R command: `lm(formula, data, ...)`

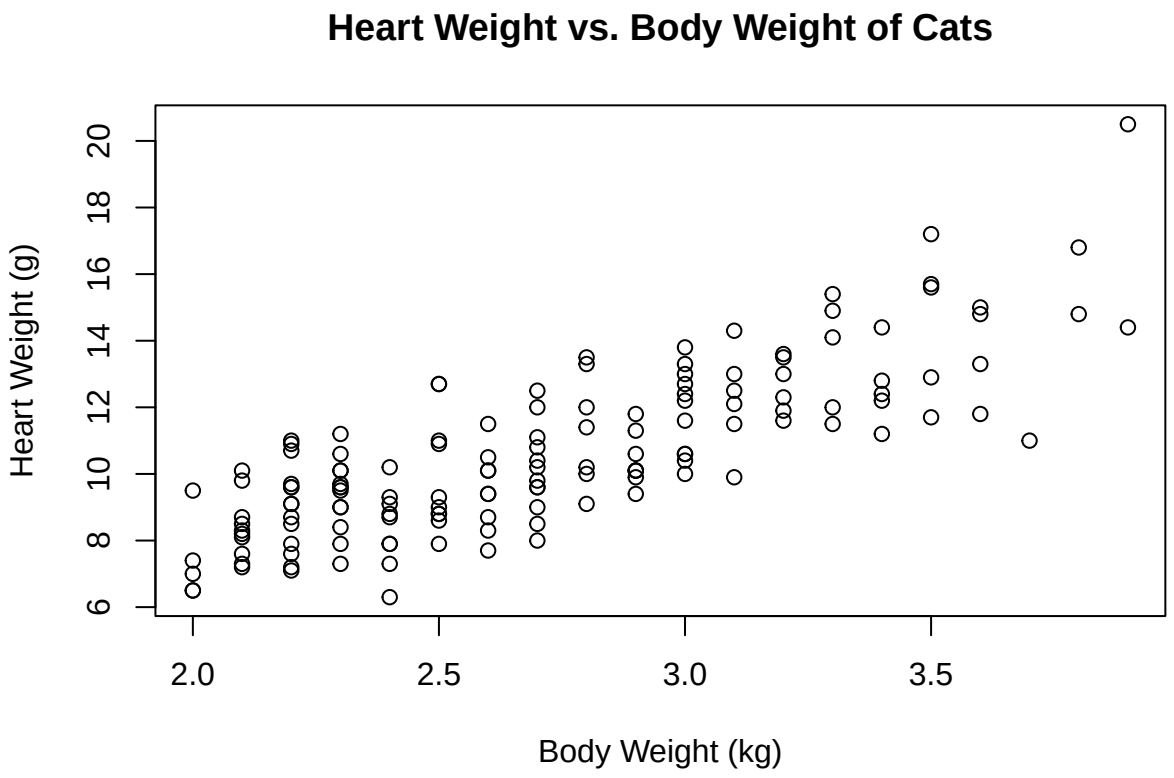
examples of `formula`:

- `y~age+sex`
- `y~I(log(age)) + as.factor(sex)`
- see `?formula` for more details

Note: `data` must be a data.frame.

Example:

```
library(MASS)
data(cats)
with(cats, plot(Bwt, Hwt, xlab="Body Weight (kg)",
        ylab="Heart Weight (g)",
         main="Heart Weight vs. Body Weight of Cats"))
```




```
with(cats, cor.test(Bwt, Hwt))
```

```
##
##  Pearson's product-moment correlation
##
## data:  Bwt and Hwt
## t = 16.119, df = 142, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7375682 0.8552122
## sample estimates:
##       cor
## 0.8041274
```

```
lm(Hwt ~ Bwt, data=cats)
```

```
##
## Call:
## lm(formula = Hwt ~ Bwt, data = cats)
##
## Coefficients:
## (Intercept)          Bwt
##     -0.3567       4.0341
```

Function summary()

- The output of `summary()` provides an overview on the model:

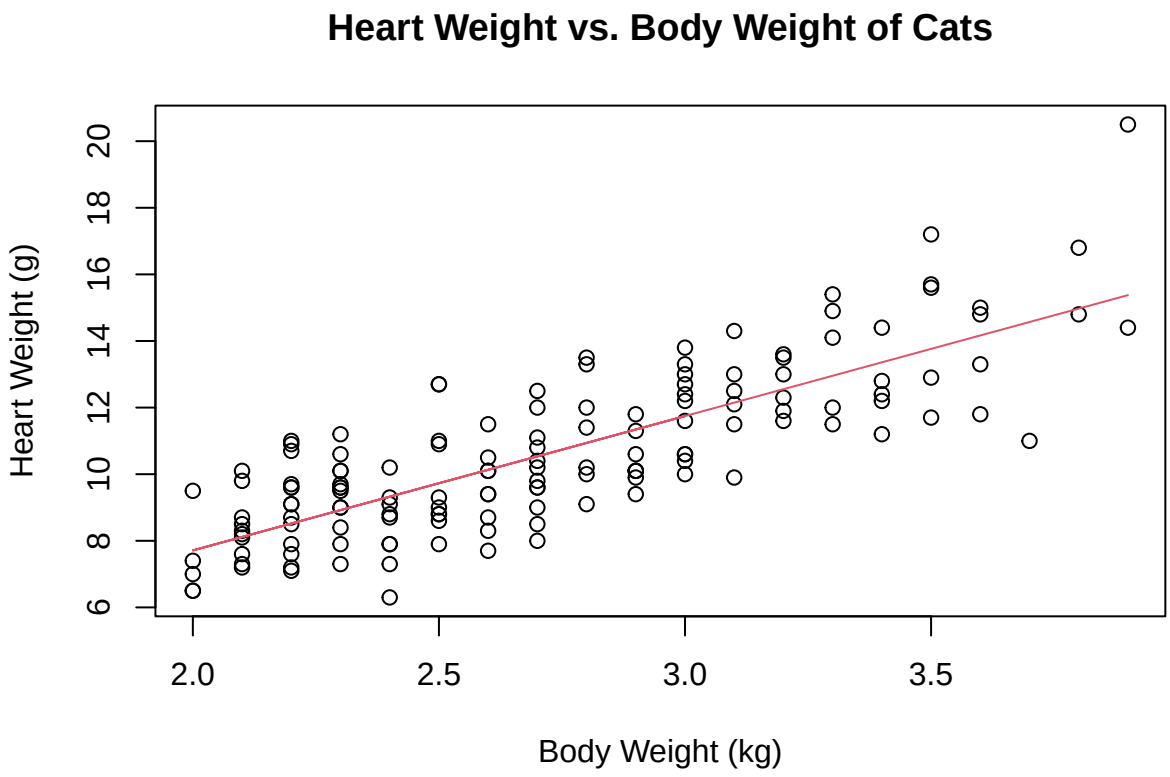
```
model <- lm(Hwt ~ Bwt, data=cats)
summary(model)
```

```
##
## Call:
## lm(formula = Hwt ~ Bwt, data = cats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5694 -0.9634 -0.0921  1.0426  5.1238
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.3567     0.6923  -0.515    0.607
## Bwt           4.0341     0.2503  16.119   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.452 on 142 degrees of freedom
## Multiple R-squared:  0.6466, Adjusted R-squared:  0.6441
## F-statistic: 259.8 on 1 and 142 DF,  p-value: < 2.2e-16
```

Graphical diagnosis

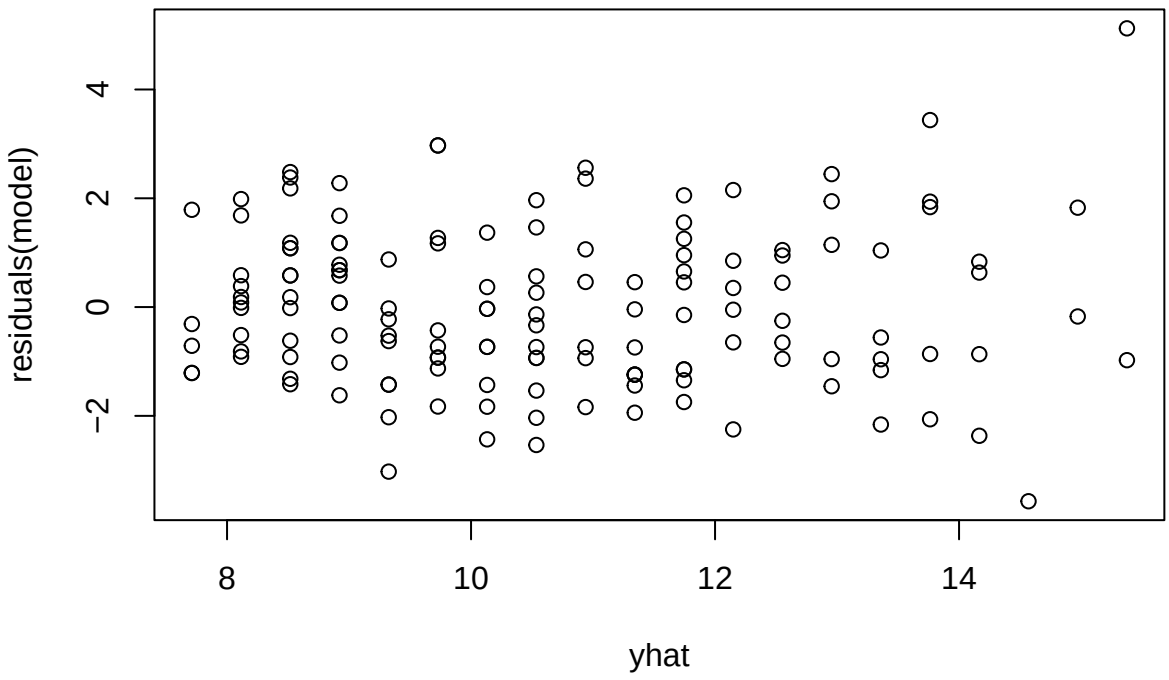
- Plot the regression line

```r
plot(cats$Bwt, cats$Hwt, xlab="Body Weight (kg)",
        ylab="Heart Weight (g)",
         main="Heart Weight vs. Body Weight of Cats")
yhat <- coef(model)[1] + coef(model)[2]*cats$Bwt

lines(x=cats$Bwt, y=yhat, col=2)
```

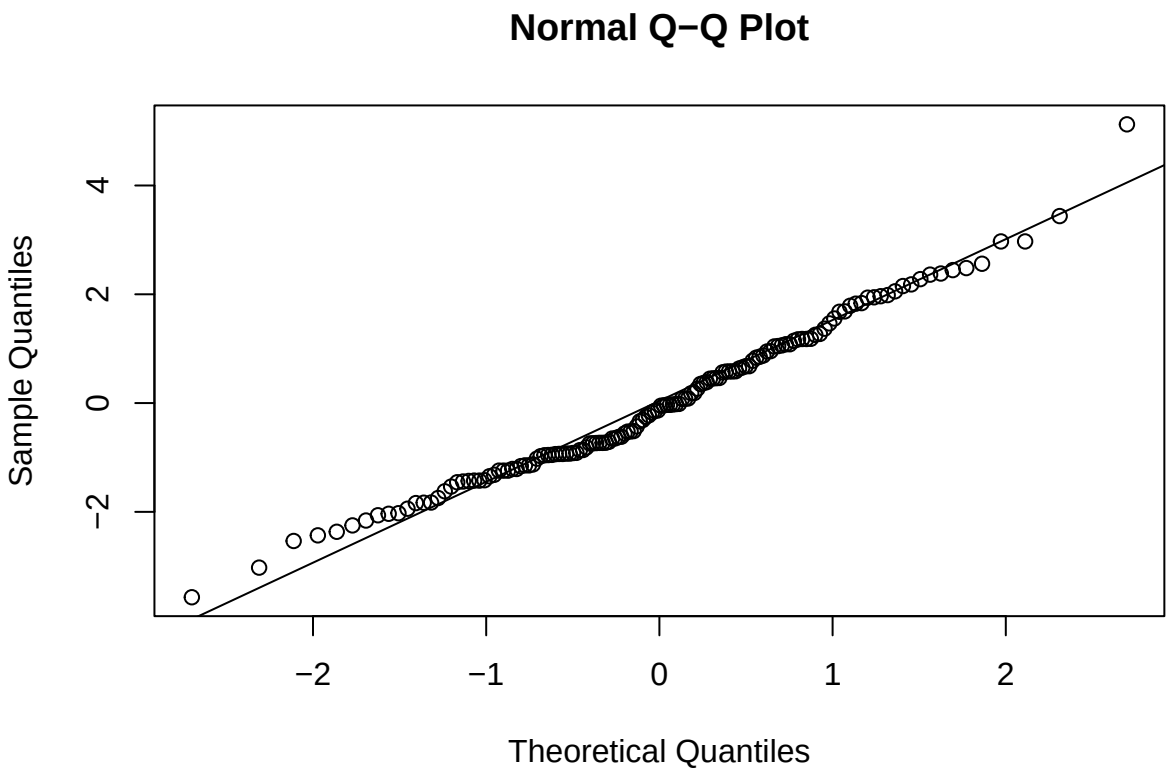


- Use the function `plot()` for diagnosis graphics:

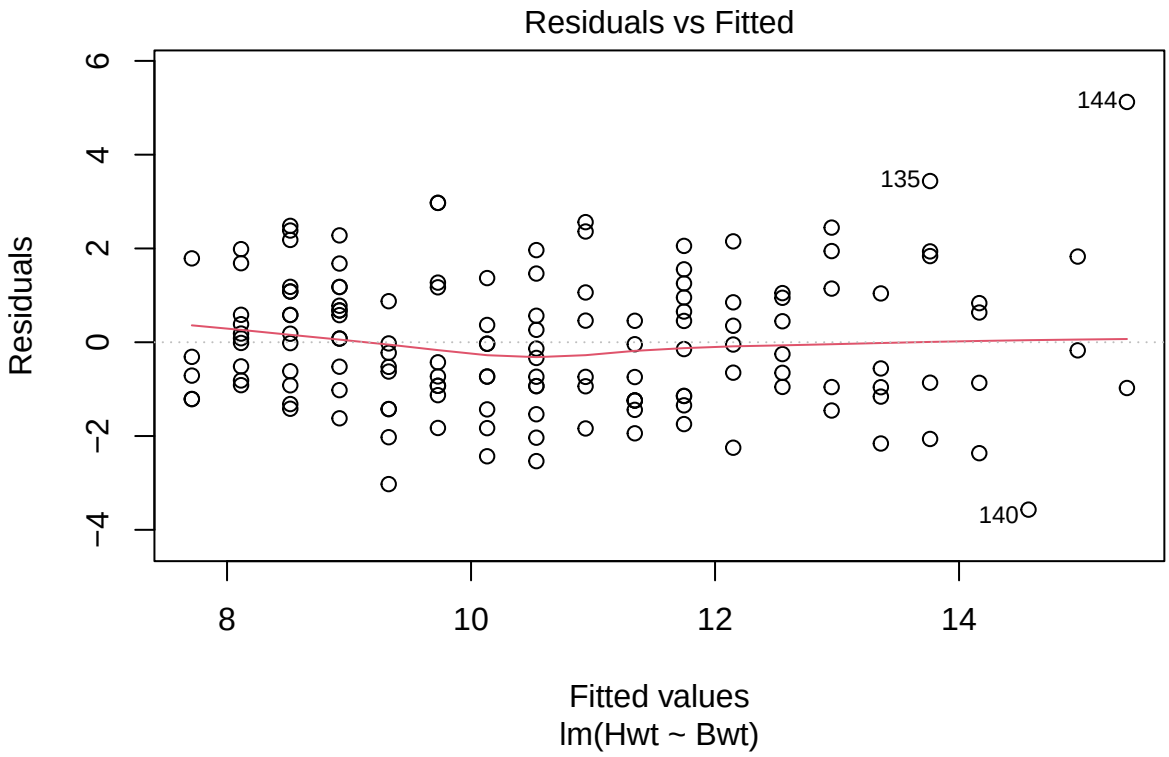
```r
plot(yhat, residuals(model))
```

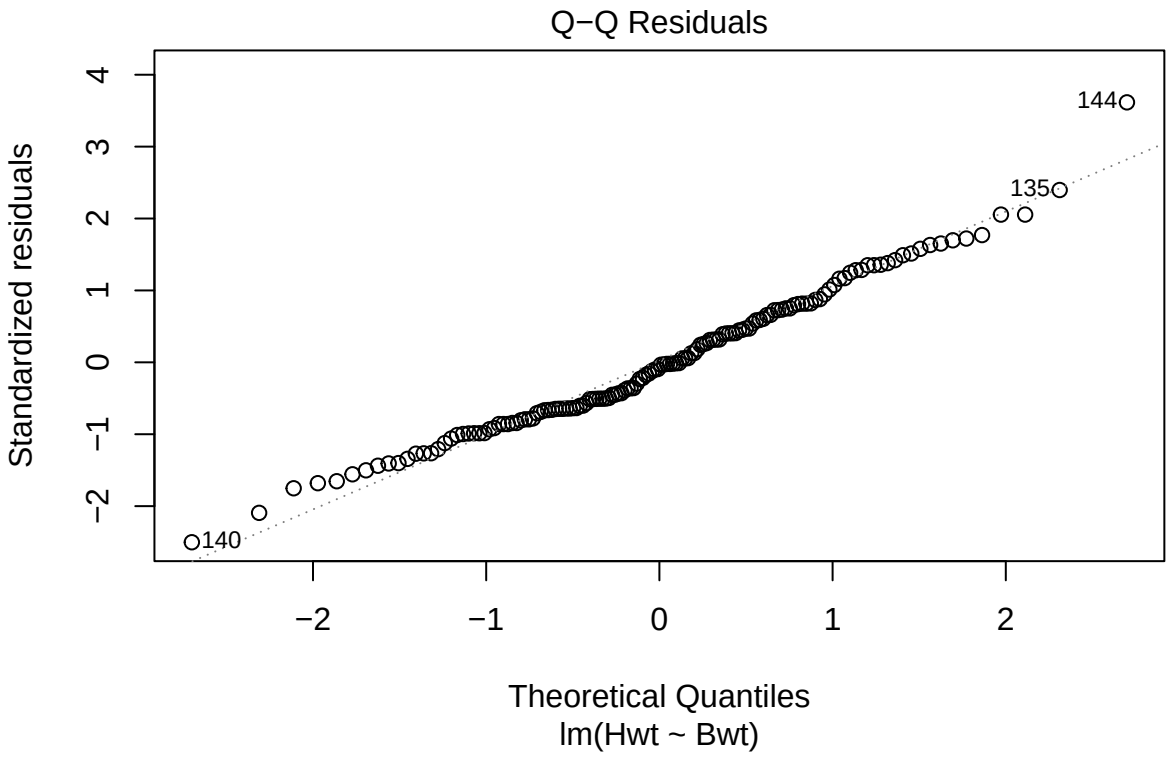


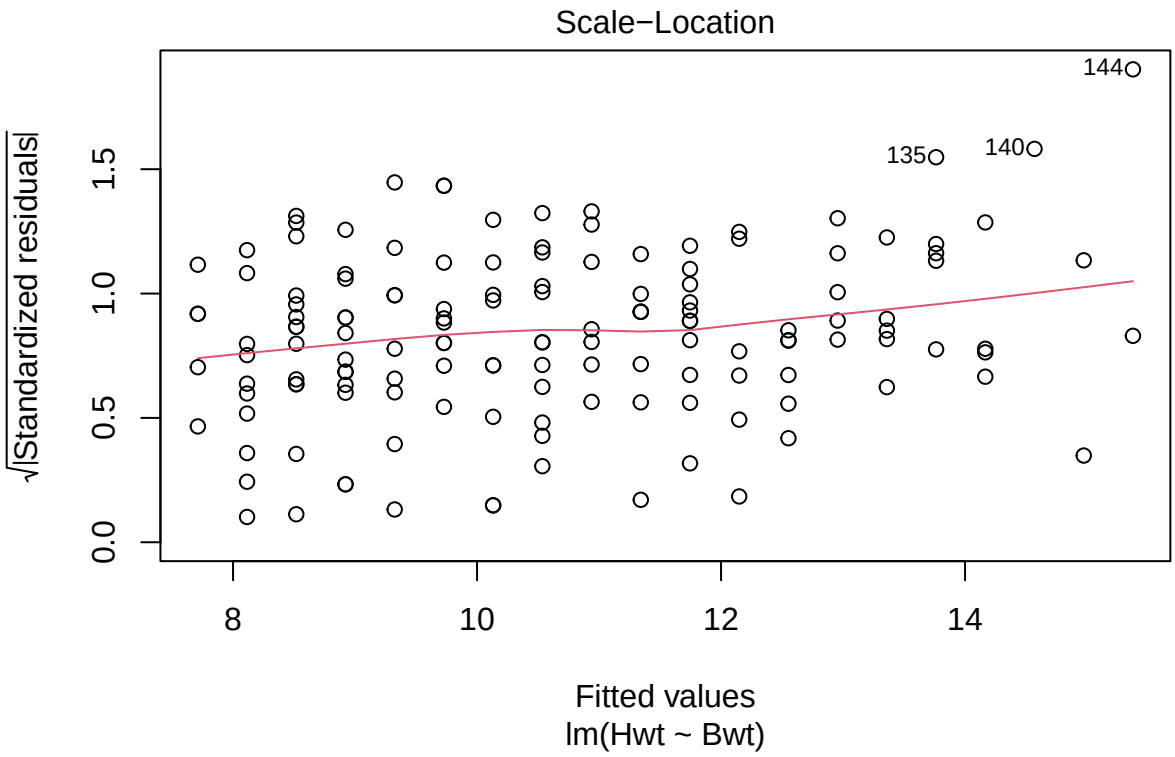

```
qqnorm(residuals(model))
qqline(residuals(model))
```

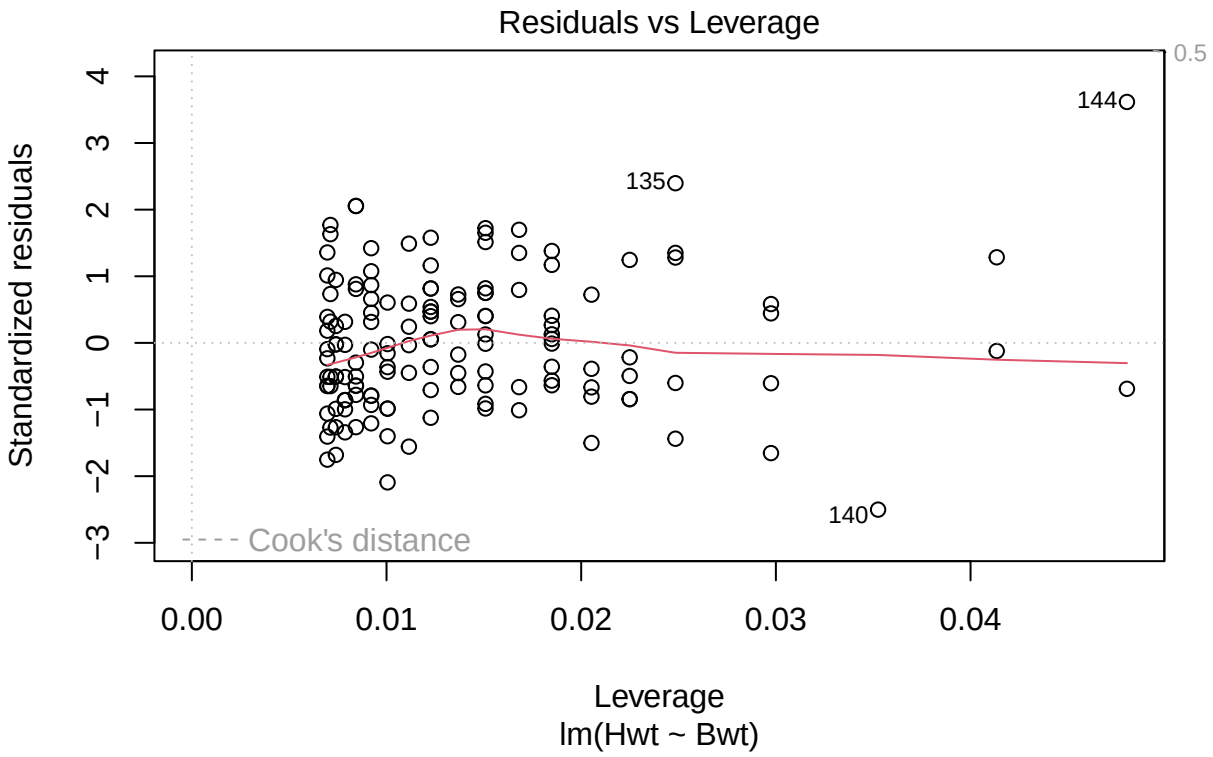

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
0
2
4
−2
−1
0
1
2

```
plot(model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Hwt ~ Bwt)
144
135
140
6
4
2
0
−2
−4
8
10
12
14


Q−Q Residuals
Standardized residuals
−2
−1
0
1
2
3
4
144
135
140
−2
−1
0
1
2
Theoretical Quantiles
lm(Hwt ~ Bwt)


Scale−Location
144
135
140
√|Standardized residuals|
0.0
0.5
1.0
1.5
8
10
12
14
Fitted values
lm(Hwt ~ Bwt)



Model selection

- In R there are pre-built functions which perform automatic model selection, for example, `stepAIC()` in the package MASS

```
stepAIC(object, scope, scale = 0, direction = c("both", "backward", "forward"), trace = 1, keep = NULL,
```

- By default, it uses AIC as stopping criterion (argument `k=2`)
- setting `k=log(n)` we can use BIC. Example:

```
library(MASS)
model <- lm(Hwt ~ Bwt + Sex, data=cats)
model.new <- stepAIC(model)
```

```
## Start:  AIC=111.39
## Hwt ~ Bwt + Sex
##
##        Df Sum of Sq    RSS    AIC
## - Sex   1      0.15 299.53 109.47
## <none>              299.38 111.39
## - Bwt   1    405.88 705.26 232.78
##
## Step:  AIC=109.47
```

```
## Hwt ~ Bwt
##
##        Df Sum of Sq    RSS    AIC
## <none>              299.53 109.47
## - Bwt   1    548.09 847.63 257.26
```

Prediction

- The function `predict()` computes the predicted value of the response for a new observation.

```
predict(object, newdata, se.fit = FALSE, interval = c("none", "confidence", "prediction"), level = 0.95
```

in particular:

`object` must be replaced by the model fit on the data

`newdata` must be a data.frame with the new observation(s)

`se.fit` allows the computation of the standard error

setting `interval="prediction"`, we can compute the prediction interval, with `level` (default `level = 0.95`);

- Example:

```
model <- lm(Hwt ~ Bwt, data=cats)
y.hat <- predict(model, newdata = cats, interval = "prediction")
predict(model) == coef(model)[1] + coef(model)[2]*cats$Bwt
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Logistic model

$logit(\pi) = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + ...$,with $\pi = Pr(Y = 1|X)$

- is a special case of generalized linear model

```
glm(formula, family = gaussian, data, ...)
```

- to fit a logistic model, the argument `family` must be set equal to binomial.

Example:

```
glm(cancer_ever ~ workpollut, family = binomial, data = mydata)
```

```
##
## Call:  glm(formula = cancer_ever ~ workpollut, family = binomial, data = mydata)
##
## Coefficients:
##    (Intercept)  workpollutTRUE
##        -2.6074          0.1048
##
## Degrees of Freedom: 4192 Total (i.e. Null);  4191 Residual
##   (807 Beobachtungen als fehlend gelöscht)
## Null Deviance:       2176
## Residual Deviance: 2175  AIC: 2179
```

As for the linear model...

- we can have an overview on the model through the function `summary()`:

```
summary(glm(cancer_ever ~ workpollut, family = binomial))
```

```
##
## Call:
## glm(formula = cancer_ever ~ workpollut, family = binomial)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -2.60744    0.08758 -29.774   <2e-16 ***
## workpollutTRUE  0.10480    0.11961   0.876    0.381
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2175.8  on 4192  degrees of freedom
## Residual deviance: 2175.0  on 4191  degrees of freedom
##   (807 Beobachtungen als fehlend gelöscht)
## AIC: 2179
##
## Number of Fisher Scoring iterations: 5
```

- we can perform model selection (e.g., based on AIC) through the function `stepAIC()`:

```
stepAIC(glm(mydata$cancer_ever ~ workpollut, family = binomial))
```

- we can use the function `predict()`:

```
predict(glm(cancer_ever ~ workpollut, family = binomial), newdata = mydata)
```