asuR – advanced statistics using R

"The aim of computing is insight, not numbers."

Introducing R

"S is a programming language and environment for all kinds of computing involving data. It has a simple goal: to turn ideas into software, quickly and faithfully." John M. Chambers

- S is a language for "programming with data".
  John Chambers of Bell Labs has been its main developer for more than two decades.
- R is an Open Source system originally written by Ross Ihaka and Robert Gentleman at the University of Auckland in about 1994.
- R is not unlike S (actually they are very similar!)
- R is now developed by a small core team, for all details see:
  `www.r-project.org`.

Introducing R ... continued

Commands to R are expressions or assignments.

expression

```
4/3 * pi * (27)^3                    : [1] 82447.96
```

assignment

```
a <- 27
```

Everything within the R language is an object.
Normally R objects are accessed by their name, which is made up from *letters*, *digits* $0-9$ in non-initial position, or a *period, "."*, that acts like a letter. R is case sensitive.

Every object has a class.

Introducing R ...help & comment

getting help
All functions and data sets in R have a documentation! For information on a function or data set,

`?function-name,`

which is equivalent to

`help(function-name).`

To search all help pages for a specific term

`help.search("term").`

Help pages can also be displayed in a HTML version, therefore

`help.start().`

writing comments
A line starting with # is treated as a comment and not processed.

Introducing R ... your workspace

Objects are normally stored in a workspace.

`ls()` lists all objects currently in your workspace

`rm(object)` removes *object* from your workspace

`save(object, file=path/file)` saves an *object* to a *file*

`load(path/file)` loads an *object* from a *file*

`save.image()` saves your workspace to a file called `.RData` in your working directory. Happens also if you type `q("yes")`.

`getwd()` shows the path of your current working directory

`setwd(path)` allows you to set a new `path` for your current working directory

Introducing R ... additional packages

The functionality of an `R` installation can be extended by packages. Additional packages provide you functions, data sets, and the corresponding documentation. A growing number of packages is available from CRAN

`CRAN.r-project.org`

`library()` shows all packages installed on your system

`library(asuR)` loads an already installed package (here asuR) (asuR is the package that accompanies this course)

`library(help=asuR)` displays all functions, data sets, and vignettes in a package (here asuR)

`data()` shows the data sets of all installed packages

`data(package="asuR")` shows data set(s) from a package, here asuR

`data(pea)` loads the data set "pea" to your workspace (therefore the package asuR has to be loaded)

`vignette()` shows vignettes from all installed packages

`vignette(package="asuR")` shows vignette(s) from a package (here asuR)

`vignette("coursenotes")` opens the pdf file with the course notes

`c()` creates a vector of the specified elements (`c` for concatenate)

```
> genus <- c("Daphnia", "Boletus", "Hippopotamus", "Salmo", "Linaria",
+     "Ixodes", "Apis")
> species <- c("magna", "edulis", "amphibius", "trutta", "alpina",
+     "ricinus", "mellifera")
> weight <- c(0.001, 100, 3200000, 1000, 2.56, 0.001, 0.01)
> legs <- as.integer(c(0, 0, 4, 0, 0, 8, 6))
> animal <- c(TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, TRUE)
```

`length()` returns the length of a vector

`paste()` takes two vectors and concatenates them as characters

```
name <- paste(genus, species)
```

`seq()` to generate sequences of numbers

```
seq(from=4, to=7)                    : [1] 4 5 6 7
4:7                                  # short form of the previous
seq(from=4, to=7, by=0.5)            : [1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0
```

`rep()` to replicate elements of a vector

```
rep(c(2,4,6), times=3)               : [1] 2 4 6 2 4 6 2 4 6
rep(c(2,4,6), each=3)                : [1] 2 2 2 4 4 4 6 6 6
rep(c(2,4,6), times=3, each=2)       : [1] 2 2 4 4 6 6 2 2 4 4 6 6 2 2 4 4 6 6
```

`sample()` takes a random sample of a vector

```
sample(genus)                        : [1] "Boletus"      "Daphnia"  "Ixodes"  "Salmo"
                                       [5] "Hippopotamus" "Linaria"  "Apis"
```

A special type of a vector that usually stores categorical variables.

`factor()` makes a factor out of a vector

```
> kingdom <- factor(c("animal", "fungi", "animal", "animal", "plant",
+     "animal", "animal"))
```

`levels()` provides a character vector with the levels of a factor

Internally a factor is stored as a set of codes and an attribute giving the corresponding levels.

```
unclass(kingdom)                    : [1] 1 2 1 1 3 1 1
                                    : attr(,"levels")
                                    : [1] "animal" "fungi"  "plant"
```

If you take a subset of a factor always all levels of a factor are included, whether they are in the subset or not.

```
levels(kingdom[1:2])                : [1] "animal" "fungi"  "plant"
```

To exclude unused levels from the subset we can use

```
levels(kingdom[1:2, drop=TRUE])     : [1] "animal" "fungi"
```

Introducing R . . . data.frame

Used to store data. It is a list of variables, all of the same length, possibly of different types.

`data.frame()` a function to generate data frames

```
> bio <- data.frame(name = I(paste(genus, species)), weight_g = weight,
+     leg_no = legs, animal = animal, kingdom = kingdom)
```

`names()` displays the names of the variables in a data frame

`row.names()` displays the row names

`str()` a useful summary of the structure of a data frame

`summary()` provides a summary of all variables in a data frame

`attach()` makes the variables of a data frame accessible by their name

`detach()` the inverse

`write.table()` to write a data frame to a text file

```
> write.table(bio, file = "~/temp/bio.txt", row.names = FALSE,
+     sep = "\t")
```

`read.table()` to read a data frame from a text file

```
> bio.new <- read.table(file = "~/temp/bio.txt", header = TRUE,
+     sep = "\t", row.names = "name", colClasses = c("character",
+         "numeric", "integer", "logical", "factor"))
```


Introducing R . . . matrix & array

A matrix has all its arguments of the same type and always two dimensions. An array is like a matrix but with a flexible number of dimensions.

`matrix()` a function to create a matrix from a vector

`rbind()` takes vectors and binds them as rows together

`cbind()` takes vectors and binds them as columns together

```
> mat <- matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
> mat1 <- matrix(c(1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 12), nrow = 3,
+     ncol = 4)
> mat2 <- rbind(c(1:4), c(5:8), c(9:12))
> mat3 <- cbind(c(1, 5, 9), c(2, 6, 10), c(3, 7, 11), c(4, 8, 12))
```

`dim()` returns the dimensions

`dimnames()` to give a name to the columns and rows

```
> dimnames(mat) <- list(c("first row", "second row", "last row"),
+     paste("c_", 1:4, sep = ""))
```

Introducing R . . . list

A list is a collection of *components* that can be from different classes and of different length.

```
> organisms <- list(animals = list(genera = c("Daphnia", "Hippopotamus",
+     "Salmo", "Ixodes", "Apis"), publications = 110), plants = list(genera = "Linaria",
+     publications = 50), fungi = "Boletus")
```

`$` to extract components by their name

```
organisms$animals$genera          : [1] "Daphnia" "Hippopotamus" "Salmo" "Ixodes" "Apis"
                                  # a character vector of length five
```

`[[` to extract a component by its position

```
organisms[[1]][[1]]               : [1] "Daphnia" "Hippopotamus" "Salmo" "Ixodes" "Apis"
                                  # a character vector of length five
```

`[` to extract a sub-vector

```
organisms[[1]][1]                 : $genera
                                  : [1] "Daphnia" "Hippopotamus" "Salmo" "Ixodes" "Apis"
                                  # a list of length one
```


Introducing R . . . function

```
function-name  <- function(argument1, argument2, ...){

      function.body

}
```

An example for calculating the t value according to

$$\frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{n_1+n_2-2} \qquad \text{with,} \quad S_p^2 = \frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}$$

```
> my.t <- function(mean1, mean2, s1, s2, n1, n2, mu.diff = 0) {
+     s.pooled <- ((n1 - 1) * s1^2 + (n2 - 1) * s2^2)/(n1 + n2 -
+         2)
+     t <- ((mean1 - mean2) - (mu.diff))/(sqrt(s.pooled * (1/n1 +
+         1/n2)))
+     cat(round(t, 2))
+     cat(paste(": compare to a t dist. with", n1 + n2 - 2, "df \n"))
+ }
```

The function can then be called with, e.g.,

```
my.t(mean1=24, mean2=18, s1=3, s2=4, n1=34, n2=50)              : 7.43; compare to a t dist. with 82 di
my.t(mean1=24, mean2=18, s1=3, s2=4, n1=34, n2=50, mu.diff=4)   : 2.48; compare to a t dist. with 82 di
```

exercise 1 Put `n=10` and compare `1:n-1` and `1:(n-1)`.

exercise 2 How can you produce a vector like,

a) *1 1 2 2 3 3*
b) *1 2 1 2 1 2*
c) *1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3*

exercise 3 How can you produce a character vector containing,

a) *"trt:1" "trt:1" "trt:2" "trt:2" "trt:3" "trt:3"*
b) *"ind:1" "ind:2" "ind:1" "ind:2" "ind:1" "ind:2"*

exercise 4 Make a data frame with a variable called "treatment" and "individual" and use therefore the character vectors of the previous exercise.

a) Check the dimensions of the data frame you created.
b) What is the class of the two variables within the data frame?
c) Add to the existing data frame a column with values from 2 to 7.
d) Save the data frame to file called "experiment.Rdata", remove the data frame from your workspace, and reload it from the file.
e) Save the data frame to a tabulator separated text file called "experiment.txt", remove the data frame from your workspace, and read it in again.

indexing

Many data manipulations in `R` rely on indexing. Indexing is used to address a subsets of vectors, matrices, or data frame. The general form for using an index is

`object[`*`index-vector`*`]` .

Indexing is used to select a subset of an object,

`new.object <- object[`*`index-vector`*`]` ,

to replace a subset of an object,

`object[`*`index-vector`*`] <- new.element` ,

or to sort an object (see below for an example).

On the following pages we will see what types the ***index-vector*** can take (for vectors and data frames & matrices respectively).

The examples below always apply to the following vector object

```
> x <- c(11, 44, 33, NA, 22)
```

logical vector

- must be of the same length as the vector
- values corresponding to `TRUE` are included; corresponding to `FALSE` are omitted (`NA` inserts an `NA` at the corresponding position)

```
x[c(TRUE, FALSE, FALSE, FALSE, TRUE)]   : [1] 11 22
x[!is.na(x)]                            : [1] 11 44 33 22
x[x>=33]                                : [1] 44 33 NA
x[(x==33 | x==44) & !is.na(x)]          : [1] 44 33
```

vector of positive integers (factors)

- the values of the integers must be smaller or equal to the length of the vector
- the corresponding elements are selected and concatenated in the order they were selected
- for factors as index vector (works like: x[unclass(factor)])

```
x[c(1,2,5)]                             : [1] 11 44 22
x[1:3]                                  : [1] 11 44 33
x[order(x)]                             : [1] 11 22 33 44 NA
```


vector of negative integers

- the absolute values of the integers must be smaller or equal to the length of the vector
- the corresponding elements are excluded

```
x[c(-1,-2,-5)]                          : [1] 33 NA
```

empty

- all components are selected
- identical to 1:length(object)

```
x[]                                     : [1] 11 44 33 NA 22
```

vector of character strings

- only applies if object has names
- the corresponding elements are selected and concatenated in the order they were selected

```
names(x) <- c("first", "largest", "middle", "non.available", "second")
x[c("largest", "first")]
                                        : largest   first
                                        :      44      11
```

indexing ... data frames & matrices

Data frames and matrices can be indexed by giving two indices ([*rows,columns*]).

```
bio[bio$animal, ]                    # all rows where animal is TRUE
bio[bio$weight_g>1, "name"]          # name of all organisms heavier >1g
```

Columns in a data frame are often selected with the $ operator

```
bio$names                            # equivalent to bio[,"names"]
```

matrix

An array (and therefore also a matrix) can be indexed by a $m \times k$ matrix. Each of the m rows of this matrix is used to select one element.

```
> mat <- matrix(1:9, ncol=3)         :      [,1] [,2] [,3]
                                     : [1,]    1    4    7
                                     : [2,]    2    5    8
                                     : [3,]    3    6    9
select <- rbind(c(2,1),c(3,1),c(3,2))
mat[select]                          : [1] 2 3 6
```

If you extract elements from a data frame or a matrix, the result is coerced to the lowest possible dimension. This default behavior can be changed by adding `drop=FALSE`,

```
bio[,"kingdom"]                      # returns a vector
bio[,"kingdom", drop=FALSE]          # returns a data frame
```

indexing ... practicals II

exercise 1 Create the following matrix,

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

a) Extract the third column.
b) Extract the diagonal elements.
c) Extract the anti diagonal elements.

exercise 2 Create a vector with all integers from 1 to 1000 and replace all even numbers by their inverse.

exercise 3 Load the data frame "swiss" from the package datasets.

a) Read the associated documentation.
b) Sort the data frame by the variable "Agriculture".
c) Add a factor called "religion" to the data frame which has the level "catholic", if "Catholic" is larger or equal to 50, and "protestant" otherwise.
d) Sort the data frame by the variable "religion" and "Agriculture"
e) Sort the data frame by the provinces (row.names).
f) Put the rows of the data frame in a random order
g) Remove the column "Education"

graphics

> Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
>
> Edward R Tufte

The graphics system of `R` is very powerful. You can get a sample gallery with

```
demo(graphics).
```

A very large gallery with many complex examples is at
`http://addictedtor.free.fr/graphiques/`.

Rank	Aspect
1	Position along a common scale
2	Position on identical but nonaligned scales
3	Length
4	Angle
5	Slope
6	Area
7	Volume
8	Color hue (poor ordering, good discrimination)

(Cleveland, William S. & McGill, Robert 1985)

general
- make data stand out, avoid superfluity, decrease the ink to information ratio
- show data (e.g. `rug()`), if too numerous, consider showing a random sample
- induce the viewer to think about the substance rather than about methodology
- directly show what is in the focus of your study (e.g. show the difference between treatments and the corresponding confidence interval instead of only the mean (and confidence intervals) for each treatment)

scale
- inclusion of zero on a axis has to be thought about carefully

error bars can show:

- sample standard deviation of the data
- estimate of the standard deviation of a statistic (often called *standard error*)
- confidence intervals for a statistical quantity

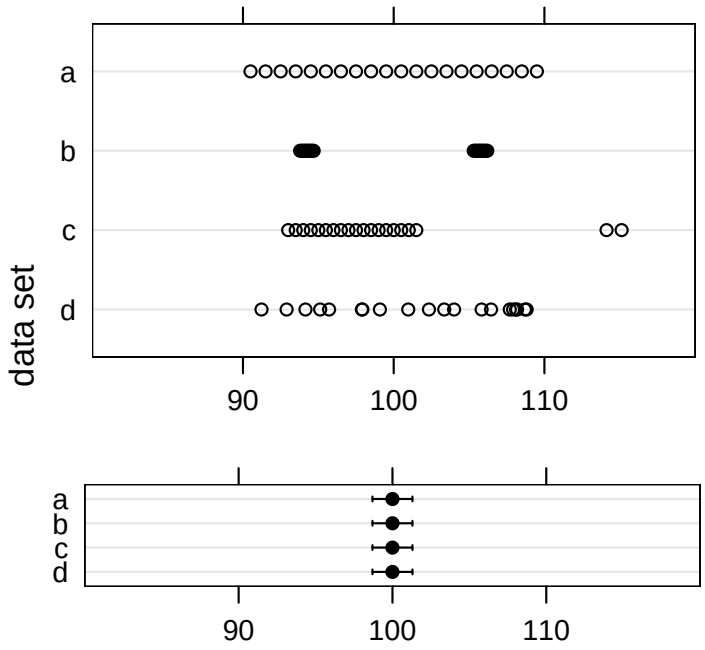


figure All four data sets have the same number of observations, the same mean and the same standard deviation. Mean and standard deviation of the mean (standard error) do often a poor job in conveying the distribution of the data.

graphics . . . technically

devices The output is always directed to a particular device that dictates the output format.

graphics functions The functions that are available can be divided into two categories,

highlevel functions create a new figure,
lowlevel functions add something to an already existing figure.

Note: In R there are two distinct graphics systems, the traditional and the grid system. All the following explanations hold for the traditional graphics system only. Many details of both systems are explained in some detail by Murrell (2005).

graphics . . . devices

hardcopy devices to create a physical picture, e.g., *pdf()*, *png()*, *jpeg()*, *postscript()*

window devices to create a graphic on a window system, e.g., *x11()*, *win.graph()*

You can see the features compiled into your build of R with

```
capabilities()
```

To produce e.g. histogram in pdf format

```
pdf(file="~/Desktop/myhistogram.pdf", width=4, height=3)
hist(rnorm(100))
dev.off()
```

Have a look at the documentation of a device, e.g., *?pdf*, to see possible customisations.

graphics ...highlevel functions

`plot()` scatterplots

`pairs()` scatterplot matrix

`hist()` histogram

`boxplot()` boxplot

`qqnorm()` normal quantile-quantile plot

`qqplot()` general quantile-quantile plot

`barplot()` barcharts

`dotchart()` dotplot (continuous vs. categorical)

`stripchart()` stripplots (one-dimentional scatterplot)

an example:

```
> y <- rnorm(100)
> qqnorm(y)
> qqnorm(y, xlab = "theoretical quantiles", ylab = "sample quantiles",
+     main = "normal quantile--quantile plot", sub = "(because I like it lower case)",
+     xlim = c(-3, 3), ylim = c(-3, 3))
```


graphics ...lowlevel functions

`abline()` adding a line definded by the form $a + bx$ (h=, v=, for horizontal and vertical lines)

`lines()` adding lines

`qqline()` adding a line to a normal quantile plot through the first and third quantile

`points()` adding points

`box()` adding box

`text()` adding text to the plot

`mtext()` adding text into the margin

`title()` adding a title

`legend()` adding a legend

`axis()` adding axis

an example, continued:

```
> qqline(y)
```

graphics . . . customizing

You can customize every aspect of the display using graphical parameters.

settings within par() affect all subsequent graphical output
settings within a highlevel function affect the whole plot
settings within a lowlevel function affect only this element of a plot

an example, revisited:

```
> par(fg = "blue")
> y <- rnorm(100)
> qqnorm(y, col = "red")
> qqline(y, col = "green")
```


graphics . . . settings within par() only

`mfrow`	numbers of figures on a page. `mfrow=c(2,3)` creates a 2-by-3 layout, filled row-by-row
`mfcol`	numbers of figures on a page. `mcol=c(2,3)` creates a x-by-y layout, filled column-by-column
`pty`	aspect ratio of the plot region. `pty="s"` for a squared plotting region
`mai`	size of figure margin in inches. `mai=c(4,3,2,1)` for a plot with 4,3,2,1 inches of margin at the c(bottom, left, top, right), respectively.
`mar`	like `mai` but in lines of text
`omi`	size of outer margin in inches (if several figures are printed on one page)
`oma`	size of outer margin in lines of text (if several figures are printed on one page)

graphics ... settings within par(), high-, or lowlevel functions

`bg`	background color
`fg`	foreground color (axis, boxes, etc.; called from within `par()` also sets `col`)
`col`	color of lines, symbols, etc.
`col.axis`	color of axis annotation
`col.lab`	color of axis labels
`col.main`	color of main title
`pch`	data symbol type e.g., 1=circel, 2=triangle, 3=plus sign, 4=times sign, 5=diamond, etc.
`cex`	multiplier for the size of text
`cex.axis`	multiplier for the size of axis tick labels
`cex.lab`	multiplier for the axis label
`cex.main`	multiplier for the main title
`lty`	line type e.g., 0=blank, 1=solid, 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash, etc.
`lwd`	line width

graphics ... color

There are three different ways to define a color

name you can see what color names are understood by typing `colors()`, e.g., `"blue"`.
function `rgb()` with the inensities of red, green and blue, e.g., `rgb(0,0,1)`.
hexadeximal string of the form `"#RRGGBB"`, e.g., `"#0000FF"`.

an example, revisited again:

```
> par(col = "blue")
> y <- rnorm(100)
> qqnorm(y, col = rgb(1, 0, 0))
> qqline(y, col = "#00FF00")
```

exercise 1 Load the data set "trees" from the package **datasets** to your workspace.

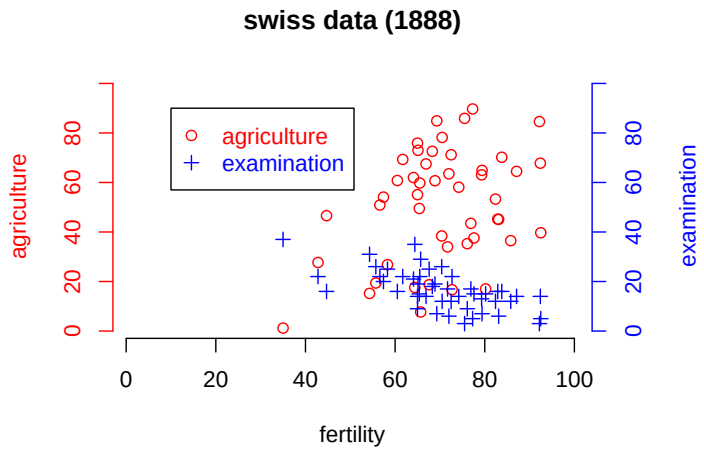
- **a)** Produce a scatterplot of all variables ("Volume", "Girth", and "Height") against each other.
- **b)** Produce a pdf-file (9cm x 13cm) with a scatterplot of "Volume" against "Girth".
- **c)** Customize the labels of the x– and y–axis, the main title, the color etc. to produce a nice and informative plot.

exercise 2 Load the data set "swiss" from the package **datasets** to your workspace.

- **a)** Produce a histogram of the variable "Fertility".
- **b)** Produce side by side a boxplot of "Fertility" for "catholic" and "protestant" (see the previous practicals).

exercise 3 Reconstruct the following scatterplot with the variables from data set "swiss".

- **a)** How can this graph be improved?



summarising data

statistical parameters

```
summary(pea$length)        :    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
                           :   56.00   58.00   60.50   61.94   65.00   76.00
```

see also:

```
mean(), median(), quantile(), min(), max(), var(), fivenum()
```

graphs

boxplots

histogram

density estimation

quantile-quantile plots

summarising data ... data characteristics

shape symmetric, left-skewed or right-skewed, and as uni-modal, bi-modal or multi-modal

location (also called measure of central tendency) Measures of location are the mean, median, mode, and mid-range.

spread Measured by the variance, the standard deviation, the interquartile range, and the range.

outliers Don't just toss out outliers, as they may be the most valuable members of a data set.

clustering Clustering means that data bunches up around certain values. It shows up clearly in a dotplot.

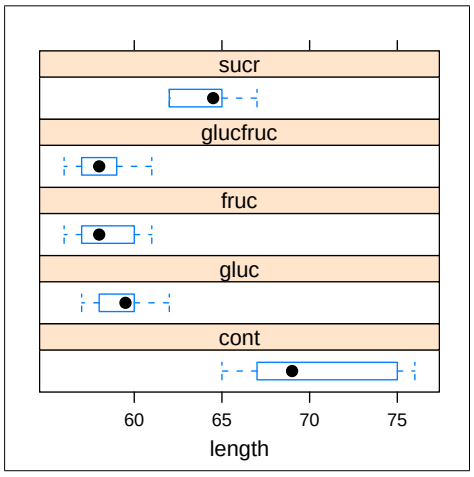
```
stripchart(pea$length~pea$trt)
hist(pea$length)
rug(pea$length)          # the function rug can be very useful to visualize
                         # clustering togeter with e.g., a histogram
```

granularity if only certain discrete values are used. Discrete data shows always some granularity, but also continous data can show granularity, e.g., after rounding. Granularity can also be detected in a dotplot.

The boxplot was invented by TUKEY. He recommends the following:

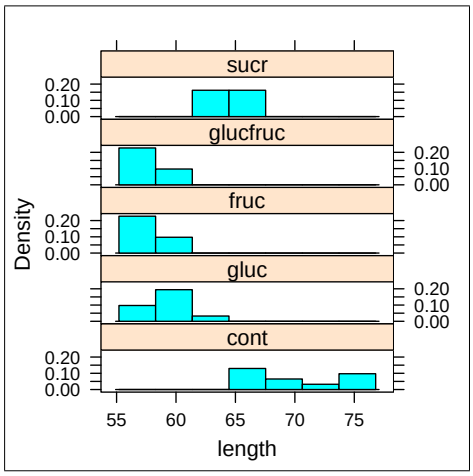
> Drawing the box:
> Find the median. Then find the median of the data values whose ranks are *less than or equal to* the rank of the median. This will be a data value or it will be half way between two data values.
> Drawing the whiskers:
> The maximum length of each whisker is 1.5 times the interquartile range (IQR). To draw the whisker above the 3rd quartile, draw it to the largest data value that is less than or equal to the value that is 1.5 IQRs above the 3rd quartile. Any data value larger than that should be marked as an outlier.

"Why 1.5 times the interquartile range?"
TUKEY anwered: "because 1 is too small and 2 is too large".



with R

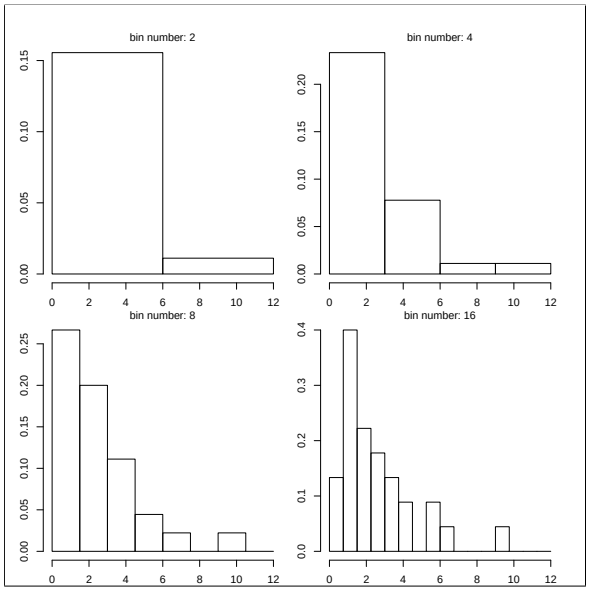
```
boxplot(your.sample)
bwplot(~length|trt, layout=c(1,5), data=pea)      # using: library(lattice)
```



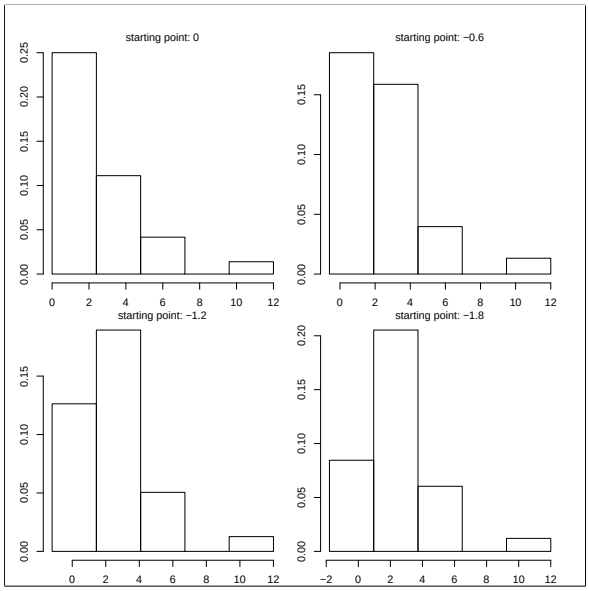
with R

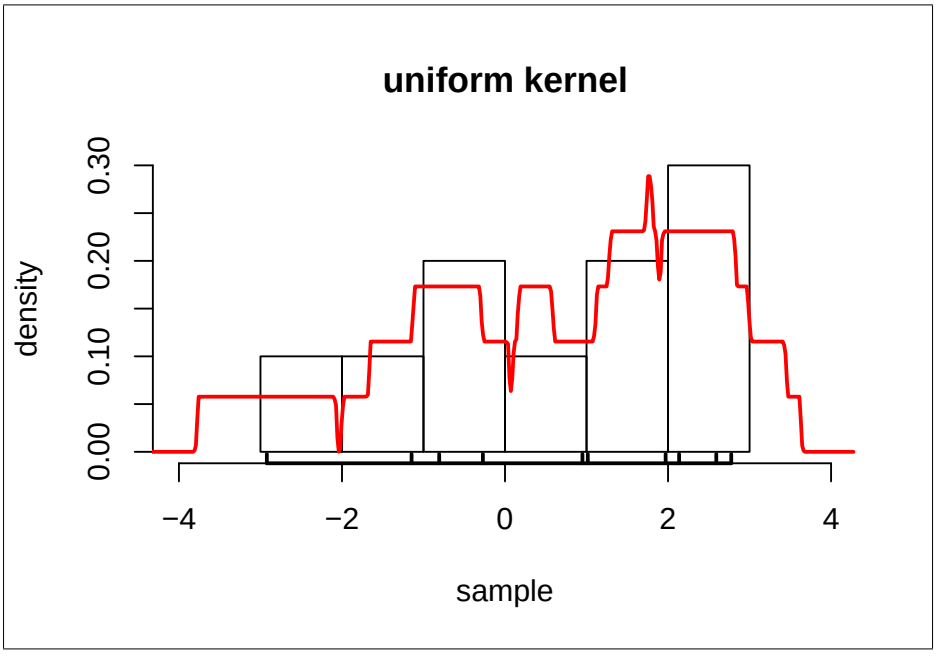
```
hist(your.sample, freq=FALSE)
truehist(your.sample)                                          # from package MASS

histogram(~length|trt, type="density", layout=c(1,5), data=pea) # library(lattice)
                                                               # used for the graph
```

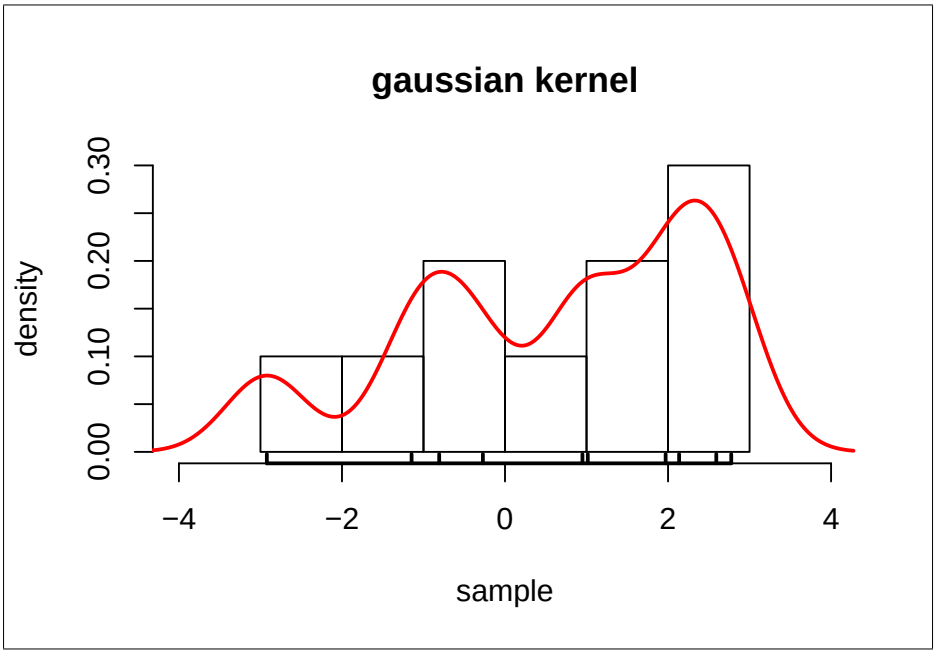

histograms depend on the bin number



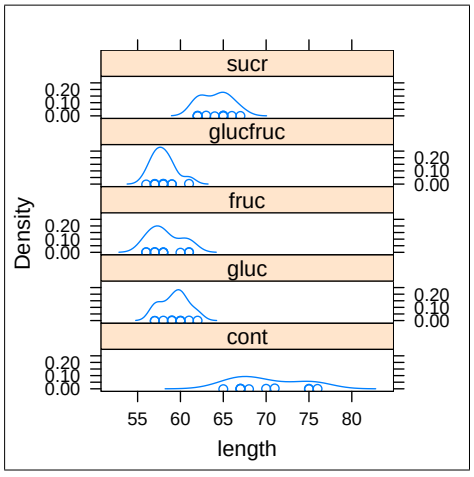
histograms depend on the starting point







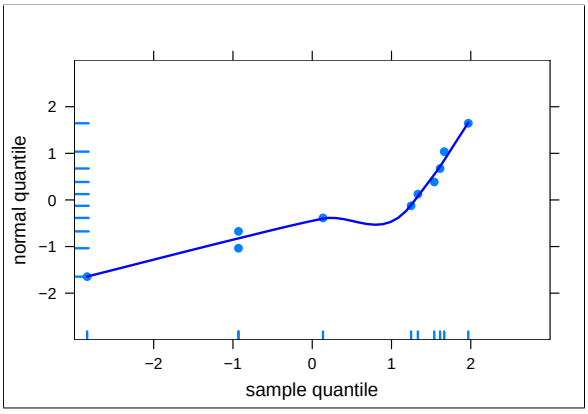
density plots are indepentent of the starting point



with R

```
hist(your.sample, freq=FALSE)                      # adjust ylim if needed
lines(density(your.sample))

densityplot(~length|trt, layout=c(1,5), data=pea) # library(lattice)
                                                   # was used for the graph you see
```

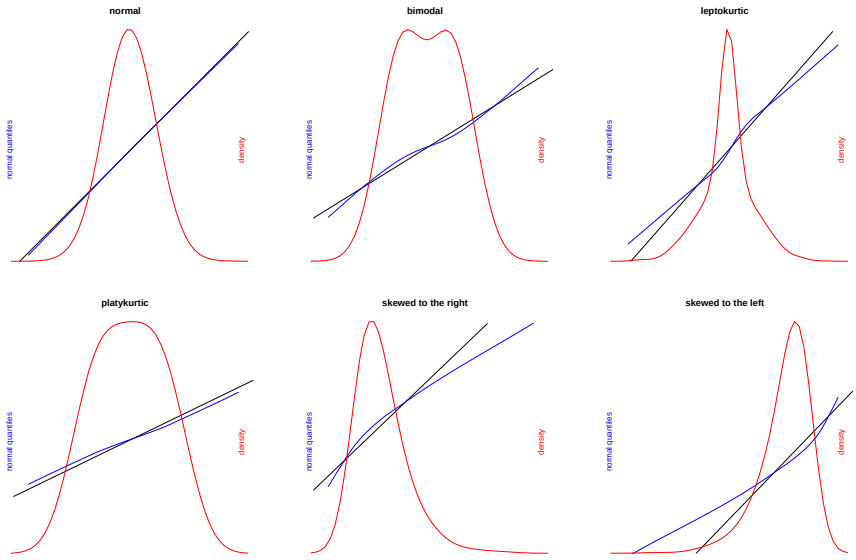
normal quantile-quantile plot



with R

```
qqnorm(your.sample)
qqline(your.sample)
```


normal quantile quantile plots for different distributions



densities and distributions

R has build–in functions starting with

`d`	for *density* with the argument `x` (quantile)
`p`	for *probability* with the argument `q` (quantile)
`q`	for *quantile* with the argument `p` (probability)
`r`	for *random* with the argument `n` (number of observations)

and ending with the name of a distribution

`beta`	Beta	`lnorm`	Lognormal
`binom`	Binominal	`logis`	Logistic
`cauchy`	Cauchy	`nbinom`	Negative binomial
`chisq`	χ^2	`norm`	Normal, Gaussian
`exp`	Exponential	`pois`	Poisson
`f`	Fisher's F	`t`	Student's t
`gamma`	Gamma	`unif`	Uniform
`geom`	Geometric	`weibull`	Weibull
`hyper`	Hypergeometric		

densities and distributions . . . practicals IV

exercise 1 Generate a sample of size 100 from a $t-$distribution with 4 degrees of freedom.

- **a)** Produce a normal QQ-plot of this sample and assess how straight the produced plot is by adding a straight line. Interprete!
- **b)** Produce a histogram of the sample.

exercise 2 Generate a sample of size 100 from a normal distribution ($\mu = 0, \sigma = 1$).

- **a)** Produce a normal QQ-plot of this sample.
- **b)** Produce a histogram of the sample.

exercise 3 Produce a plot with all four plots side by side (pdf-file, 20cm x 20cm).

- **a)** Adjust in all four plots the range displayed on the x- and y-axis.
- **b)** Compare the sample from the normal distribution with the sample from the $t-$distribution.

exercise 4 Load the package asuR. Generate a sample of random numbers, e.g., `x <- rchisq(20, df=2)`. Use the function `norm.test(x)` and try to identify your data sample among 8 samples from a normal distribution of the same size.

- **a)** How large must the sample be, that you can clearly identify it?
- **b)** Load the data set swiss again. Can you distinguish the variable "Fertilty" from a sample taken from a normal distribution? Interpret!
- **c)** Can you distinguish the variable "Catholic" from a sample taken from a normal distribution? Interpret!

classical statistics

`t.test()` student's t-test for comparing two sample means; (paired=logical)

`pairwise.t.test()` for pairwise comparisons among means with correction for multiple testing

`prop.test()` test for given or equal proportions

`pairwise.prop.test()` pairwise comparisons for pairwise proportions with correction for multiple testing

`chisq.test()` Pearson's Chi-squared test for count data

`fisher.test()` Fisher's exact test for count data

`binom.test()` an exact binomial test

`mantelhaen.test()` Cochran-Mantel-Haenszel test for three-dimensional contingency tables

`mcnemar.test()` McNemar's Chi-squared test on two-dimensional contingency tables

`var.test()` an $F-$test to compare two variances

`cor.test()` for testing an association between paired samples

`bartlett.test()` for testing the homogeneity of variances

`fligner.test()` for testing the homogeneity of variances

classical statistics ... continued

Non-parametric tests:

`friedman.test()` an alternative to analysis of variance based on ranks

`kruskal.test()` one-way analysis of variance with ranks

`ks.test()` Kolmogorov-Smirnov test; a non-parametric test for comparing both! the location and scale of two samples

`shapiro.test()` Shapiro-Wilk normality test

`wilcox.test()` Wilcoxon signed rank test; an alternative for the paired $t-$test the same function is also used to perform a "Mann-Whitney"-test; to test if the median of two samples differ

classical statistics ... t-test

The $t-$test can be used to test if two samples were taken from a population with the same mean. To use the $t-$test this two populations need to be normally distributed and have an equal variance.

```
data(pea)
glucose <- pea[pea$trt=="gluc","length"]
fructose <- pea[pea$trt=="fruc","length"]
t.test(glucose, fructose)
```

```
my.t(mean1=mean(glucose), mean2=mean(fructose),              # the function my.t was defined p.12
+    s1=sqrt(var(glucose)), s2=sqrt(var(fructose)),
+    n1=10, n2=10)
pt(1.4, df=18)                                               # the probability for a larger value
1-pt(1.4, df=18)                                             # the one-sided $p-$value
(1-pt(1.4, df=18))*2                                         # the two-sided $p-$value
```

linear models in R

```
fitted-model  <- model-fitting-function (formula , data-frame)
```

A list of model fitting functions,

`aov()` for analysis of variance

`lm()` for regressions and analysis of covariance

`glm()` for generalized linear models

`lmer()` for linear mixed effect models (library lme4)

`lme()` for linear mixed effect models (library nlme, "older", but much better documented, see book by (Pinheiro & Bates 2000))

linear models in R ... continued

A fitted model object contains the information from the analysis. Several functions are available to extract this information, to inspect the distributional assumptions of the fitting process, and to look for models that better fit to the data.

`plot()` for diagnostic plots

`summary()` for a summary of the analysis

`summary.lm()` for a

`anova()` computes an analysis of variance table (see below)

`coef()` for the coefficients

`resid()` for the residuals

`fitted()` for the fitted values

`predict()` to predict new means for new data

`deviance()` for the residual sum of squares

`df.residual()` for the residual degrees of freedom

`step()` stepwise ("backward" or "forward") model selection

To function `anova()` can additionally be used to compare fitted model objects.

model formula

A model formula is of the form

```
y  ~ model.
```

It describes how the response `y` is modelled by a linear predictor specified by the `model`. Operators used to specify a model

`+` include additive terms
`-` remove a term
`*` crossing of factors
`^a` crossing to the ath degree
`:` interaction
`%in%` and `/` "nested in"
`|` indicates grouping
`I()` operators inside are used in an arithmetic sense

By default a formulae produces a model matrix with an intercept. If you want to remove it use `-1`.

model formula ... continued

Arithmetic expressions can be used directly within a formula, e.g, `log(y) ~ x`. If you use operators that are already used symbolically in the model formulae (see above) you must tell `R` explicitly that you want to use them in the arithmetic sense (with function `I()`, meaning: "inhibit the interpretation").

```
y ~ I(a-1)                          # subtracts 1 from a
y ~ a - 1                           # removes the intercept
y ~ a + I(b+c)                      # two terms: a and (b+c)
y ~ a + b + c                       # three terms: a, b, and c
```

Model formulae are often long and cumbersome to handle. The operators `*` and `^` are in this situations useful.

```
(a+b+c)^2     and     a*b*c - a:b:c     # are interpreted as: a + b +c + a:b + a:c + b:c
```

analysis of variance

```
model <- aov(length ~ trt, data=pea)
```

BEFORE you interpret a model, inspect it!

```
plot(model)
```

You can also inspect your model with your own graphs!

```
res <- resid(model)
qqnorm(res)...etc!
norm(model)                          # from library(asuR)
inspect(model)                       # from library(asuR)
```


contrasts & comparisons

For factors with more than two levels, comparisons among different levels are usually of interest (also comparisons among means of different levels).

planned vs. unplanned (*a-priori* vs. *a-posteriory*) Planned comparisons are chosen *independently* of the results of the experiment and *before* the experiment has been carried out. Unplanned comparisons suggest themselves *as a result* of an experiment and they include all possible pairs ($\#levels(\#levels-1)/2$) of comparisons. Therefore they are also called *multiple* comparisons. To test whether comparisons are significant we need to distinguish these two cases!

orthogonality The number of orthogonal comparisons is restricted to #levels−1 (two contrasts are orthogonal if the product of their coefficients sum to zero)

```
> data(pea)
> boxplot(x = split(x = pea$length, f = pea$trt), xlab = "levels of the factor \"trt\"",
+     ylab = "length of pea sections")
> m0 <- lm(length ~ trt, data = pea)
> coefficients(m0)
> options("contrasts")
```

contrasts & comparisons . . . setting contrasts manually

To set the contrasts manually you specify a vector with *coefficients for a linear comparisons*. To compare the addition of sugar we can compare the control to the mean of the four sugar treatments. Therefore we subtract the mean pea length of the four sugar treatments from the length of the control.

```
> contr1 <- rbind("control-sugar" = c(1, -1/4, -1/4, -1/4, -1/4))
> m1 <- lm(length ~ trt, data = pea, contrasts = list(trt = mycontr(contr = contr1)))
> summary(m1)
```

To test whether a mixture of sugars is different from pure sugars we calculate a second contrast.

```
> contr2 <- rbind("control-sugar" = c(1, -1/4, -1/4, -1/4, -1/4),
+     "pure-mixed" = c(0, 1/3, 1/3, -1, 1/3))
> m2 <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(contr = contr2)))
```

We can control whether it is orthogonal to the first by checking whether the product of the coefficients add up to zero.
$(1 \times 0) + (-1/4 \times 1/3) + (-1/4 \times 1/3) + (-1/4 \times -1) + (-1/4 \times 1/3) = 0$

contrasts & comparisons . . . setting contrasts manually . . . continued

In this particular example the investigator was interested in two other contrast. One is the difference between monosaccharides and disaccharides and the other the difference between glucose and fructose.

```
> contr3 <- rbind("control-sugar" = c(1, -1/4, -1/4, -1/4, -1/4),
+     "pure-mixed" = c(0, 1/3, 1/3, 1, 1/3), "monosaccharides-disaccharides" = c(0,
+         1/2, 1/2, 0, -1))
> m3 <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(contr = contr3)))
> contr4 <- rbind("control-sugar" = c(1, -1/4, -1/4, -1/4, -1/4),
+     "mixed-pure" = c(0, 1/3, 1/3, 1, 1/3), "monosaccharides-disaccharides" = c(0,
+         1/2, 1/2, 0, -1), "gluc-fruc" = c(0, 1, -1, 0, 0))
> m4 <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(contr = contr4)))
```

contrast options in R

By default R uses **treatment contrasts**. You can set the treatment contrasts explicitly with

```
> options(contrasts = c("contr.treatment", "contr.poly"))
```

For the example with pea data you could set the treatment contrasts manually

```
> treatment.contrast <- rbind(c(0, 1, 0, 0, 0), c(0, 0, 1, 0, 0),
+     c(0, 0, 0, 1, 0), c(0, 0, 0, 0, 1))
> treatment.contrast.names <- list("control - gluc", "control - fruc",
+     "control - glucfruc", "control - sucr")
> m.treatment <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(treatment.contrast,
+     contr.names = treatment.contrast.names)))
```

This shows that the treatment contrasts are not true contrasts. Every coefficient represents a comparison of one level with level 1; ommiting level 1 itself, which is given as intercept. For the data set "pea", the first coefficient (Intercept) is the mean of the *cont* treatment, the second the difference between *cont* and *gluc*, the third the difference between *cont* and *fruc*, etc.

The same coefficients with an intercept equal to the mean of all treatments can be obtained with:

```
> new.trt.cont <- rbind("contr-gluc" = c(1, -1, 0, 0, 0), "contr-fruc" = c(1,
+     0, -1, 0, 0), "contr-glucfruc" = c(1, 0, 0, -1, 0), "contr-sucr" = c(1,
+     0, 0, 0, -1))
```


contrast options in R ... continued

Three other useful contrast options and their meaning:

sum contrasts

```
> options(contrasts = c("contr.sum", "contr.poly"))
```

setting the sum contrasts manually:

```
> sum.contrast <- list(c(4/5, -1/5, -1/5, -1/5, -1/5), c(-1/5,
+     4/5, -1/5, -1/5, -1/5), c(-1/5, -1/5, 4/5, -1/5, -1/5), c(-1/5,
+     -1/5, -1/5, 4/5, -1/5))
> sum.contrast.names <- list("control - mean of all", "gluc - mean of all",
+     "fruc - mean of all", "glucfruc - mean of all")
> m.sum <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(contr = sum.contrast,
+     contr.names = sum.contrast.names)))
```

helmert contrasts

```
> options(contrasts = c("contr.helmert", "contr.poly"))
```

setting the helmert contrasts manually:

```
> helmert.contrast <- list(c(-1/2, 1/2, 0, 0, 0), c(-1/6, -1/6,
+     1/3, 0, 0), c(-1/12, -1/12, -1/12, 1/4, 0), c(-1/20, -1/20,
+     -1/20, -1/20, 1/5))
> helmert.contrast.names <- list("gluc-contr", "fruc-mean(contr,gluc)",
+     "glucfruc-mean(contr,gluc,fruc)", "sucr-mean(contr,gluc,fruc,glucfruc)")
> m.helmert <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(contr = helmert.contrast,
+     contr.names = helmert.contrast.names)))
```

successive differences (from package MASS)

```
> options(contrasts = c("contr.sdif", "contr.poly"))
```

setting the successive differences contrasts manually:

```
> sdif.contrast <- list(c(-1, 1, 0, 0, 0), c(0, -1, 1, 0, 0), c(0,
+     0, -1, 1, 0), c(0, 0, 0, -1, 1))
> sdif.contrast.names <- list("gluc-control", "fruc-gluc", "glucfruc-fruc",
+     "sucr-glucfruc")
> m.sdif <- lm(length ~ trt, data = pea, contrasts = list(trt = mancontr(contr = sdif.contrast,
+     contr.names = sdif.contrast.names)))
```


Theory of p–values of hypothesis tests and of coverage of confidence intervals applies to *a-priory* comparisons only!

The following example illustrates the problem:

```
> y <- rnorm(1000, sd = 1)
> x <- rep(1:100, each = 10)
> dat <- data.frame(y = y, x = as.factor(x))
> m <- aov(y ~ x, data = dat)
> summary(m)
```

Tukey's HSD

Student (WS Gosset) discovered the distribution of the t statistic when there are *two* groups to be compared and there is no underlying mean difference between them. When there are x groups, there are $x(x-1)/2$ pairwise comparisons that can be made. Tukey found the distribution of the *largest* of these t statistics when there are no underlying differences. Because the number of groups is accounted for, there is only a 5% chance that Tukey's HSD (Honest Significant Difference) will declare something to be statistically significant when all groups have the same population mean.

Although stated differently in several textbooks, Tukey's HSD is no *per se* conservative! If you test only a subset of the possible pairwise comparisons, it is of course conservative, because it assumes that you compare all.

The calculation of Tukey's HSD in R is straightforward:

```
> m0 <- aov(length ~ trt, data = pea)
> t0 <- TukeyHSD(m0)
> plot(t0)
```


a paradox?

A researcher compares the treatments a, b, and c. With a multiple comparison procedure no result achieves statistical significance. On the same planet there are three researcher one makes an experiment to test treatment a vs. b and another b vs. c. They both find no statistical significance. But the third researcher investigates the difference of a vs. c finds a significant result. He does not have to make adjustments and can impress others with his findings.

> The issues (of multiple comparisons) are non–technical and indeed there may be some concern that they may be overlooked if attention is focused only on technical aspects of multiple-testing procedures.
> (Cook, Richard J. & Farewell, Vern T. 1996)

Further reading:
Sokal, R. R. & Rohlf, J. F. (1995): Biometry: the principles and practice of statistics in biological research. Freeman, New York

exercise 1 Load the data frame "immer" from the package MASS. It has the variables "Var" for variety, "Loc" for locality and "Y1" and "Y2" for the yield in two successive years.

- **a)** Summaries the data set graphically with at least 4 different plots.
- **b)** Is there a significant difference in average yield (across Y1 and Y2) between varieties?
- **c)** Inspect the model you have fitted. What steps do you need to do? What are you looking at?
- **d)** You have an *a priory* strong interest to know whether there is a significant difference in average yield between the locality "C" and "D". Is there a significant difference?
- **e)** How many other comparisons can you make without adjusting for multiple comparisons.
- **f)** Select one full set of contrasts manually.
- **g)** You are *a priory* interested to know wheter one of the varieties M, P, T, or V has a higher average yield (across years) than the mean of all varieties?
- **h)** You have no *a priory* expectation for the different varieties of barley. Compute and plot the simultaneous confidence intervals for all pairwise differences in the average yield. Are there varieties that differ significantly?

linear models I

one–way ANOVA:

$$\begin{aligned} \mathrm{E}[y_{ij}] &= \mu + \alpha_i \\ y_{ij} &\sim \textit{indep.} \quad \mathcal{N}(\mu + \alpha_i, \sigma^2) \end{aligned}$$

two–way ANOVA:

$$\begin{aligned} \mathrm{E}[y_{ijk}] &= \mu + \alpha_i + \beta_j \\ y_{ijk} &\sim \textit{indep.} \quad \mathcal{N}(\mu + \alpha_i + \beta_j, \sigma^2) \end{aligned}$$

simple linear regression:

$$\begin{aligned} \mathrm{E}[y_i] &= \mu(x_i) = \alpha + \beta x_i \\ y_i &\sim \textit{indep.} \quad \mathcal{N}(\alpha + \beta x_i, \sigma^2) \end{aligned}$$

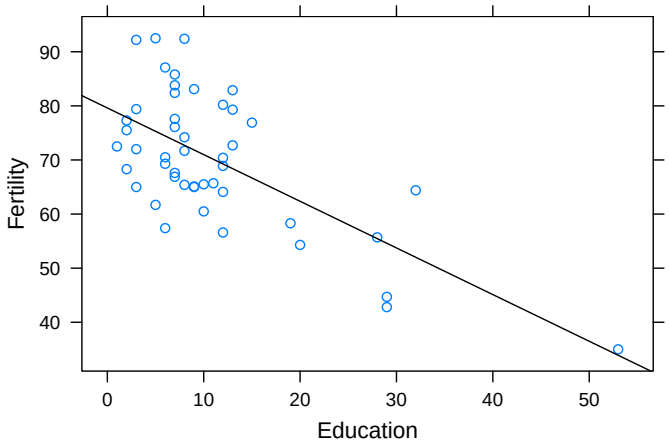
multiple linear regression:

$$\begin{aligned} \mathrm{E}[y_{ij}] &= \beta_0 + \beta_1 x_{1_i} + \beta_2 x_{2_j} \\ y_{ij} &\sim \textit{indep.} \quad \mathcal{N}(\beta_0 + \beta_1 x_{1_i} + \beta_2 x_{2_j}, \sigma^2) \end{aligned}$$

regression

regression ...a regression session

```
par(pch=16)
data(swiss)
plot(Fertility ~ Education, data=swiss, col="blue")
m0 <- lm(Fertility ~ Education, data=swiss)
abline(m0)
```




```
identify(swiss$Fertility ~ swiss$Education, )
```

```
plot(m0)
inspect(m0)                                  # library(asuR)
norm(m0)                                     # library(asuR)

# to see the influence of row 45
m0.s1 <- lm(Fertility ~ Education, data=swiss[row.names(swiss)!="V. de Geneve",])

summary(m0)
          :Call:
          :lm(formula = Fertility ~ Education, data = swiss)
          :
          :Residuals:
          :    Min      1Q  Median      3Q     Max
          :-17.036  -6.711  -1.011   9.526  19.689
          :
          :Coefficients:
          :             Estimate Std. Error t value Pr(>|t|)
          :(Intercept)  79.6101     2.1041  37.836  < 2e-16 ***
          :Education    -0.8624     0.1448  -5.954 3.66e-07 ***
          :---
          :Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
          :
          :Residual standard error: 9.446 on 45 degrees of freedom
          :Multiple R-Squared: 0.4406,        Adjusted R-squared: 0.4282
          :F-statistic: 35.45 on 1 and 45 DF,  p-value: 3.659e-07
```

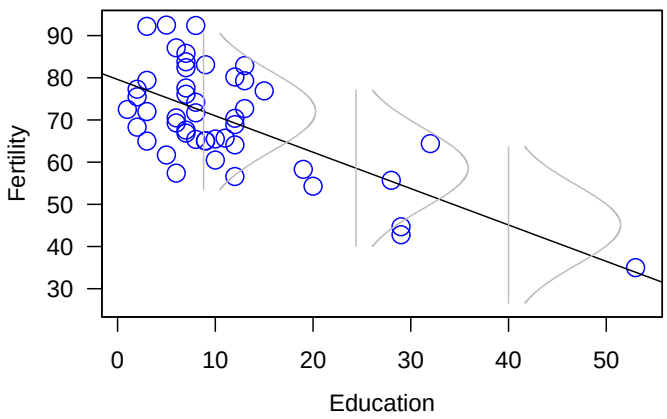



figure distribution of $y_i \mid x$ around the regression line

	Description
1.	Linearity
	Follows directly from the definition of the model
	$Y = X\beta + \epsilon$
2.	Computational
	The predictor variables are linearly independent. This is needed to find a unique estimate of β.
	rank(X)=k
3.	Distributional
a)	X is non–random
b)	X is measured without error
	The errors $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$
c)	−are normally distributed
d)	−are independent of each other
e)	−have zero mean
f)	−have a constant (but unknown) variance (σ^2)
4.	Implicit
a)	[a] The observations are equally reliable
b)	The observations have an equal influence

[a] Only needed if assumption 3.a) is violated.

Name	Assumption[a]	Expected Pattern	`Function`	Suggestion
Stud. residuals vs. each predictor	1, 3e,f)	random scatter about zero	`rxp()`	transformation missing/obsolete variable
Stud. residuals vs. predicted values	1, 3e,f)	random scatter about zero	`ryp()`	transformation
Normal prob. plot of stud. residuals	3c)	straight through origin (slope: 1)	`nrp()`	
Index plot of stud. residuals	3d,e)	random scatter about zero	`irp()`	
Index plot of the leverage	4b)	random scatter and small	`ilp()`	transformation
Index plot of Hadi's influence measure	4b)	random scatter and small	`ihp()`	transformation
Potential-Residual plot	4b)	random scatter (in lower,left corner)	`prp()`	look for better model or better data

[a] Numbered according to previous table

So far we assumed that the variables that go into the model equation are chosen in advance. In many situations, however, the set of variables to be included in the model equation is not *a priori* determined. Then the selection of predictor variables is an important step.

There are several predictor variable procedures. Procedures where one variable at a time is added or dropped are the *backward elimination*, the *forward selection*, or the *stepwise* method.

- *forward selection*:
  1) start with a model equation containing only a constant term
  2) add the most significant variable until no other significant variable can be added
- *backward elimination*:
  1) start with the most complex model equation
  2) drop the least significant variable until all variables are significant
- *stepwise*:
  1) start with a model equation containing only a constant term
  2) add the variable with the smallest $p-$value (if it is significant)
  3) drop the variable with the largest $p-$value (if it is not significant)
  repeat 2) and 3) until there is no variable to be added or dropped

This procedures should be used with caution and not mechanically! Without collinear data they will give nearly the same result.

```
f.min <- formula(Fertility ~ 1)
f.max <- formula(Fertility ~ Education*Agriculture*Examination*Catholic*Infant.Mortality)

m1 <- lm(Fertility ~ 1, data = swiss)              # step 1
addterm(m1, scope=f.max, test="F")

m2 <- update(m1, .~.+Education)                    # step 2
dropterm(m2, test="F")
addterm(m2, scope=f.max, test="F")

m3 <- update(m2, .~.+Catholic)                     # step 3
dropterm(m3, test="F")
addterm(m3, scope=f.max, test="F")

m4 <- update(m3, .~.+Infant.Mortality)             # step 4
dropterm(m4, test="F")
addterm(m4, scope=f.max, test="F")

m5 <- update(m4, .~.+Education:Catholic)           # step 5
dropterm(m5, test="F")
addterm(m5, scope=f.max, test="F")

m6 <- update(m5, .~. + Agriculture)                # last step
dropterm(m6, test="F")
addterm(m6, scope=f.max, test="F")

# automatic model selection with AIC: stepwise
step(m1, scope=list(upper=f.max, lower=f.min), direction="both")
# automatic model selection with AIC: foreward
step(m1, scope=list(upper=f.max, lower=f.min), direction="forward")
# automatic model selection with AIC: backward
m99 <- lm(f.max, data=swiss)                       # maximal model
step(m99, scope=list(upper=f.max, lower=f.min), direction="backward")
```


regression . . . practicals VI

exercise 1 Load the data set "hills" from package MASS.
The aim of analysing this data set is to find a formula that allows to predict the time of a race by the distance and the climb.

- **a)** If you go on a walk how do you predict the time you need as a function of distance and climb?
- **b)** For a simpler interpretation, transform the variable "dist" in kilometers (1mile=1.609km) and the variable "climb" in meters (1feet=304.8mm).
- **c)** Fit a multiple regression using the variable "dist" and " climb" as predictors (without interaction). Does the model meet your expectations from exercise *a)*?
- **d)** Inspect the model. Are there races with a high residual and/or leverage? -which?
- **e)** How many minutes of difference is between the predicted and the observed time of the "Knock Hill" race.
- **f)** If you study the coefficients of your model and their statistical significance carefully, you find a result that is impossible on physical grounds. Explain which coefficient is involved and why this result is impossible?

linear models II

a regression that is nonlinear in its parameters:

$$\begin{aligned} \textit{1)}\quad \mathrm{E}[log(y_i)] &= \alpha + \beta x_i \\ log(y_i) &\sim \textit{indep.}\quad \mathcal{N}(log(\alpha) + \beta log(x_i), \sigma^2) \end{aligned}$$

$$\begin{aligned} \textit{2)}\quad log(\mathrm{E}[y_i]) &= \alpha + \beta x_i \\ y_i &\sim \textit{indep.}\quad \mathcal{N}(e^{(\alpha+\beta x_i)}, \sigma^2) \end{aligned}$$

The decision between this two is made by checking the homoscedasticity,
in *1)* $\log y_i$ is homoscedastic,
in *2)* y_i is homoscedastic.

linear models II ...a second regression session

```
data(mytrees)                              # library(asuR)
m0 <- lm(log(Volume) ~ log(Girth)+ log(Height), data=mytrees)
```

$$\begin{aligned} \mathrm{E}[y_{ij}] &= \beta_0 + \beta_1 x_{1_i} + \beta_2 x_{2_j} \\ y_{ij} &\sim \textit{indep.}\quad \mathcal{N}(\beta_0 + \beta_1 x_{1_i} + \beta_2 x_{2_j}, \sigma^2) \end{aligned}$$

$$\begin{aligned} \mathrm{E}[\log Volume_{ij}] &= \log Intercept + \beta_1 \log Girth_i + \beta_2 \log Height_j \\ \mathrm{E}[Volume_{ij}] &= Intercept * Girth_i^{\beta_1} * Height_j^{\beta_2} \end{aligned}$$

$$\log Volume_{ij} \sim \textit{indep.}\quad \mathcal{N}(Intercept * Girth_i^{\beta_1} * Height_j^{\beta_2}, \sigma^2)$$

tapply & table

```
tapply(flowers$flower, flowers$alt, mean)
```

```
:    high      low
: 20.7804 999.6610
```

The function `tapply()` is used to apply a function, here `mean()`, to each group of components of the first argument, here `flower`, defined by the levels of the second component, here `alt`.

It is also possible to specify several grouping factors within a list,

```
tapply(plants$height, list(plants$family, plants$type), mean, na.rm=FALSE)
```

To get a contingency table (a table with the number of elements in each segment) you can use,

```
tapply(plants$height, list(plants$family, plants$type), length)
table(plants$family, plants$type)
```

```
:            herbaceous shrub tree
:   Fabaceae        218    56    6
:   Rosaceae         76    76   28
```


linear models III

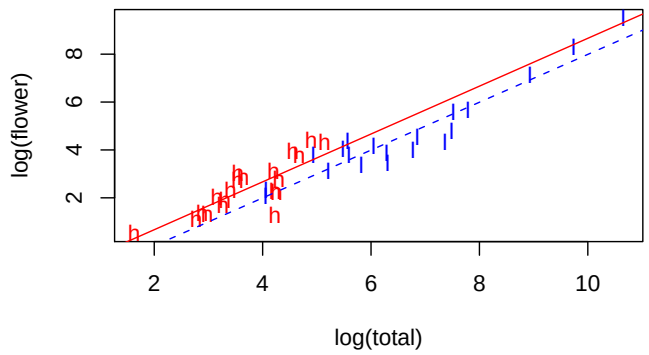
ANCOVA, analysis of covariance:

$$\begin{aligned} \mathrm{E}[y_{ij}] &= \beta_0 + \beta_{1_j} + \beta_2 x_i \\ y_{ij} &\sim \textit{indep.} \quad \mathcal{N}(\beta_0 + \beta_{1_j} + \beta_2 x_i, \sigma^2) \end{aligned}$$

```
plot(log(flower) ~ log(total), type="n", data=flowers)
points(log(flower) ~ log(total), data=flowers[flowers$alt=="high",], col="red")
points(log(flower) ~ log(total), data=flowers[flowers$alt=="low",], col="blue")
abline(lm(log(flower) ~ log(total), data=flowers[flowers$alt=="high",]),col="red")
abline(lm(log(flower) ~ log(total), data=flowers[flowers$alt=="low",]), col="blue")
identify(log(flowers$flower) ~ log(flowers$total))

# a model with two intercepts and two slopes
m1 <-lm(log(flower) ~ alt/log(total) -1, data=flowers)
# a model with two intercepts and one slope
altdiff <- rbind("high-low"=c(1,-1))
m2 <- lm(log(flower) ~ alt + log(total), data=flowers, contrasts=list(alt=mancontr(contr=altdiff)))
 # are separate slopes needed?
anova(m1, m2)

#inspection
inspect(m2)

#different subsets
subset <- c(1,9)
m1.s1 <-update(m1, .~., subset=-c(subset))
m2.s1 <- update(m2,.~., subset=-c(subset))
anova(m1.s1, m2.s1)
print(coef(m2))
print(coef(m2.s1))
print(diff <- coef(m2.s1)-coef(m2))
summary(m2.s1)
```



```
#drawing a new plot with the fitted lines
plot(log(flower) ~ log(total), type="n", data=flowers)
points(log(flower) ~ log(total), data=flowers[flowers$alt=="high",], col="red", pch="h")
points(log(flower) ~ log(total), data=flowers[flowers$alt=="low",], col="blue", pch="l")
abline(-1.33038, 0.99920, col="red")
abline(-2.0019, 0.99920, col="blue", lty=2)
```



exercise 1 Load the data set "gala" from package asuR. Read the documentation.

- **a)** Select the predictors for a regression model of the number of tortoise species on the 30 Galapagos islands using stepwise model selection.
- **b)** What predictors would you select with a backwards and forewords model selection.
- **c)** Inspect the model you found in *a)* carefully and describe possible deviations from the model assumptions. (islands with high leverage?, are this islands influential?, on which parameters? are there islands with high residuals?)
- **d)** Do you find a good transformation for the response variable or the predictors that bring you closer to the model assumptions?

exercise 2 Load the data set "cathedral" from package asuR. Read the documentation.

- **a)** Perform an analysis of covariance.
- **b)** Make a scatterplot of the data and add your final model line(s).

Generalized Linear Models

glm ... basic structure

A generalized linear model is determined by:

- the form of the linear predictor, η (coding and selection of predictor variables)
  $\eta = \mathbf{z}_i^T \beta \quad (\text{e.g.}, \alpha + \beta x_i)$
- the response or link function, h or g
  $\mu_i = h(\eta_i) = h(\alpha + \beta x_i)$
  link function, g (the inverse of h):
  $g(\mu_i) = \nu_i = \alpha + \beta x_i$
- the type of the exponential family which specifies the distribution of y_i, given z_i

glm ... continuous response: normal regression

$$
\begin{array}{rcll}
\eta_i & = & \alpha + \beta x_i & \text{linear predictor} \\
\mathrm{E}[y_i] & = & h(\eta_i) = \eta_i & \text{response function: identity} \\
y_i & \sim & \textit{indep.} \quad \mathcal{N}(\alpha + \beta x_i, \sigma^2) & \text{distribution: normal with constant variance}
\end{array}
$$

```
> m.lm <- lm(Fertility ~ Education, data = swiss)
> m.glm <- glm(Fertility ~ Education, data = swiss, family = gaussian(link = identity))
```

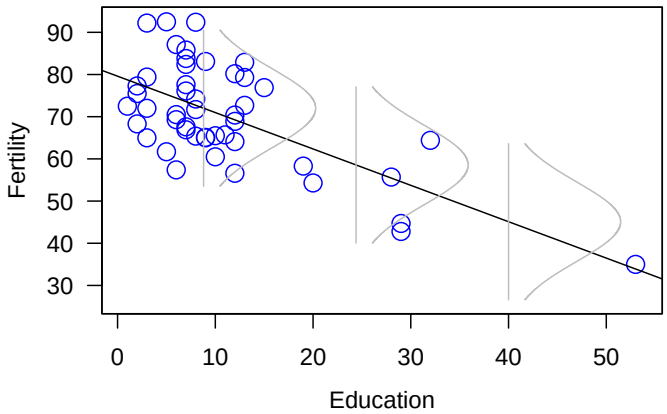


figure distribution of $y_i \mid x$ around the regression line

glm ... countinous response: normal regression with log link

$$
\begin{array}{rcll}
\eta_i & = & \alpha + \beta x_i & \text{linear predictor} \\
\mathrm{E}[y_i] & = & h(\eta_i) = e^{(\eta_i)} = e^{\alpha} \cdot e^{\beta x_i} & \text{response function: exponential} \\
y_i & \sim & \textit{indep.} \quad \mathcal{N}(e^{\alpha} \cdot e^{\beta x_i}, \sigma^2) & \text{distribution: normal with constant variance}
\end{array}
$$

```
> m.glm.gl <- glm(Fertility ~ Education, data = swiss, family = gaussian(link = log))
```

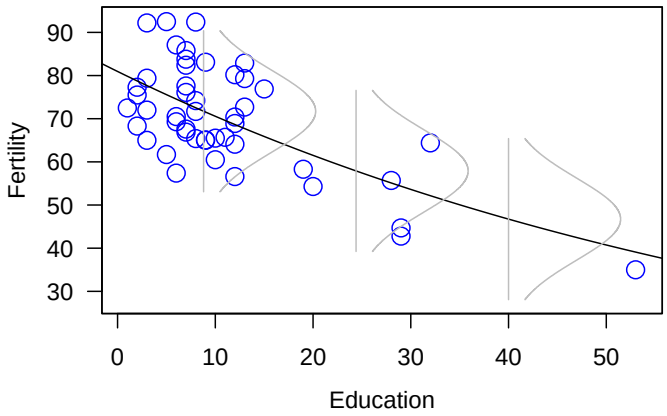


figure distribution of $y_i \mid x$ around the regression line

glm ... countinous response: log link with a gamma distribution

$$
\begin{array}{rcll}
\eta_i & = & \alpha + \beta x_i & \text{linear predictor} \\
\mathrm{E}[y_i] & = & h(\eta_i) = e^{(\eta_i)} = e^{\alpha} \cdot e^{\beta x_i} & \text{response function: exponential} \\
y_i & \sim & \textit{indep.} \quad \mathbf{\Gamma}\{\phi, \phi/(\alpha + \beta x_i)\} & \text{distribution}
\end{array}
$$

```
> m.glm.Gl <- glm(Fertility ~ Education, data = swiss, family = Gamma(link = log))
```

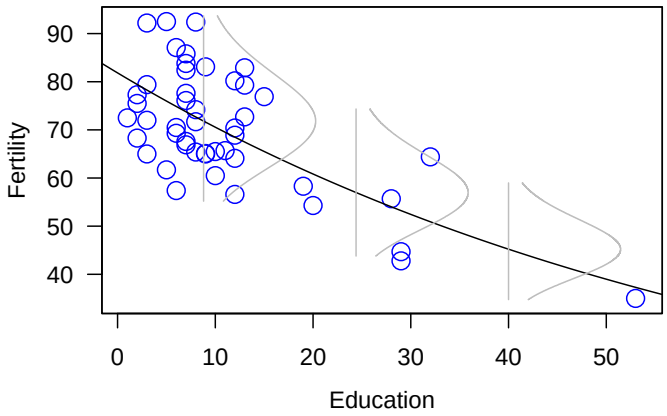


figure distribution of $y_i \mid x$ around the regression line

Plot the deviance residuals versus the estimated values of the linear predictor

```
> dep(your.model)
```

Questions:

A) is the relationship linear?

otherwise:

1. change the choice of predictors
2. change the transformations of the predictor
3. change the link function (but there are only few choices...)
4. do not transform the response in glm since this would change the distribution of the response (you would do this in a lm)

B) is the variance constant?

if not:

1. change the variance function
2. use weights if you identify some features of the data that suggest a suitable choice

relationship between predictors and response

A partial residual plot allows to study the effect of the predictor in focus and taking the other predictors into account. Partial residuals are calculated as:

$$e^*_{ij} = e_i + \hat{\beta}_j \mathbf{X}_{ij} \quad (1)$$

```
> rpp(your.model)
```

Question:

Is the relationship linear?

otherwise:

1. change the transformations of the predictor

checking the link assumption

Plotting the linearized response against the linear predictor, $\hat{\eta}$

```
> lep(your.model)
```

Question:

Is the relationship linear?

otherwise:

1. change the link function

unusual points

```
> hnp(your.model)
```

Question:

are there points off the trend?

otherwise:

1. Is the data point correct?

glm ...binomial response with continuous covariate

odds: $\gamma(\pi) = \frac{\pi}{1-\pi}$, where π is the probability of success.

$$
\begin{array}{rcll}
\eta_i & = & \alpha + \beta x_i & \text{linear predictor} \\
\mathrm{E}[y_i] & = & \pi_i = h(\eta_i) = exp(\eta)/(1+exp(\eta)) & \text{link function g(): logit} \\
y_i & \sim & indep. \quad \mathcal{B}(\pi_i), \text{withvar}(y-i) = \pi_i(1-\pi_i) & \text{distribution: Bernoulli}
\end{array}
$$

$\frac{\pi}{1-\pi} = exp(\alpha + \beta x_i)$, and therefore
$e^{\beta} = \frac{\gamma(x+1)}{\gamma(x)}$

This shows that the coefficient β can be interpreted in a natural way because e^{β} corresponds to the factor by which odds, $\gamma(x)$, increase (if $\beta > 0$) or decrease (if $\beta < 0$) if x is increased by one unit.

glm ...binomial response with continuous covariate ...example

```
> model.glm <- glm(fail ~ temperature, data = oring, family = binomial)
> summary(model.glm)
```

```
: ...
: Coefficients:
:             Estimate Std. Error z value Pr(>|z|)
: (Intercept)  15.0429     7.3786   2.039   0.0415 *
: temperature  -0.2322     0.1082  -2.145   0.0320 *
: ...
```

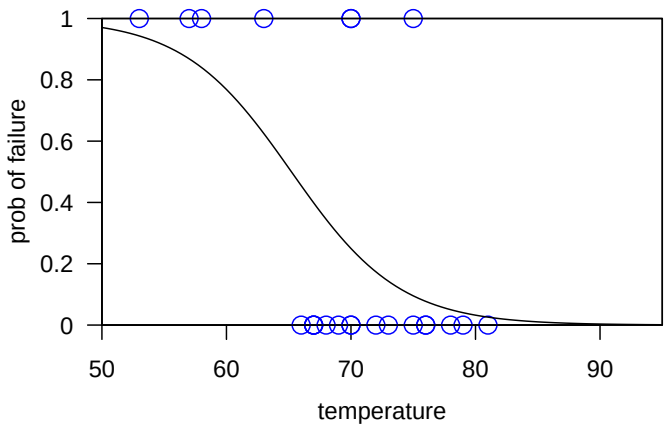


figure probability of an O-ring failure over a temperature range. In this example $\beta = -0.2322$, which means that the odds decrease with every degree Fahrenheit by $e^{-0.2322} = 0.793$. Another helpful mark is the temperature where half of all O-rings fail $-\alpha/\beta = -15.0429/-0.2322 = 64.8$

glm ...binomial response with categorical covariate

There are three forms to specify a generalized linear models for this data:

1.
```
y.matrix <- cbind(#success, #faillures)
model <- glm(y.matrix ~ ..., family=binomial)
```

2. with success as a logical vector or a two-level factor (the first level is treated as 0, all others as 1!)

```
success <- c(TRUE, FALSE, TRUE, TRUE, TRUE, FALSE,...)
```

   or

```
success <- factor(c(0,1,0,1,0,1,1,0,0,...))
model <- glm(success ~ ..., family=binomial)
```

3.
```
y <- #success/#total
weight <- #success*#total
model <- glm(y ~ ..., family=binomial, weights=weight)
```


glm ...binomial response with categorical covariate ...example

Duration of unemployment:

	≤ 6 months	> 6 months
male	403	167
female	238	175

```
> data(unemployment, package = "asuR")
> my.contrast.matrix <- rbind("male-female" = c(1, -1))
> my.model.object <- glm(success ~ gender, data = unemployment,
+     family = binomial, contrasts = list(gender = mancontr(my.contrast.matrix)))
> summary(my.model.object)
> exp(0.5735)
> p <- c(female = 175/(238 + 175), male = 167/(403 + 167))
> odds <- p/(1 - p)
> odds["female"]/odds["male"]
```

exercise 1 Describe the three elements that determine a generalized linear model?

exercise 2 Load the data set "gala" from package asuR again.

a) Fit a generalized linear model to the data by selecting an appropriate distribution, a link function, the predictor variables and their transformation. Discuss your selections.
b) Compare with the model that you selected in the previous practicals.

exercise 3 Load the data set "budworm" from package asuR.

a) Is there a difference in death rate between genders?
b) Do the genders respond differently to an increasing dose of toxin?

random effects

fixed effect An effect that is attributable to a finite set of levels of a factor that occur in the data and which are there because we are interested in them. The interest in fixed effects lies in estimating the mean.

random effect An effect that is attributable to a (usually) infinite set of levels of a factor, of which only a random sample are deemed to occur in the data. The interest in random effects lies in estimating their variance. Random effects are not defined on a continuum but they are generally real objects.

$$\begin{aligned} \mathrm{E}[y_{ijk} \mid a_i] &= \mu + a_i + \beta_j \\ y_{ijk} \mid a_i &\sim \textit{indep.} \quad \mathcal{N}(\mu + a_i + \beta_j; \quad \sigma_a^2 + \sigma^2) \end{aligned}$$

random effects ... nested random factors

$$\begin{aligned} \mathrm{E}[y_{tijk} \mid s_i, c_{ij}] &= \mu + \beta_t + s_i + c_{ij} \\ y_{tijk} \mid s_i, c_{ij} &\sim \textit{indep.} \quad \mathcal{N}(\mu + \beta_t + s_i + c_{ij}; \quad \sigma_s^2 + \sigma_c^2 + \sigma^2) \end{aligned}$$

```
m0 <- lmer(y ~ gen + (1|school/class), data=schoolclass)
```


random effects ... crossed random factors

$$\begin{aligned} \mathrm{E}[y_{tijk} \mid r_i, c_j] &= \mu + \beta_t + r_i + c_j \\ y_{tijk} \mid r_i, c_j &\sim \textit{indep.} \quad \mathcal{N}(\mu + \beta_t + r_i + c_j; \quad \sigma_r^2 + \sigma_c^2 + \sigma^2) \end{aligned}$$

```
m0 <- lmer(int ~ trt + (1|row) + (1|col), data=wellplate)
```

exercise 1 Load the data set "flowers" form package asuR.

a) Sort the data set by the variable "alt" and within "alt" sort it by the variable "total".
b) Count the number of plants with flowers that are heavier than 20 mg at high and low altitude.
c) Put the rows of the data set in a random order.
d) Produce a scatterplot of with the variable "total" on the $x-$axis and the variable "flower" on the $y-$axis. Use a different plotting style for plants from high and low altitude, that on a black and white print it is easy to distinguish between them.
e) Produce a histogram with a density estimate for the total plant mass of species from high altitude. Redo the same plot after taking the logarithm of plant mass. Which data characteristics change (text)? Is the data normally distributed (text)?

exercise 2 Load the data set "weight" from package asuR.

a) Calculate the mean and variance for all four combinations of protein `source` and `amount`.
b) To study the `weightgain` in rats, fit an two-way analysis of variance with an interaction term to the data.
c) (text) Inspect the model. Describe one diagnostic plot you are looking at. Which assumption(s) can you inspect with this plot? What pattern do you expect to see. Do you find this pattern? Are there deviations from you expectation?
d) You do not remember how the contrasts option is set on your computer anymore. Set the contrasts manually to compare the `weightgain` for proteins from "Beef" and "Cereal". What is the estimate for the difference between the two protein sources?

exercise 3 Load the data set "BtheB" from package asuR. (bdi means: "Beck Depression Inventory")

a) Rearrange the data from the "wide form" in which it appears into the "long form" in which each separate repeated measurement and associated covariate values appear as a separate row in a data frame. Make one column called `drug, length, treatment, bdi.pre, subject, time, bdi`.

b) Construct the boxplots of each of the five repeated measures separately for each treatment group. (You can run the example code from the documentation of the data set to see how the result should look; use the methods and functions we learned in the course to construct similar plots)

c) Fit a linear mixed effect model where the subject has a random effect on the intercept (using `lmer()` from library lme4).

d) Fit a linear mixed effect model where the subject has a random effect on the intercept and the slope.

e) Test whether the more complex model is significantly better than the simple one.

bootstrap

bootstrap . . . nonparametric bootstrap

```
> data(flowers)
> plantmass <- flowers[flowers$alt == "high", "total"]
> flowers.fun <- function(data, i) {
+     mean(data[i])
+ }
> b0.n <- boot(data = flowers[flowers$alt == "high", "total"],
+     statistic = flowers.fun, R = 99)
> plot(b0.n)
> print(b0.n)
> mean(plantmass) - qt(0.975, length(plantmass) - 1) * sqrt(var(plantmass))/sqrt(length(plantmass))
> mean(plantmass) + qt(0.975, length(plantmass) - 1) * sqrt(var(plantmass))/sqrt(length(plantmass))
> sqrt(var(plantmass))/sqrt(length(plantmass))
> boot.ci(b0.n, type = c("perc"), conf = 0.9)
> sort(b0.n$t)
> boot.ci(b0.n, type = c("basic"), conf = 0.9)
> boot.ci(b0.n, type = c("norm"), conf = 0.9)
```


bootstrap . . . parametric

```
> flowers.fun <- function(data) {
+     mean(data)
+ }
> flowers.sim <- function(data, mle) {
+     rnorm(length(data), mean = mle[1], sd = mle[2])
+ }
> flowers.mle <- c(mean(plantmass), sqrt(var(plantmass)))
> b0.p <- boot(data = plantmass, statistic = flowers.fun, R = 99,
+     sim = "parametric", ran.gen = flowers.sim, mle = flowers.mle)
> plot(b0.p)
> print(b0.p)
> boot.ci(b0.p, type = c("perc", "basic", "norm"), conf = 0.9)
```

```
> fit.model <- function(data, i) {
+     l1 <- lm(log(flower) ~ log(total) + alt, data = data[i, ])
+     coef(l1)["altlow"]
+ }
> b0 <- boot(data = flowers, statistic = fit.model, R = 99)
> boot.ci(b0, type = c("norm", "perc", "basic"))
```


References

Burdick, R. K. & Graybill, F. A. (1992): Confidence Intervals on Variance Components. Marcel Dekker, New York.

Cleveland, William S. & McGill, Robert (1985): Graphical Perception and Graphical Methods for Analyzing Scientific Data. Science **229**(4716):828–833.

Cook, Richard J. & Farewell, Vern T. (1996): Multiplicity Considerations in the Design and Analysis of Clinical Trials. Journal of the Royal Statistical Society. Series A (Statistics in Society) **159**(1):93–110.

Murrell, P. (2005): R Graphics. Chapman & Hall.

Pinheiro, J. C. & Bates, D. M. (2000): Mixed-effects models in S and S-PLUS. Springer-Verlag New York.

Satterthwaite, F. E. (1946): An Approximate Distribution of Estimates of Variance Components. Biometric Bull **2**(6):110–114.

Sokal, R. R. & Rohlf, J. F. (1995): Biometry: the principles and practice of statistics in biological research. Freeman, New York.

exercise 1 see the sequence of logical operators in the appendix

exercise 2
```
> rep(1:3, each = 2)
> rep(1:2, times = 3)
> rep(1:3, each = 2, times = 3)
```

exercise 3
```
> paste("trt:", rep(1:3, each = 2), sep = "")
> paste("ind:", rep(c(1, 2), times = 3), sep = "")
```

exercise 4
```
> experiment <- data.frame(treatment = paste("trt:", rep(1:3, each = 2),
+     sep = ""), individual = paste("ind:", rep(c(1, 2), times = 3),
+     sep = ""))
> dim(experiment)
> str(experiment)
> experiment <- cbind(experiment, 2:7)
> save(experiment, file = "~/Desktop/experiment.Rdata")
> rm(experiment)
> load(file = "~/Desktop/experiment.Rdata")
> write.table(experiment, file = "~/Desktop/experiment.txt")
> rm(experiment)
> experiment <- read.table(file = "~/Desktop/experiment.txt", header = TRUE)
```


exercise 1
```
> mat <- matrix(1:16, nrow = 4)
> mat[, 3]
> diagonal <- diag(mat)
> dia <- cbind(c(1:4), c(1:4))
> diagonal <- mat[dia]
> antidia <- cbind(c(1:4), c(4:1))
> antidiagonal <- mat[antidia]
```

exercise 2
```
> vec <- as.integer(1:1000)
> even.numbers <- seq(from = 2, to = 1000, by = 2)
> even.numbers <- vec[vec%%2 == 0]
> vec[even.numbers] <- 1/vec[even.numbers]
> vec[seq(from = 2, to = 1000, by = 2)] <- 1/(vec[seq(from = 2,
+     to = 1000, by = 2)])
```

exercise 3
```
> help(swiss)
> swiss[order(swiss$Agriculture), ]
> swiss <- cbind(swiss, religion = factor(NA, levels = c("catholic",
+     "protestant")))
> swiss$religion[swiss$Catholic >= 50] <- "catholic"
> swiss$religion[swiss$Catholic < 50] <- "protestant"
> swiss[order(swiss$religion, swiss$Agriculture), ]
> swiss[order(row.names(swiss)), ]
> order(row.names(swiss))[dim(swiss)[1]:1]
> order(row.names(swiss), decreasing = TRUE)
> swiss[sample(row.names(swiss)), ]
> swiss$Education <- NULL
```

solutions . . . practicals III

exercise 1

```
> data(trees)
> pairs(trees)
> xxp(trees)
> pdf(file = "~/Desktop/trees_example.pdf", width = 13/2.54, height = 9/2.54)
> plot(trees$Volume ~ trees$Girth)
> plot(Volume ~ Girth, data = trees)
> plot(trees$Girth, trees$Volume)
> dev.off()
```

exercise 2

```
> hist(swiss$Fertility)
> plot(swiss$religion, swiss$Fertility)
```

exercise 3

```
> pdf("~/Desktop/swissdata.pdf", width = 13/2.54, height = 9/2.54)
> par(mar = c(4, 4, 4, 4))
> plot(Agriculture ~ Fertility, data = swiss, main = "swiss data (1888)",
+     xlab = "fertility", xlim = c(1, 100), ylim = c(1, 100), axes = FALSE,
+     ylab = "", col = "red")
> points(Examination ~ Fertility, data = swiss, pch = 3, xlim = c(1,
+     100), col = "blue")
> axis(side = 1)
> axis(side = 2, col.axis = "red", col = "red")
> mtext(text = "agriculture", side = 2, line = 3, col = "red")
> axis(side = 4, col.axis = "blue", col = "blue")
> mtext(text = "examination", side = 4, line = 3, col = "blue")
> legend(x = 10, y = 90, legend = c("agriculture", "examination"),
+     col = c("red", "blue"), text.col = c("red", "blue"), pch = c(1,
+         3))
> dev.off()
```


solutions . . . practicals IV

exercise 1

```
> t.sample <- rt(100, df = 3)
> qqnorm(t.sample)
> qqline(t.sample)
> hist(t.sample, freq = FALSE)
> lines(density(t.sample))
```

exercise 2

```
> norm.sample <- rnorm(100)
> qqnorm(norm.sample)
> qqline(norm.sample)
> hist(norm.sample)
```

exercise 3

```
> pdf("~/Desktop/comparet_norm.pdf", width = 20/2.54, height = 20/2.54)
> par(mfrow = c(2, 2))
> qqnorm(t.sample)
> qqline(t.sample)
> qqnorm(norm.sample)
> qqline(norm.sample)
> hist(t.sample)
> hist(norm.sample)
> dev.off()
```

solutions . . . practicals V

exercise 1

```
> m <- aov((Y1 + Y2)/2 ~ Loc + Var, data = immer)
> summary(m)
> co <- rbind("C-D" = c(1, -1, 0, 0, 0, 0))
> m1 <- aov((Y1 + Y2)/2 ~ Loc + Var, contrasts = list(Loc = mancontr(co)),
+     data = immer)
> tk <- TukeyHSD(m, which = "Var")
> plot(tk)
> densityplot(~(Y1 + Y2)/2 | Var, data = immer)
> tapply((immer$Y1 + immer$Y2)/2, immer$Var, mean)
```


solutions . . . practicals VI

exercise 1

```
> data(hills)
> hills.si <- hills
> hills.si$dist <- hills.si$dist * 1.609
> hills.si$time <- hills.si$time/60
> hills.si$climb <- hills.si$climb * 0.3048
> n0 <- lm(time ~ dist + climb, data = hills.si)
> plot(n0)
> inspect(n0)
> influence.measures(n0)
> summary(n0)
> residuals(n0)["Knock Hill"]
> hills.si[row.names(hills.si) == "Knock Hill", "time"]
> n0.s1 <- update(n0, . ~ ., subset = -18)
> n1 <- lm(time ~ dist * climb, data = hills.si, subset = -18)
> inspect(n0)
```

exercise 1

```
> data(flowers)
> flowers[order(flowers$alt, flowers$total), ]
> subset <- flowers[flowers$flower > 20, ]
> table(subset$alt)
> flowers[sample(row.names(flowers)), ]
> plot(flower ~ total, type = "n", data = flowers)
> points(flower ~ total, data = flowers[flowers$alt == "high",
+     ], col = "red", pch = "h")
> points(flower ~ total, data = flowers[flowers$alt == "low", ],
+     col = "blue", pch = "l")
> par(mfrow = c(1, 2))
> x <- flowers[flowers$alt == "high", "total"]
> hist(x, freq = FALSE, main = "untransformed")
> lines(density(x))
> hist(log(x), freq = FALSE, main = "log transformed")
> lines(density(log(x)))
> norm(log(x))
```


exercise 2

```
> tapply(weight$weightgain, list(weight$source, weight$type), mean)
> tapply(weight$weightgain, list(weight$source, weight$type), var)
> m1 <- lm(weightgain ~ source * type, data = weight)
> levels(weight$source)
> newcontr <- rbind("beef-cereal" = c(1, -1))
> m1 <- lm(weightgain ~ source * type, data = weight, contrasts = list(type = mancontr(newc
```

exercise 3

```
> data(BtheB)
> BtheB$subject <- factor(row.names(BtheB))
> nobs <- nrow(BtheB)
> BtheB.long <- reshape(BtheB, idvar = "subject", varying = c("bdi.2m",
+     "bdi.4m", "bdi.6m", "bdi.8m"), direction = "long")
> BtheB.long$time <- rep(c(2, 4, 6, 8), rep(nobs, 4))
> plot(bdi ~ time, data = BtheB.new[treatment == "TAU", ], xlab = "Time (in months)",
+     ylab = "BDI", main = "Treated as Usual", ylim = c(0, 55))
> plot(bdi ~ time, data = BtheB.new[treatment == "BtheB", ], xlab = "Time (in months)",
+     ylab = "BDI", main = "Beat the Blues", ylim = c(0, 55))
> btb.1 <- lmer(bdi ~ bdi.pre + time + treatment + drug + length +
+     (1 | subject), data = BthtB_long, na.action = na.omit)
> btb.2 <- lmer(bdi ~ bdi.pre + time + treatment + drug + length +
+     (time | subject), data = BthtB_long, na.action = na.omit)
> anova(btb1, btb.2)
```



```
data                    : [1] 0.6 0.5 0.2 0.1 0.9 0.4 0.3 0.8 0.7
sort(data)              : [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
order(data)             : [1] 4 3 7 6 2 1 9 8 5
rank(data)              : [1] 6 5 2 1 9 4 3 8 7
sample(data)            : [1] 0.5 0.8 0.1 0.4 0.3 0.7 0.2 0.9 0.6
```

Application:

```
dat <- data.frame(trt=c("cold","hot","cold","cold","hot","hot"), growth=c(8.5,9.8,5.1,6.2,7.1,8.6))
```

Sort a data frame according to the treatment (variable "trt"):

```
dat[order(dat$trt),]
```

Put the rows of the data frame in a random order:

```
dat[sample(row.names(dat))],
```

appendix . . . locigal operators

`$`	list extraction
`@`	slot extraction
`[ [[`	vector & list element extraction
`^`	exponentiation
`-`	unary minus
`:`	sequence generation
`%% %/% %*%`	and other special operators %. . . %
`* /`	multiply, divide
`+ - ?`	addition, subtraction, documentation
`< > <= >= == !=`	comparison operators
`!`	logical negation
`& &&`	logical operators
`| ||`	logical operators
`~`	formula
`<<-`	assignment within a function
`<-`	assignment

appendix . . . missing, indefinite, infinite values

`NA` Not Available

The value `NA` marks a "not avalailable" or "missing" value. Missing values are handeled differently by different functions. Many function have an argument like `na.rm=TRUE` to remove missing values. Other functions ignore the total row of a data frame if it contains missing values.

`NaN` Not a Number

```
1/0 - 1/0                    : NaN
```

`Inf` Infinity

```
1/0                          : Inf
```

family	accepted links	
gaussian	*identity*	
	log	
	inverse	
binomial	*logit*	(logistic)
	probit	(normal)
	cauchit	
	log	
	cloglog	complementary log-log
Gamma	*inverse*	
	identity	
	log	
poisson	*log*	
	identity	
	sqrt	
inverse.gaussian	*1/mu^2*	
	inverse	
	identity	
	log	

Note: Models with the same linear predictor, η, but with different link functions, g, can be compared informally, but not tested, by comparing the residual deviance.

appendix . . . recycling rule

In expressions that combine long and a short vectors, the shorter vectors are *recycled* until they match the length of the longest one. Fractional recycling is allowed but will print a warning message.

```
x <- c(2,3,4,5)                 # a vector of length 4
y <- 2                          # a vector of length 1
z <- c(3,4,5)                   # a vector of length 3
x*y                             : [1]  4  6  8 10
                                # a vector of length 4
x*z                             : [1]  6 12 20 15
                                : Warning message:
                                : longer object length
                                :        is not a multiple of shorter object length in: x * z
                                # a vector of length 4 (with a warning!)
```


appendix . . . labelling axes

strings with a special meaning

`\n`	new line
`\t`	tabulator

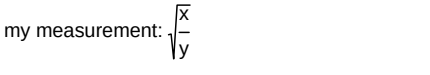
For an overview of possible mathematical notation see:

```
demo(plotmath)
```

```
expression(paste("temperature (", degree,"C)"))
```


```
expression(paste("conc. (", mg/mg[plain("dry mass")],")"))
```


```
expression(paste("my measurement: ", sqrt(frac(x,y))))
```




```
expression(paste(y[i] - bold(z)[i]^T, bold(beta)))
```

figure Some examples of the usage of the function expression. If you need to put labels on several lines it is often easier to use several calls to the low level function `mtext()` than creating one label that spans over several lines.

appendix ... Satterthwaite

Table 1: ANOVA table for balanced data

	df	Mean Squares (MS)	E(MS)
factor A	$A-1$	$\sum_{a=1}^{A}\sum_{\mathrm{b}=1}^{B} I(\bar{z}_a - \bar{z})^2$ / df_A	$\sigma^2_\mathsf{R} + I\sigma^2_\mathsf{B} + IB\sigma^2_\mathsf{A}$
factor B	$A(B-1)$	$\sum_{a=1}^{A}\sum_{\mathrm{b}=1}^{B} I(\bar{z}_{a\mathrm{b}} - \bar{z}_a)^2$ / df_B	$\sigma^2_\mathsf{R} + I\sigma^2_\mathsf{B}$
residual, R	$T-AB$	$\sum_{a=1}^{A}\sum_{\mathrm{b}=1}^{B}\sum_{i=1}^{I}(z_{a\mathrm{b}i} - \bar{z}_{a\mathrm{b}})^2$ / df_R	σ^2_R

where
factor A with the levels ($a = 1, 2, \ldots, A$)
factor B with the levels ($\mathrm{b} = 1, 2, \ldots, B$) in every level of the factor A
and for every level of the factor B the replicates ($\mathrm{i} = 1, 2, \ldots, I$)
and T is the total number of measurements, z.

appendix ... Satterthwaite

Very often approximate intervals on sums of expected mean squares are constructed using the Satterthwaite procedure (first proposed by Smith 1936 and later by Satterthwaite 1941 and SATTERTHWAITE (1946)). It was developed to estimate the distribution of sums of expected mean squares, where expected mean squares are a linear combination of variance components. This approach is based on a chi-squared approximation of the estimator for γ, $\hat{\gamma} = \sum_i c_i s_i^2$. One determines the value of m that equates the first two moments of $m\hat{\gamma}/\gamma$ to those of a chi-squared random variable with m degrees of freedom.

$$m = \frac{\hat{\gamma}^2}{\sum_i \frac{c_i^2 s_i^4}{n_i}} \tag{2}$$

This approximation works well when the n_i values are all equal or all large. However, when differences among the n_i are large, the Satterthwaite approximation can produce unacceptably liberal confidence intervals. Lower intervals (upper bounds) are more liberal than upper intervals (lower bounds). This approximation should not be used if some c_i are negative and some are positive (BURDICK & GRAYBILL 1992).

appendix ... some useful results

$\hat{y}_i$ is determined by $\frac{1}{h_{ii}}$ observations (h_{ii} is the leverage of the i^{th} observation).

$$\text{var}(e_i) = \sigma^2(1 - h_{ii}) \tag{3}$$

internally studentized residuals:

$$e_i' = \frac{e_i}{s\sqrt{1 - h_{ii}}} \tag{4}$$

externally studentized residuals: (because if an error is very large then s will be too large)

$$e_i^{*\prime} = \frac{e_i}{s_{(i)}\sqrt{1 - h_{ii}}} \tag{5}$$

where $s_{(i)}$ is calculated with all but the i^{th} observations.

Because internally and externally studentized residuals are monotonously related you can graphically not distinguish them.

asuR exam

I would like to get a folder called *your_family_name* containing one file with `R` code that can be processed line by line (*your_family_name.R*) and all graphical output (*exercise_xy.pdf*).
duration: 2 hours / materials: everything that is helpful

elegant R syntax (*+1pt*), clearly structured syntax with comments (*+1pt*)

exercise 1 The data set "plants" shows the height of all species of the Fabaceae and Rosaceae growing in Switzerland together with their growth type.

- **a)** Sort the data frame by `family`, `type`, and `height` (*1pt*).
- **b)** Select the `height` of all herbaceous species from the family Fabaceae. Store your selection that you can use it in the following examples (*1pt*).
- **c)** Construct a histogram with a density line of this selection (pdf 9×13 *cm*; *1pt*).
- **d)** You can see that your selection is not at all normally distributed. Do you find a transformation to normalize your selection? How can you inspect whether your transformed selection is normally distributed; use three different ways to inspect your transformed selection (*2pt*).
- **e)** Construct a data frame from all Rosaceae that are of growth type shrub and remove all unused levels for the factors `family` and `type`; hint: you can see the levels of a factor with the function `levels()` (*2pt*).

asuR exam ... continued

exercise 2 The data set "houseflies" shows the mean duration of development (in days) for 3 strains of houseflies for 2 different treatments.

- **a)** Calculate the mean developmental time for all strain and treatment combinations (*2pt*).
- **b)** Use a stepwise model selection procedure to find a suitable model, where the response variable `duration` is explained by the predictors `treatment` and `strain` (*1pt*).
- **c)** You have no *a priory* expectation for the developmental duration of different houseflies strains. Are there two strains that differ significantly from each other in the length of the developmental period (*text*). Plot the simultaneous confidence intervals for all pairwise differences (pdf 9×13 *cm*; *2pt*).
- **d)** Draw three boxplots side by side to compare the duration of the developmental among different housefly strains (pdf 9×13 *cm*; *1pt*).

exercise 3 The influence of soil nitrogen content on the growth of two different weed species was studied (data set "growth).

a) Construct a scatter-plot of soil nitrogen content against weight gain. Use a different colour and different plotting symbol for both species (pdf 9×13 *cm*; *2pt*).

b) Make an analysis of covariance to describe the weight gain with increasing nitrogen content for both species. Test whether a model with *a)* different slopes and different intercepts, *b)* one slope and different intercepts, or *c)* one slope and one intercept is needed to describe the response of the two species (*4pt*).

c) Add the regression line(s) you estimated to the plot you have already constructed under **a)** (*2pt*).