Electronic Lecture Note

Statistical Problem Solving in R

Prőhle Tamás + Zempléni András

April 4, 2016

Preface

Our aim with this note is to help our students, taking the courses "Statistical Program Packages" 1-2. We are happy that there is an increasing number of international students coming to our institution. Quite a few of them have interest in statistics, and these courses are especially useful for an application-oriented curriculum. Here we mostly use the **R** program, this motivates our choice for the computing background of this note. There are quite a few tutorials available on **R** as well as textbooks on statistics, so we decided to write a lecture note, which may serve both purposes: it is an introduction to the most important applied statistical methodology and a practical manual for the programming language **R**, without being too technical. The target group includes advanced undergraduate students with a basic understanding of probability and those graduate students, who need applied statistics to their work.

The note consists of five sections: after this brief introduction, the first section gives an insight into **R**, focusing on the practical issues, needed for its successful usage. The second section focuses on the descriptive statistics, necessarily a first step in data analysis. Next, we present the basic theory of hypothesis testing and show the most important statistical tests, including both parametric and nonparametric methods. The fourth section contains the important task of regression modelling. These linear models are an important part of many applied statistical projects. Finally we give a short introduction to the Analysis of variance, which is formally also a linear model, but it has so many unique features that it could have filled not only a short section, but a whole book. We close the lecture note by a list of references.

Acknowledgement

The preparation of this lecture note was supported by the Higher Education Restructuring Fund allocated to ELTE by the Hungarian Government.

I. The **R** programming language

Introduction

The programming and statistical package **R** has by now a history of over 20 years. The reason for its short name can also be found at the beginning. One of the first statistical packages was called S (obviously coming from Statistics). As it has been developed by a profit-oriented company, one had to pay annual licence-fee for it. These amounts were considered as being too high by some university professors and its relative inflexibility also motivated the creation of another, continuously developed package, which contains all of the most recent methods. The needed syntax was very similar to that of S, this was also emphasized by the name. The success of this initiative was probably a surprise even for the organisers: nowadays almost every researcher in mathematical statistics uses this language. The number of libraries, which have to fulfill strict standards, is over 7500!

The use of **R** looks more complicated as the usual menu-driven programs (like SPSS – which is especially popular in the social sciences). But those have also to be programmed, if similar analyses are to be repeated. However, if we consider the fact that here we speak about a completely functional programming language, then its use is not more complicated than any of its similar rivals. But it has an enormous advantage, namely that the statistical functions are readily available and it is easy to process their results, being standard objects in the **R** environment. In most of the cases we need just a few simple commands to apply these programs to our data sets. We show these simple methods in this lecture note.

R uses the GNU program-license (GPL) together with its approach. It means that in most cases the programs written in R can be used even for programs that are sold (Lesser GPL). There are some libraries, however, which pose stricter conditions and can only be used to free programs. One can find out the conditions from the license of the given package. The legal rights are at the foundation 'The **R** Foundation for Statistical Computing ©'. The official informations are available at the webpage `www.r-project.org`. Here the complete source code of the program as well as a vast number of handbooks written in different languages can also be found.

– —— –

The structure and application of the R environment

1.1. Installation of **R**

One can download the latest version of **R** from the webpage `www.r-project.org`, but the earlier versions can be found here as well. The basic system is contained in an .exe file, which installs the system to our computer without downloading further informations from the internet. This base system should definitely be enough during the first steps of learning **R**. However, we shall se that even the installation of add-on packages in a very simple task.

1.2. An introductory sequence of commands

We start the introduction to the package **R** with the overview of some basic commands.

On the webpage `https://cran.r-project.org/` one can choose '`Help » Manuals (in PDF) » An introduction to R`' to get to a very useful, approximately 100-pages introduction of **R** It can be opened directly in **R** by the `RShowDoc("R-intro")` command.

There is a three-page commented sequence of commands in `R-intro.pdf`, starting on page 84 (numbered as 78). If we study this part, we get a comprehensive impression on the logic of the program. Unfortunately to understand this presentation, one needs quite deep mathematical-statistical knowledge.
As with the writing of this note we had in mind such students who intend to get deeper statistical knowledge with its help, before starting the statistics-part, we give such an introduction to **R**, which needs less professional knowledge and gives insight into some more technical details without bothering with a systematic but lengthy syntactic introduction.

The presentation will contain the following parts:

1.2.a Starting and ending an **R** session

Under Windows operating system, after the usual setup, an icon, forming traditionally a large blue letter "R", is created. It refers to the program `Rgui.exe` within the actual version. Let us start it!

After the first lines, containing some introductory informations, a red '`>`' sign appears. It shows that the graphic interface of our program waits for a command.

If we type an arbitrary character sequence after the sign '`>`', and let it run with the help of the key `[Enter]` then the system evaluates and runs the character sequence. The textual part of the result will be shown in the command window, the graphical part in a separate sub-window.

If we *'succeeded'* to type such a sequence of characters, that cannot be understood as a complete sequence — like '`abc(`', — then we get no message, but a *'continue'* prompt, i.e. a '`+`' sign. We can get rid of it by pushing the key '`[ESC]`' which results the system to go back to its baseline state, giving a '`>`' sign, i.e. an *'ordinary'* prompt.

The character sequences which have been typed earlier, can be recalled by the up-arrow key. They may be edited and run by the key `[Enter]` again.

Having started a new program, the most important question is: How can one exit? What determines the status of the environment at the start and at the end? One can exit the system either through the menu by '`File » Exit`' ot by typing the command `q()` (followed by `[Enter]`). Both ways activate the same exiting process and before the actual closing of the program we get the question in an internal window:

```
Save workspace image?
```

The question refers that whether do we wish that at the next call of the system, the same variables, commands should be actual, and be ready to use by the up-arrow key as they were at the moment of the exit. If we answer the question with `yes`, then the system writes a file (`.Rhistory`) containing the last typed commands and another file (`.RData`) with the values of the actual variables and functions. The system at the start investigates whether these files can be found. If yes, then their content becomes actual: the data and the functions are loaded into the working space and the commands in `.Rhistory` become ready to use by the up-arrow key, so we can continue our work in the same environment as it was at the time of the last exit.
We may avoid the `Save...` question, if we write the desired answer as the argument of the `q()` function: as `q("yes")` or `q("no")`.
`q` is actually the abbreviation of the command `quit()`, which can be seen if we type `q` and push the `Tab` key, then it becomes visible, which further calls are available starting with `q` at the given state of the system.

It is very practical to set the working directory as we wish. In order to do so, let us first create a directory `C:/TMP`! We may either set the properties of the **R** icon so that the directory `C:/TMP` should be the working directory of **R** at the start, or we may type the command `setwd "C:/TMP"`. Either way, when typing the

```
getwd()
```

command, we get the

```
[1] "C:/TMP"
```

line as answer.

There is an important point to be mentioned here. The distinction between data and commands (functions) is based on the simple observation that commands are always followed by an opening bracket. That is why we have not got the continuation prompt after `‘abc(’` as it was waiting for the closing bracket `‘)’`. Should we write the sequence `‘abc()’`, then we get

```
Error: could not find function "abc"
```

as the sequence `‘abc’` was considered as a potential function. If we give `‘abc’` without any brackets, then the error message would be

```
Error: object ‘abc’ not found
```

as `‘abc’` was considered as a data base.

———

Now let us show, how can the system be used e.g. for simple calculations. Let us type and ‘run’ by `[Enter]` the string `3^2+4^2` The command and the result is as follows:

```
> 3^2+4^2
[1] 25
>
```

So the system calculates the value of the given formula and writes the result to the screen, and gives another *prompt*, waiting for the next command.

The output `[1]` before the actual result is due to the fact that the result is part of a structure, where the different parts are numbered. Here the result consists of a single part, which is of course the first.

Let us type and ‘run’ with `[Enter]` the following strings one-by-one: `‘X<-2’`, `‘X’` and `‘(X<-3)’`!
The result is the following:

```
> X<-2
> X
[1] 2
> (X<-3)
[1] 3
>
```

With this we have demonstrated how to create variables in **R**. The first command created the object named `X`, which has got the value `2` by the two-character combination `‘<-’`. The second line confirms that `X` was indeed created and its value has become `2`. In the third command we gave a value to the variable `X` again. This command shows two things: first that we can overwrite the values of a given variable without any problems, and that the brackets do not have any effect on the output.

Now let us type to the command line the following: `(X<-NULL);ls();rm(X);ls()`! After the `[Enter]` The result on the screen is the following:

```
> (X<-NULL);ls();rm(X);ls()
NULL
[1] "X"
character(0)
>
```

We have written four commands in a line, separated by the semicolon character `‘;’`.

The first command gives to `X` the special `NULL` value. It is important to note that **R** distinguishes between

the lower- and the uppercase letters! 'NULL' has to be written capitalized! The result is the empty object, named "X", as it is seen from the output 'NULL'. The next line of the result is the answer to the command ls(), which is the list of all, actually existing objects. To the third command there is no output. It just deletes the variable X from the working space. The character(0) result of the last command shows that after the deletion of X there were no variables left in the working space.

Now we show the behaviour of the logical variables.

```
> X<-3;Y<-4;
> X^2+Y^2==25
[1] TRUE
> c(TRUE,FALSE,T,F)
[1] TRUE FALSE TRUE FALSE
> (F<-1);FALSE<-1
[1] 1
Error in FALSE <- 1 : invalid ...
> ls()
[1] "F" "X" "Y"
> rm(list=ls())
> F
[1] FALSE
>
```

In the first command we see a double equation sign, '=='. It tests if the two sides of the equation are equal or not. In our case the commands in the first line ensure that the value of x^2+y^2 is indeed 25. So the result of the test is TRUE. Should we have written 24 instead of 25 the result would be FALSE.

One has to be careful: the simple equation sign would be essentially equivalent to assigning a value: ' Z = X^2 + Y^2 ' would assign a value to Z.

The second command creates and prints a logical vector of length 4. Its result shows that TRUE and FALSE are built-in logical constants. But as default, the values of the variables T and F is also TRUE and FALSE, respectively.

The third command line shows that the variables T and F can be overwritten, but TRUE and FALSE can't. The system performs the first command, but the second one results in an error message.

The result of the fourth command shows that besides the variables X and Y another new F variable was created, which has value 1.

The last command line is again a rm() command. But this case its arguments re not variables rm(X,Y), but the list parameter of the rm() was assigned the value ls(), which resulted in the deletion of all the three variables.

The fact that the variables were deleted, is shown by the result of the last line. Here we see that the variable F has got back its original value, since we have deleted the variable, which covered it. So we have to be very careful when using these abbreviated variables T and F for logical purposes.

Typing the question mark activates the help function. So ?"==" gives that the negation sign is !, i.e. the exclamation mark. By the binary operation "!=" we may test the non-equality of the expressions on the two sides. But the results of !TRUE is FALSE.

1.2.b Arithmetics, receiving help and examples

Assigning values

For assigning values we have always used the two-character code "left arrow" <-. In theory it is also possible to give values to the right by the "right arrow" ->. It looks as the double arrows <<- and ->> have seemingly the same effect. Even more it looks as the equality sign = does just the same.

But it is not completely true!

The original meaning of the equality sign in **R** is assigning values to parameters. We have already seen this meaning in the command rm(list=ls()).

Now it is acceptable in the core functions of **R**, but there are minor differences, which may be affecting more sophisticated programs – not covered in this introductory note.

However, there are minor differences in the effect of these commands. The `<<-` and `->>` double arrows define values not only in the actual evaluation environment (like the simple arrows), but also in the calling environment. This can be seen best when defining functions as follows. In the next program segment we define a function `tt` by two different codes. First observe the definition itself: it has to be started by a definition: `tt<-function(t)` followed by the actual definition in curly brackets. The last command `return(y)` is the actual result of the function.

```
x<-y<-NULL

tt<-function(t) {x<-2;y<-x+t; return(y)};
x<-3;rm(y);tt(4);x;y

tt<-function(t) {y<<-x+t; return(y)};
x<-3;tt(4);y
```

The result of the first call:

```
[1] 6
[1] 3
Error:  object 'y' not found
```

We see that the function uses its 'own' value `x` so `6` is returned. However the assignment `<-` does not change the existing value of the variable `x` and does not define a variable `y` in the calling environment either.

By the definition of the second function together with its calling environment:

```
x<-3;rm(y);tt(4);y
```

we get:

```
[1] 7
[1] 7
```

Which shows that the function 'sees' the `x` variable of the calling environment and by the value given by the double arrow, a variable `y` has also been created, taking the value from the function environment.

Summarising

- When a function is called in **R**, the system creates an internal environment, which is deleted after the commands were run.

-The commands of the functions are first evaluated within this internal environment, but if needed, it can use values from the calling environment as well.

- Information to the calling environment can only be transferred by the double arrows and the `return()` command.

- This returned value can be used for giving values to variables.

We shall use the left arrow `<-` for giving values, following a 'traditional' approach. The use of the 'arrow' as the value assigning code has also historical reasons, but its use has a practical meaning as well:
so the value giving `<-`
the parameter assignment `=` and
relation code `==` can easily be distinguished.

Special constants

`NA` (i.e. Not Available): missing value
`NaN`=0/0 (k.e. Not a Number): not defined
`Inf`=1/0 : infinte

Interesting operation signs

`x %% y` the residual of a division (`x mod y`)
`x %/% y` the integer part of a division
Thus the result of the next sequence of calculation is `TRUE` for any `x` and `y`:

```
x<-4;y<-3;((x%/%y)*y+x%%y)==x
```

Vectors are the simplest data structures

Until now we have used variables that contain a single number, except in the case when we have used the concatenate command `c(x,y)`. This `c()` concatenate function joins the two variables, resulting in a *vector* of length `2`. The length of such joined elements can be longer as well:

```
x <- 1;y<-2;
z <- c(x,y)
length(z)
length(c(x,y,z))
```

As the result of the second last row shows, `z` is a vector of length `2`, while the length of the concatenated

vector of the last row is 4.

There are much more convenient tools for creating longer vectors.

The ‘:’ where the ’:’ is between two numbers can be understood as an arithmetic sequence of numbers, where the difference is 1, the starting number is the one on the left hand side and the last one is the last number, which falls within the interval between the two given numbers. So 1:3 means the sequence c(1,2,3), 3:1 the sequence c(3,2,1), and .5:pi is the same as c(.5,1.5,2.5) as pi has the default value 3.14159.

Using ‘:’ we have to take care that the evaluation of : precedes the evaluation of the basic arithmetic operations, so 1:5+4 is a sequence of numbers from 5 to 9, while 1:4/2=c(.5,1,1.5,2). On the othe side 1:3^2 has 9 elements, and -1:3==c(-1,0,1,2,3) but 2-1:3==c(1,0,-1).

Another, more sophisticated tool for creating sequences is the use of the function seq(). Here one can define as parameters the step size and the length of the sequence to be created. The next four lines define the same sequence:

```
seq(2,4)/2
seq(1,2,by=.5)
seq(1,2,length=3)
seq(1,2,along=1:pi)
```

The operations for vectors are to be understood elementwise. We can see this for the example 1:5+4 E.g. the value of 1:4+2:5 is the same as the vector c(3,5,7,9).

Logical operations

In **R** ‘and’ is denoted by ‘&’, and ‘or’ by the sign ‘|’

Zero is the only number with logical value FALSE all the other numbers have value TRUE. The result of the logical operation 1:3==3:1 is a logical vector c(FALSE,TRUE,FALSE). If we want to get that the vector 1:3 is elementwise equal to the vector c(1,3), then we can see it e.g. by applying the function all(): all(1:3==c(1,2,3)). It can be reached also by the operation ‘&&’: (1:3==c(1,2,3))&&TRUE (if we have used the single &, then the result would be a vector of TRUE having length 3). We can see that among the elements of 1:3 and 3:1 there are equals, by the command any(1:3==3:1).

But we have to be careful! If we do calculations with two vecors and the vectors are *of different length*, then the system repeats the shorter vector until a vector of the same length as the longer is got.

This means that the result of the function 1:10-c(1,2) is the same as rep(seq(0,8,by=2),each=2), i.e. c(0,0,2,2,4,4,6,6,8,8). Here we have used another function, very useful for creating vectors, namely the function rep().

The above-mentioned cyclic addition is logical if we want to subtract e.g. 1 from every element o a vector: c(7,3,5)-1 is the same as c(6,2,4). But the command c(6,2,4)-c(1,2) is not correct, as the length of the longer vector is not an integer times the length of the shorter vector. We get a warning: (’Warning message:...’), but the value of the result is the vector c(5,0,3).

Ordering: sort(), order() and rank()

```
v<-c(.3,.1,.5,1);
sort(v)
(s<-sort(v,index=TRUE))
v[s$ix]
(ii <- order(x <- c(1,1,3:1,1:4,3),
             y <- c(9,9:1),
             z <- c(2,1:9)  )      )
cbind(x,y,z)[ii,]
rank(c(7,4,3,5))
```

The default value of sort() is the ordered vector of the elements of its argument vector index=FALSE

If we call the function sort() with the parametrization index=TRUE then the result is a list of two elements. The upper element of it, $x, is the sorted vector, while the second is $ix, an index vector, which gives the number of the elements of the sorted vector in the original one. This means that the original v vector indexed by s$ix: v[s$ix] is the same as sort(v).

The result of the order() shows that this command is just a lexicographic ordering. In our example it gives that the 3 vectors of length 10 arranged in columns, and reading the resulted matrix *row-wise*,

which is the first, the second etc. from the `10` rows of length `3`.

The result of `rank` is a permutation. It gives that what is the rank of the elements of the original vector after the ordering.

Requesting help

We can get immediate help by the function `help()` or the question mark '`?`'. So e.g. for the function `seq()` we may get help by calling `help(seq)`, or `help("seq")`, possibly `?seq`.

We must note that on the upper-left corner of the information sheet we got, the text `seq(base)` is to be seen. This is slightly misleading, as "base" does not refer to the possible parametrization, but for the fact that the given function belongs to the `base` package.

Consulting e.g. the help page of `seq()` it can be seen that it consists of parts, which are standard within **R**: Description // Usage // Arguments // Details // Value // References // See Also // Examples. The last part is especially useful, as these commands can be tried not only using copy-paste option, but also simply using the function `example()`.

More general help may be got by the command `help.start()`. This activates the default web browser with a page, where the same handbooks can be reached in .pdf format, as from the Help button of the Rgui menu line. Continuing along "Packages" we get a list of all the extensions which are installed under the given version of **R**.

Coincindence of names

In the next sequence of commands we show an example, which can easily be realized during programming. We define a variable `cor` which has the same name as a function.

```
find("cor")
cor<-1:pi
find("cor")# ".GlobalEnv" "package:stats"
find("cor",mode="function")
find("cor",mode="integer")
cor[2];cor(1:3,3:1)
cor<-function(x,y) return(sum(x^2+y^2))
find("cor",mode="function")
cor(1:3,3:1)
rm(cor)
find("cor")
cor(1:3,3:1)
```

The `".GlobalEnv"` denotes the base environment of "Rgui.exe". The value of `cor[2]` is `2`. The value of `cor(1:3,3:1)` is `-1`, as the correlation between the vectors is -1 due to the second being a linear transform of the first, with negative steepness.

In the next part we have rewritten `cor` again, this time with a function. Now calling `cor` it is not clear, which one we have in mind. **R** chooses the one, which is first found, i.e. `.GlobalEnv`. This implies that the result of the call `cor(1:3,3:1)` is `28`. In this case if we want to call the `cor` function of package `stats`, then we must state explicitly that we need the one from package `stats` by `stats::cor(1:3,3:1)`.

If we use three colons '`:::`' instead of two, then the expression `pkg:::name` returns the value of the internal variable name in package `pkg` if the package has a name space.

The '`datasets`' package contains the most important demo data sets for the system. We can get a list of these and data sets from given packages as follows:

```
data() # every data set
data(package="cluster")
```

Examples, i.e. the `demo()`

This is left as last – but we may have started with it –: the demo materials of the base system and other packages.

```
demo()
demo(smooth)
demo(graphics)
```

The command `demo()` shows those demonstrational programs, that are loaded at the moment. (See e.g.: `search()`, or based on their location in the operating system: `searchpaths()`). The sequence of plots `demo(smooth)` is one of the demonstrational sequence of the package `stats`. The `demo(graphics)` is interesting visually as one of the demo of the package `graphics`.

1.2.c More complex data, figures

Until now, we have seen that one can build vectors from numbers or logical values. The same is true for character type variables as well: `a<-"abc";b<-"def";c(a,b)`, which results in a vector with two components: `"abc"` being the first and `"def"` being the second. If we want to paste them, it can be done by the help of the `paste()` command: `paste("abc","def",sep="")`. Here the parameter `sep` was needed, ensuring that there is no space left between the two character sequences. Thus we have omitted the default space, but we may even change it, like `paste("abc","xyz",sep="(beginning-end)")` .

The substrings of character sequences can be got by the function `substr()`. The result of `substr("abcdef",3,4)` is `"cd"`. Further important functions for manipulating character-valued variables is the `nchar()` giving the length of a character sequence and the `strsplit()`, which allows for cutting the sequence.

Matrices

The matrices are also vectors with respect the definitions of the **R** system. More exactly such vectors, which have a `dim` attribute, which can be declared and be handled as follows.

```
(M<-matrix(1:6,2))
class(M)
attributes(M)
str(M)

A<-1:6
attributes(A)$dim<-c(2,3)
class(A)
str(A)
list
```

The code for matrix-multiplication is `%*%`. The common multiplication code `*` means the element-wise product.

```
(M<-matrix(c(1,3,2,5),2))
(x<-matrix(1:2,2))
M%*%x
solve(M)
(I<-solve(M)%*%M)
round(I,5)
round(I-diag(1,2),5)
```

The function `solve()` computes the inverse of a matrix. The result of the product $\texttt{M}^{-1}\texttt{M}$ shows that we get identity up to an error of 10^{-16}. The result of the `diag(c(1,2,3),3)` function call is a 3×3 diagonal matrix with the vector `c(1,2,3)` in its diagonal. The example shows that the result of `diag(1,2)` is a 2×2 identity matrix.

Elements or submatrices of the matrix can be subtracted. One may get new matrices from the elements, as the following example shows.

```
(M<-matrix(1:12,3))# 3x4-row-wise
M[2,3] # this is 8
M[2:3,c(1,3,4)]  # 2x3
M[-1,-2] # the same with deletion
M[c(2,2),c(2,2,1)] # this is 2x3 as well
```

Matrices may be 1 or 2 dimensional. But be careful:

```
(M<-matrix(1:12,3))
M[,1] # this is not a matrix
M[,1,drop=FALSE] # while this is one...
```

So the submatrix, if it has dimension 1 in one direction, looses its matrix attribute. This can be hindered by applying the '`drop=FALSE`' option. The same is true for the 'array' type variables.

The columns, rows, or even its elements can be named, as the following commands show.

```
M<-cbind(a=1:3, pi=pi)
class(M)
str(M)
dimnames(M)
dimnames(M)[[1]]<-c("a","b","c")
dimnames(M)
names(M)<-c("A","B","C","D","E","F")
M
M["A"]  # an element
M["a",] # a row
M[,"a"] # a column
rm(M)
```

A matrix can be built column-wise by the command `cbind`. The same for rows is achieved by the command `rbind`. We may get the size of a matrix on different methods:

```
(M<-rbind(1:3,1));dim(M)
nrow(M);ncol(M);dim(M)[2]
```

Arrays

A `array(1:120,c(2,3,4,5))` creates a 4 dimensional array, consisting of 120 elements. These arrays can be added, pasted and used in other operations like matrices.

A special operation for arrays is the outer product `%o%` The result is an array, which has as dimension the sum of the dimensions of the original arrays. The elements are simply the products of each element of the first array with each element of the second array.

```
A<-c(1:3)%o%array(1:24,c(2,3,4))
dim(A);class(A)
B<-c(1:3)%o%c(1:24)
dim(B);class(B)
attributes(A);attributes(B)
```

"A" is an array of $1 + 3 = 4$ dimensions, while "B" is a matrix.

Most of the calculations are done with matrices. However from the statistical analytical point of view, the so-called '*data.frame*'s are more important. These are actually '*list*'s with special format. The '*list*'s have several similar aspects as vectors (or matrices).

Variables with a structure 'list'

The *list* is like a long rod, for which – at any point – an arbitrary variable (or even a new list) can be hung.

```
a<-3;bb<-c(1,3,4);M<-matrix(1:8,2)
(Li<-list(a,bb,M))
(L<-list(A=a,B=bb,XX=M,Utolso=Li))
```

The second list differs from the first that its elements can be reached by names as well.

```
L[1];L[c(3,1)];
L[[1]]
class(L[[1]]);class(L[[4]]);
L$A;class(L$A)
attributes(Li);attributes(L)
```

We can see that one can refer to a given element of the list by the double bracket '`[[`' command. The single bracket '`[`' defines just a part of the list.

Data having the list attribute are important in **R**, because the result of the functions in most of the cases is a special list. This is not obvious for the first glance, as the result is fed to the `print()` function, which has different output methods for different types.

The following is a simple example for this:

```
D<-density(c(1,1,2))
D
class(D)
attributes(D)
str(D)
as.numeric(D)
D;fivenum(D$x);fivenum(D$y)
plot(D$x,D$y)
plot(D)
```

The function `density()` calculates an estimate for the continuous density of the sample, based on 3 observations. It is purely technical, such a small sample size cannot have any statistical meaning. But it is a good example for investigating the structure of the result. `print.density()` gives just the basic statistics for the two variables, upon which the graph of the estimated density is based. The last two commands show that we get essentially the same figure based on the two coordinates of the result D as the class-assigned `plot.density()`.

The format '*data.frame*' differs from the format '*list*' only in that respect that all of the elements are '*vector*'s of the same length .

Operations with data frames

The following operations are the most common ones.
Picking rows of the data.frames
joining two data.frames by given key,
calculating statistics for given groups of rows.

Their realization in **R**:

```
subset.data.frame
merge.data.frame
aggregate.data.frame
```

The command `subset(x, subset, select)` helps to choose the rows we want to keep and at the same time to delete those which are not needed. This operation is based on a logical vector, corresponding to the rows of the data.frame.

The command `merge(x,y,...)` is suitable to joining two data frames by a common key. It can be suitably parametrized with respect to the joined rows (e.g. are partially filled rows allowed etc.).

The command `aggregate()` is very useful for calculating statistics for groups of rows, defined by different values of a grouping variable.

If the question is whether elements of a vector can be found among the elements of another vector, then we may get a logical vector containing the answer by the operation `%in%`. For example the result of the `c(1:2)%in% 2:12` command is a `c(FALSE,TRUE)` vector.

Drawing tools

The `plot()` command draws the data given as its arguments into a new graphical window, as determined by its additional parameters.
The result of the next sequence of commands is a plot, consisting of three red points.

```
x <- c(1,2,3);y <- c(2,5,4);
plot(x,y,cex=2,pch=19,col="red")
```

In this parametrization `cex` defines the size of the dots and `pch` defines the character to be plotted. The limits of the x and y axes are taken from the data. But it may be unsuitable if we add a broken line or new points.

```
y2 <- c(2,7,4);
y3 <- c(3,1,3);
lines(x,y2,lw=5,col="green")
points(x,y3,cex=2,pch=19,col="blue")
```

In these cases it may be needed to set these limits beforehand:

```
plot(x,y,cex=2,pch=19,col="red",
            xlim=c(0,4),ylim=c(0,8))
lines(x,y2,lw=5,col="green")
points(x,y3,cex=2,pch=19,col="blue")
```

In the simplest case, if we want to plot a graph of a function, we may use a command like this: `plot(sin)`. The only problem here is that the values of the x axis are within $[0, 1]$. The following parametrization gives a result we expect, knowing the form of the function $\sin(x)$.

```
plot(sin,xlim=c(-4*pi,8*pi))
abline(h=0);abline(v=0)
```

If we have a thorough look at the figure, we may observe that the curve is slightly rugged. It is caused by the too few points, upon which the graph is based. We can change the number of points as follows:

```
x  <- seq(-8*pi,18*pi,l=1001)
x2 <- seq(-8*pi,18*pi,by=13*pi/6)
x3 <- seq(-8*pi,18*pi,by=11*pi/6)
par(mfcol=c(3,1))
plot(x,sin(x),t="l")
plot(x2,sin(x2),t="l")
plot(x3,sin(x3),t="l")
```

All three curves are similar to a sine function, but it can be seen that under a wrong sampling frequency neither the frequency nor the phase will be found.

Further parameters of the drawing window can be searched in the list, got by the command `par()`.

Further commands related to the drawing

The following commands do not open a new graphical environment (figure), just paste the informations based on their parameters.

`points`	points
`abline`	a line with given parameters
`lines`	broken line
`segments`	line segments
`arrows`	arrows
`polygon`	given by its vertices
`axis`	axes
`title`	title
`mtext`	text wrritten on the margins
`legend`	legend
`text`	writing of text

The next sequence of commands is a simple game showing the use of the `locator()` routine:

```
x <- rnorm(9); y <- rnorm(9)
target <- cbind(x,y)
T1 <- "Very good (%.4f) !"
T2<-paste("More accurately and quicker",
                              "(%.4f) !!")
```

```
par(mar=c(1,1,1,1))
plot(target,pch=49:58,col="red",
                  axes=FALSE,frame=TRUE)
title("Locate the 9 points!")
z <- Sys.time();
p <- locator(n=9,type="p")
arrows(x,y,p$x,p$y,length = .1)
span <- as.numeric( Sys.time()-z )
dpt <- sqrt(
    sum((cbind(p$x,p$y)-target)^2))/span
dx <- sum((x-p$x)^2);
dy <- sum((y-p$y)^2);
s <- if(dx<dy) 1 else 4
sprintf(if(dpt<.02) T1 else T2, dpt)
mtext("This direction is worse!!!",s=s)
```

In the game three drawing functions were used: `title()`, `locator()`, `mtext()`.

```
locator(9,type = "p") # points
locator(9,type = "l") # connecting line
locator(9,type = "o") # line+points
```

In our case the `locator()` command draws a point where the mouse-click indicates and then the accurateness of the guess is calculated.

Several figures at once

It is the simplest way to use the command `par()` for viewing more figures at the same time. By resetting its `mfrow` (multi frame by row) parameter e.g. as `par(mfrow=c(2,3))`, then we get a graphical window which is divided into 2 rows and 3 columns.

```
par(mfrow=c(2,3))
plot(runif(2),runif(2),
pch=19,cex=75,col=c("red","blue"))
```

Let us run the plot command several times! The first figure is drawn into the upper left corner. The next one continues the first row, then comes the second row etc. If the last row is completed, then it starts drawing into the first window of the first row again. If we delete the graphical window, the multi-window setup is deleted as well. We can change back to the single-window mood by the command `par(mfcol=c(1,1))`.

We may use another method for splitting the graphical window. The following code splits the graphical window into 4 parts of equal size and plots the same points into them, using different colours.

```
x<-rnorm(6);y<-rnorm(6)
split.screen(c(2, 2))
screen(1);plot(x,y,pch=19,col="red")
screen(2);plot(x,y,pch=19,col="blue")
screen(3);plot(x,y,pch=19,col="green")
screen(4);plot(x,y,pch=19,col="black")
```

`split.screen` can also be used for creating windows of different sizes. If its parameter is a matrix having 4 columns, then the rows give the places of the corners for the sub-windows (left, right, lower, upper), calculated as proportions, compared to the whole window:

```
split.screen(matrix(c(0,.75,0,.75,
            .25,1,.25,1),2,byrow=TRUE))
screen(1);plot(x,y,pch=19,col="red")
screen(2);plot(x,y,pch=19,col="blue")
close.screen(all = TRUE)
```

Here we have created two windows of size 3/4, which overlap in the middle of the screen. The last command exits the split-screen mode.

If we want to draw more figures from a program, then we may wish to achieve that the next figure overwrites the previous one only after we let it to be done (by pushing a key for example). The `getGraphicsEvent()` command can be helpful in these cases.

The command `getGraphicsEvent()` can be made sensitive to three different events: the movement of the mouse, the button of the mouse or a keyboard button.

1.2.d Program organisation

It is worth to organise such combination of commands, which are often used into functions. We have already seen examples for user-defined functions. Its syntax is: '`function(){}`', where the first set of brackets include the arguments, the second the commands. Let us see a simple example:

```
pelda<-function(x,y)
 {a<-x+2;b<-y*2
  return(list(a=a, b=b))}
(z<-pelda(3,4))
args(pelda)
```

Here the result is a list of two element. The first is 'a', having a value of 5, the second is 'b' which is 8.

Next we show two classical cycle-organising commands:

```
repeat ...  break
while(cond) expr
```

The `break` and the `next` option can be part of the `for` cycles as well as the `while` cycles.

As we have seen, one may write several commands into a single line, if they are separated by semicolon. It is also possible to organize the commands into an array, if we write them between curly brackets '`{`' '`}`'.

The `for`-cycle

The `for`-cycle consist of a control enclosed by brackets '`(`' '`)`' and an array of commands enclosed by curly brackets '`{`' '`}`'. The result of the next run is 10! (factorial).

```
f<-1; for(i in 2:10) {f<-f*i} ; f
```

The `for`-cycle is relatively slow in **R**. But there are other methods for arranging our cycles, so if possible the long and nested `for`-cycles should be avoided.

If we want to see the earlier commands, we might get them in a separate window by the `history(m=99)` command. The needed parts then can be edited and copied from this window.

The work is easier if we use the possibility that after having typed the first characters of a command, we push the 'Tab' button twice:
 - if the extension is unanonymous, then the command is completed
 - if there are more possible completions, then the possible variants are printed.

If we intend to run a longer program several times, then the best method is to type commands into an ASCII file and call it from there by `source("file name")` to the command window of `Rgui`. The `source()` command can also be a part of another script file.

We may call a function repeatedly without any changes via the `replicate()` command. It has two parameters: the second is the name of the called function and the first is the number of repetitions. Its result is a vector, containing the results of the function call.

E.g. the command
`replicate(100, mean(rexp(10)))`
calls 100 times the average function for 10 random numbers, having unit exponential distribution. Its result is a 100-element vector.

Conditional commands

There are two ways to construct such commands which are executed only in case of given conditions.

The command `if (test) {yes} else {no}` can be applied in simpler cases. Here `yes` and `no` can be an array of commands.

The command `ifelse(test,yes,no)` is slightly more complicated. In a typical case the three parameters and the result is four vectors of the same length. The system determines the result by row-wise evaluation of the three parameters The result in a row will be equal to the `yes` or `no` expression, depending on the value of the `test` in a given row.

The command `switch()` operates similarly to `ifelse()` in case of a single parameter. The difference is that by the help of the `switch()` we may choose not only from two, but an arbitrary number of labelled function values.

The functions, we can choose from are given as the second, third, etc. labelled parameters of `switch()`. The question, which function will be evaluated is decided by the first parameter. The system calls the function, which has identical label to the value of the first parameter.

1.2.e Input and output of data

By the `options()` command we can list the basic settings of our system. A more important one is `getOption('digits')`. It shows that the system prints the numbers with seven digits. Its operation is shown by the constants π and e:

```
options()$di# 7
pi# 3.141593
options(digits=19)
options()$di# 19
pi# 3.1415927 (it has just 18 digits)
(pi- 3.141592653589793)*10^17# as stored
exp(1)# e=2.718282
options(digits=7);pi
```

It was enough to write the first two characters of the needed component of the `options()` list, as it gave a unique prefix in this case. Unfortunately there is no built-in constant e but it was substituted by `exp(1)`.

Here are the special elements of the character strings:

`\'` single quote
`\"` double quote
`\n` newline
`\r` carriage return
`\t` tab character
`\b` backspace
`\a` bell
`\f` form feed
`\v` vertical tab
`\\` backslash itself
`\nnn` a character given by an octadecimal code, the same for hexadecimal `\xnn` and for unicode `\unnnn` or `\Unnnnnnnn`.

If we intend to print from the program, then it is not enough to simply give the name of the variable, as the following example shows.

If we type into the file `rov` of the worikng directory by the following commands:

```
"Not seen!"
("This neither ...");x<-2;x;(x)
print(paste("abc","\n"))
print(paste("bcd","\n"))
cat(paste("def","\n","efg","\n"))
cat(paste("The first printed row...","\n",
          "... the second!\n",sep=""))
```

and let it run by the following command: `source("fv.rov")`, the result is the following:

```
> source("fv.rov")
[1] "abc \n"
[1] "bcd \n"
def
 efg
The first printed row...
 ... the second!
>
```

Note that the command `print()` prints the informations, but the editing commands are not used. The command `cat()` produces output as expected.

Handling external data

R can work with different input formats and can provide results in several formats by the help of its packages. The direct input and output of the most famous statistical software packages is ensured by the package `foreign`, which belongs to the basic program. Examples: `read.spss`, `write.spss`. Several similar functions are available, like `.S`, `.dta`, `.octave`, `.mtp` and `.systat`.

We present now the three most frequent input/output procedures.

The first is the most effective, the own data format of **R**. The second is the least effective, but which is the easiest to control: the input-output in ASCII format.

The own data format of **R**

One can save data by the `save()` command, and load data by the `load()` command. In the `save()` command first we have to list those variables (separated by comma) which we intend to save, then the `file="..."` parameter has to specify the file (including its directory) where we want the data to be saved.

The `load()` command is simpler. Here only the name of the file has to be given, and there are practically no further parmetrizations possible. This means that we cannot control that the variables to be loaded should not overwrite existing contents. Not having better options, we may utilize the feature of **R** that it is possible to run several of its copies concurrently and load the suspicious file to an "empty" copy of **R** and there we can check its content.

ASCII data

It is typically used for input and output *data.frame* variables:

```
A<-data.frame(e=c(1,3,2),m=2:4,
       row.names=c("A","B","C"))
write.table(A,"C:/TMP/data.csv",sep=',')
D<-read.table("C:/TMP/data.csv",
                  header=TRUE,sep = ",")
  edit(D)
  labels(D)
```

The most important options:
`header = TRUE`
`sep = ","` separating character between columns
`dec="."` the decimal point of the numbers
`quote=TRUE` quotation mark around strings
`stringsAsFactors=TRUE` should those columns, where there are characters, converted to factors? It may cause unpleasant results that the default value here is `TRUE` while quite commonly we may get better results by `FALSE`.

The next sequence of commands show, how can we convert a variable having type *factor* to a variable of character type. The commands in between show, why is the task not obvious.

```
f<-as.factor(c("one","two","one","three",
                    "six","four","six"))
f;class(f)
as.numeric(f)
unclass(f)
levels(f)[as.numeric(f)]
```

The last one is the command, suitable for converting.

Handling `EXCEL` files

There is a purposely written package `xlsx`, which allows the user to read `EXCEL` files. Here we also have to specify, which sheet has to be read, by the `sheetIndex` or `sheetName` parameters.

A possible command for writing an `EXCEL` file:
`write.xlsx(df,"df.xlsx",sheetName="DFrame")`

II. Descriptive statistics

In this part we present some simple methods, useful as a first step in data analysis. Here we use our own data, so not only the scripts, but the results are shown as well.

2.1. Getting acquainted with the data

Introduction

The first step is the input of our data. First we use our own data about the students, which was recorded at a statistics course.

```
diak <-
   read.table("D:\\oktatas
                  \\diakok.txt",header<-T)
```

It contains 47 rows (observations) and 8 variables:

- `maths` (the mathematics note from last semester)
- `law` (the law note from last semester)
- `beer` (the number of bottles drank last week)
- `residence`
- `height (in cm)`
- `shoe size` (European)
- `age (year)`
- `gender`

Of course such self-declared data is not always very reliable, but for our purposes it is just perfect. First let us overview the most important characteristics of the data:

```
summary(diak)
```

The results:

```
     Maths            Law
Min.   :1.000   Min.   :2.000
1st Qu.:1.000   1st Qu.:3.000
Median :2.000   Median :3.000
Mean   :1.957   Mean   :3.404
3rd Qu.:2.000   3rd Qu.:4.000
Max.   :5.000   Max.   :5.000

     Beer                  Residence
Min.   :  0.000   Budapest     :16
1st Qu.:  0.000   Pilisvörösvár: 5
Median :  0.000   Miskolc      : 3
Mean   :  4.213   Baja         : 1
3rd Qu.:  1.500   Bajót        : 1
Max.   :115.000   Békés        : 1
                  (Other)      :20

Height          Shoe size
Min.   :152.0   Min.   :36.00
1st Qu.:165.0   1st Qu.:38.00
Median :170.0   Median :39.00
Mean   :171.8   Mean   :39.64
3rd Qu.:177.5   3rd Qu.:41.00
Max.   :188.0   Max.   :46.00

      Age         Gender
Min.   :18.00   F:35
1st Qu.:19.00   M:12
Median :19.00
Mean   :19.45
3rd Qu.:20.00
Max.   :22.00
```

The notes provide little surprise: maths seems to be much more difficult than law (note that 1 is the worst note - it is a fail - and 5 is the best). Actually, nobody has failed from this subject. The average note from maths is 1.96, while of those from law is 3.4. For beer drinking, the 115 being its maximum value is very surprising. After checking the original questionnaires, it turned out that the error was not in the typesetting (this should always be the first step, seeing such suspicious data). Such so called outliers - as we shall see later - have substantial effect on the results of the data analysis. We have several options for treating such problems: we may omit the most suspicious values, which in this case looks as the joke of one of the students. Or we may think that while the value itself is most likely exaggerated, it is probably not recorded by an abstinent, so we leave it in the data and choose such methods that are not sensitive to these values (they are called robust methods). The mean is heavily influenced by this value, as it became 4.2 (while when omitting this single outlier, it reduces to 1.8). The median is just 0, showing that more than half of the students have not drunk any beer in the given period. This is an excellent example for robust statistics.

Residence is not a numeric variable, its results are character strings, so here we cannot calculate the shown descriptives, only the frequencies can be calculated.

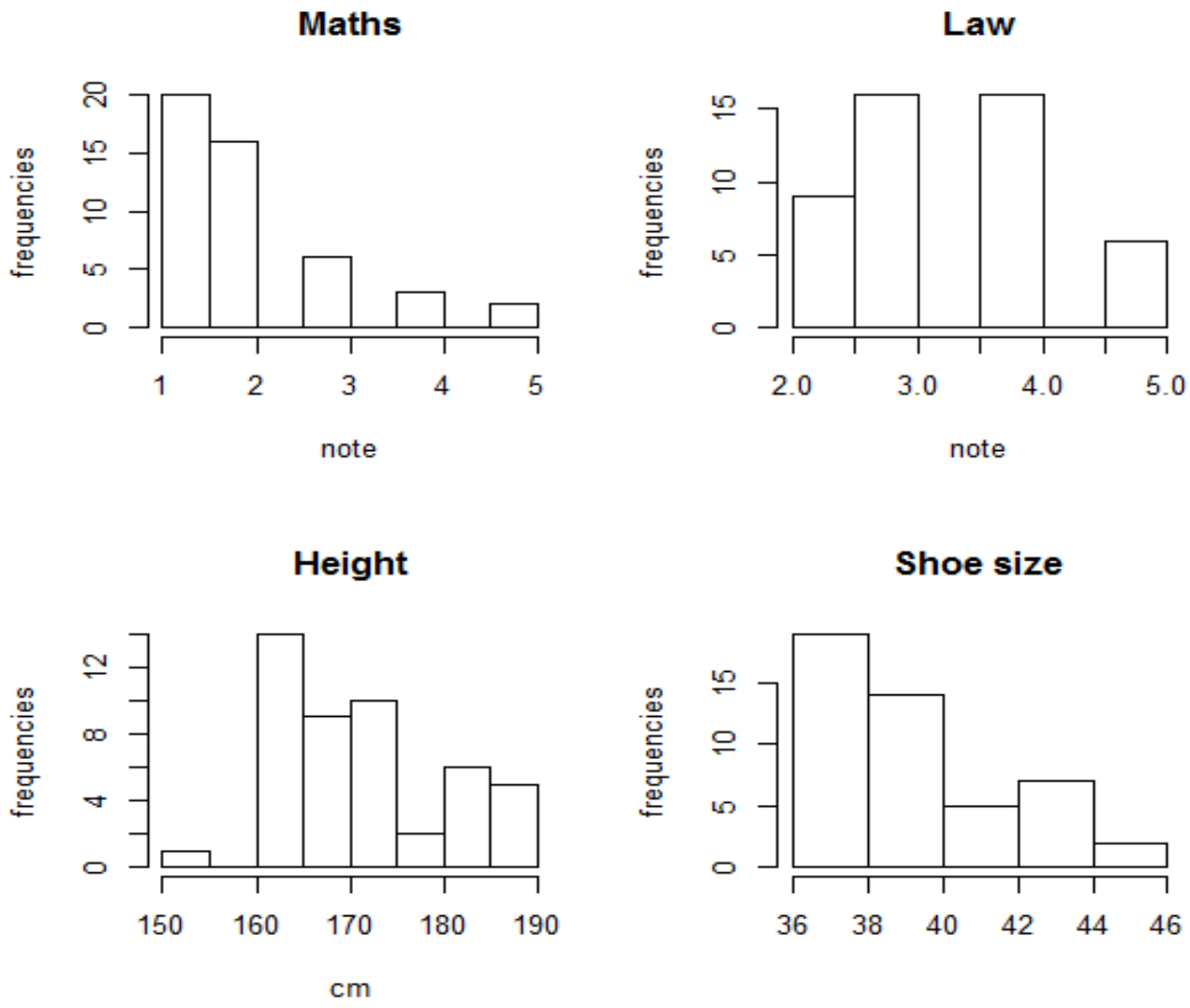
There is also a suspicious value among the height data: for an adult the height of 88 cm is hardly a real value. The control has shown that here a typesetting error has occurred, so we have worked in the sequel with the correct value of 188 cm.

The values of variable "age" seem to be realistic. Gender is again a qualitative variable, which is special, as it has only two values: `F=female`, and `M==male`. This allows us, that in certain cases, when the method is not sensitive to the linear transformation, we may consider it as a binary variable in the analysis. We shall come back to this point in the section about linear regression. It is worth noting that among the 47 elements in the sample, there were only 12 men.

2.2. Graphics

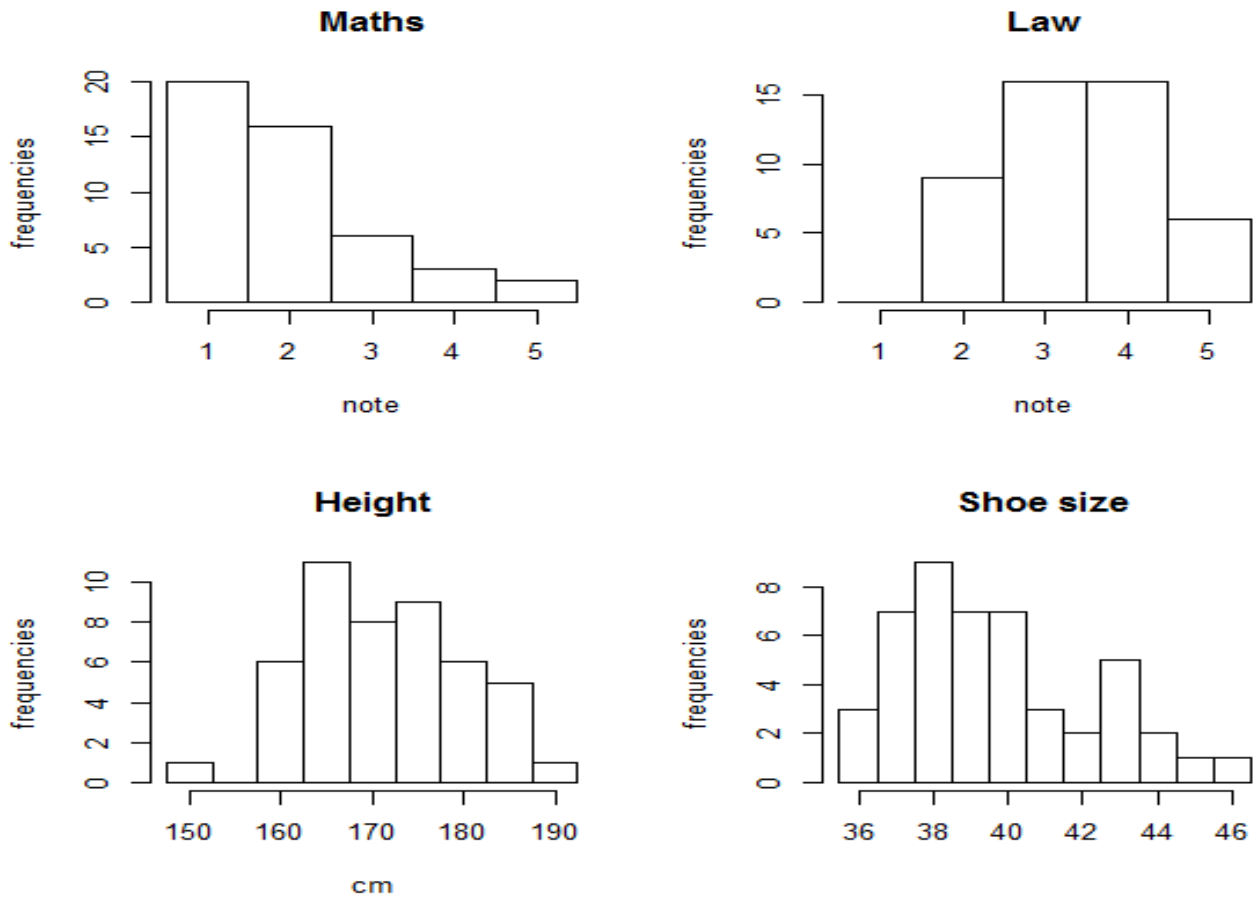
The best method for visualising univariate distributions is the histogram.
This is what the next code produces.

```
par(mfrow=c(2,2))

hist(diak[,1],xlab="note",
    ylab="frequencies",main="Maths")
hist(diak[,2],xlab="note",
    ylab="frequencies",main="Law")
hist(diak[,5],xlab="cm",
    ylab="frequencies",main="Height")
hist(diak[,6],xlab="",
    ylab="frequencies",main="Shoe size")
```



Seeing the results we immediately face the problem that for discrete variables the automatic division of the real line into intervals, produced by **R** is by far not ideal. It is especially problematic at the variable "notes", as one cannot decide, where belong the values, which coincide with the endpoints of the intervals. So after the failure of this first try it is worth setting the limits manually, as follows.

```
par(mfrow=c(2,2))

u1 <- hist(diak[,1],breaks=1:6-.5,
                    xlab="note",
                    ylab="frequencies",
                    main="Maths")

hist(diak[,2],breaks=1:6-.5,
                    xlab="note",
                    ylab="frequencies",
                    main="Law")

hist(diak[,5],breaks=5*1:10+142.5,
                    xlab="cm",
                    ylab="frequencies",
                    main="Height")

hist(diak[,6],breaks=c(1:12)+69/2,
                    xlab="",
                     ylab="frequencies",
                    main="Shoe size")
```



The integers fall here into the middle of the intervals, providing an excellent comparison of the maths and law notes. The histograms show, that especially the maths notes and the shoe sizes have a skewed distribution. The formula for the theoretical skewness is $E((X-EX)^3)/D^3(X)$,which corresponds to the empirical version:

$$\frac{\sum(x_i-\overline{x})^3/n}{(\sum(x_i-\overline{x})^2/n)^{3/2}}.$$

Interestingly, this function is not to be found in the basic **R** package, one has to load the additional library `moments`.

```
> skewness(diak$Maths)
[1] 1.166492
> skewness(diak$Law)
[1] 0.046062
>
```

The difference can be spotted immediately: the value near to 0 for the law-notes means an almost perfect symmetry, visible on the previous figure.

```
> skewness(diak$Shoe)
[1] 0.6532392
> skewness(diak$Height)
[1] 0.2382522
>
```

There is a difference here, but by far not as substantial: both distributions are slightly skewed. Based on this observation we may expect that in all four cases the median will be larger than the mean. But this is not the case as for discrete distributions (especially for those, which have just a few possible values, the median can just follow the changes in distribution with big jumps only (as by its original definition the median is always a real observed value - or possibly the average of these). If needed, it is possible to use a correction formula, suitable for classified observations:

$$m^* = x_l + d\frac{n/2 - f_l}{f} \tag{1}$$

where m^* is the corrected median, x_l is the lower bound of the class of the median, d is the width of this class, n the total number of observations, f_l the cumulated frequency, f the frequency of the class of the median. For example for the maths notes, from

the $u1$ object, which we have got from the last histogram:

```
> u1$counts
[1] 20 16  6  3  2
```

In our example this means that we suppose for the notes that they are in fact rounded values - uniformly distributed in the possible region: this means that a 2 falls between i.5 and 2.5 so in the formula (1) $d = 1$ and $x_l = 1.5$. $n = 47$, $f_l = 20$, $f = 16$, so $m^* = 1.72$, which is in accordance with the property that for positively skewed distributions the median is smaller than the expected value.

Finally, the kurtosis is also suitable for checking the normality (for unimodal, symmetric distributions). Its formula is $E((X - EX)^4)/D^4(X)$, which can be estimated as

$$\frac{\sum(x_i - \overline{x})^4/n}{(\sum(x_i - \overline{x})^2/n)^2}.$$

Its value for the normal distribution is 3. If we get a larger value, then the distribution is peaked (leptokurtic), otherwise it has a more rounded peak.

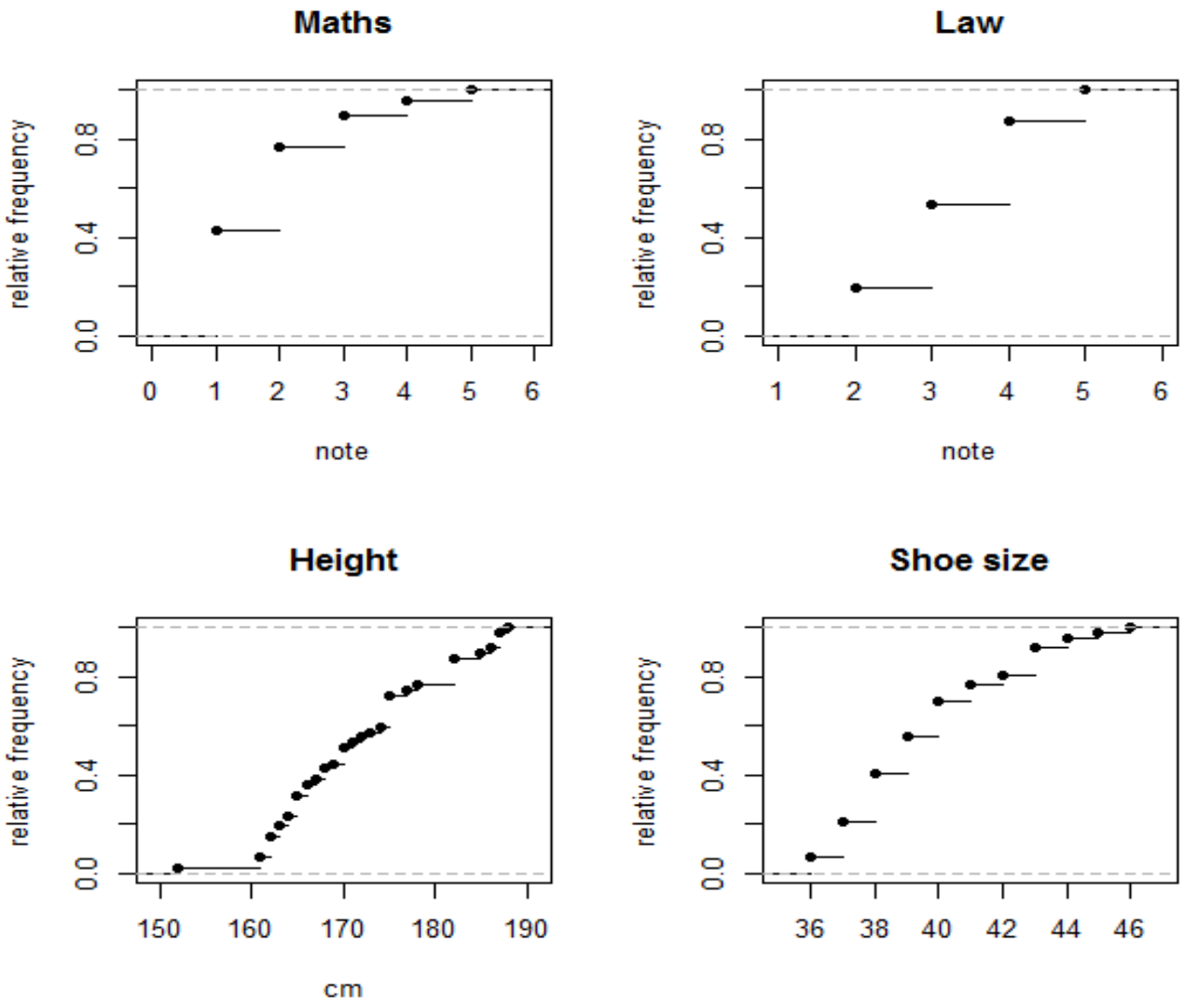
```
> kurtosis(diak$Shoe)
[1] 2.57627
> kurtosis(diak$Height)
[1] 2.20524
>
```

For both cases, the kurtosis is slightly smaller than that of the normal distribution.

If we want to add a hypothesized or fitted distribution to the histogram, then it is needed that we plot the relative frequencies instead of frequencies, ensuring that the total area of the rectangles equal one. Thus the values are comparable to the density function. We shall come back to this possibility at the goodness of fit procedures.

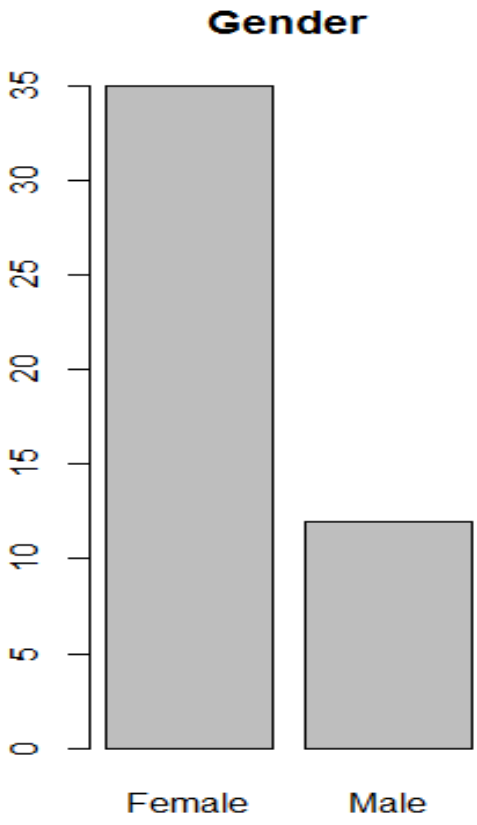
Another informative figure is the empirical distribution function, which has the advantage, that there is no need for binning, like in the case of histograms. In order to get it, we first have to construct the empirical distribution function object, and then it can be plotted, as it is indicated in the next code.

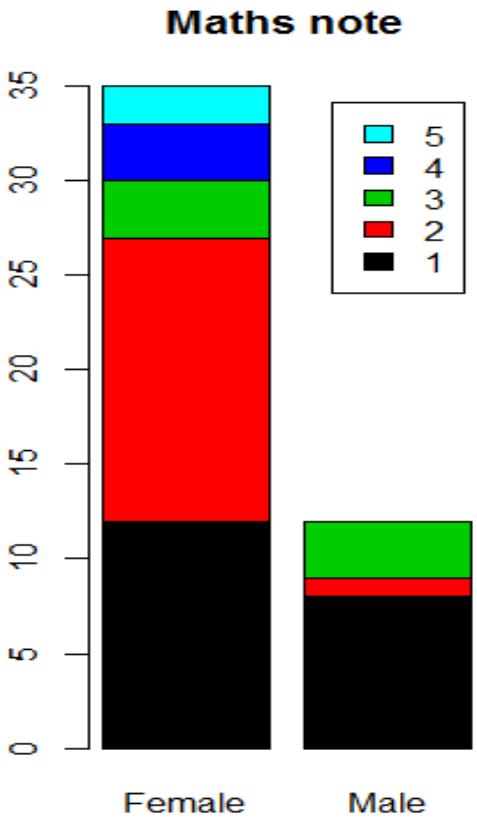
```
par(mfrow=c(2,2))

e1 <- ecdf(diak[,1])
plot(e1,xlab="note",
    ylab="relative frequency",
    main="Maths")

e2 <- ecdf(diak[,2])
plot(e2,xlab="note",
    ylab="relative frequency",
    main="Law")

e3 <- ecdf(diak[,5])
plot(e3,xlab="cm",
    ylab="relative frequency",
    main="Height")

e4 <- ecdf(diak[,6])
plot(e4,xlab="",
    ylab="relative frequency",
    main="Shoe size")
```



We come back to this figure later as well, in the section about goodness-of-fit.
If we have a qualitative variable, then we can draw a simple bar chart instead of the histogram.
The next sequence of commands produce this diagram for variable gender, first for all observations, then separately for each maths note. The legend is generated automatically, if we provide its text.

```
par(mfrow=c(1,2))
no <- sum(diak[,8]=="F")
ffi <- sum(diak[,8]=="M")

nomat <-
     c(sum(diak[,8]=="F" & diak[,1]==1),
       sum(diak[,8]=="F" & diak[,1]==2),
       sum(diak[,8]=="F" & diak[,1]==3),
       sum(diak[,8]=="F" & diak[,1]==4),
       sum(diak[,8]=="F" & diak[,1]==5))
ffmat <-
     c(sum(diak[,8]=="M"&diak[,1]==1),
       sum(diak[,8]=="M"&diak[,1]==2),
       sum(diak[,8]=="M"&diak[,1]==3),
       sum(diak[,8]=="M"&diak[,1]==4),
       sum(diak[,8]=="M"&diak[,1]==5))

barplot(c(no,ffi),main="Gender",
          names.arg=c("Female","Male"))

barplot(cbind(nomat,ffmat),
        names.arg=c("Female","Male"),
        col=c(1:5),main="Maths note",
       legend.text=as.character(c(1:5)))
```

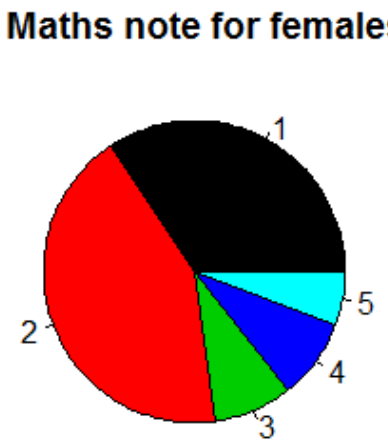


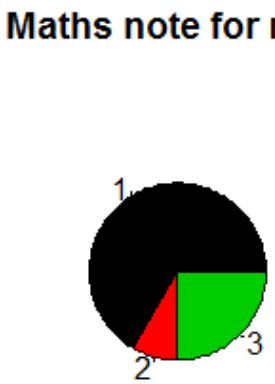


The results show that the best two notes (4 and 5) were to be found only among the female students and that the proprtion of failiure (note 1) is also much higher for males than for females.

It can also be shown by a pie chart:

```
par(mfrow=c(1,2))
pie(nomat,radius=1,col=c(1:5),
          main="Maths note for females")
pie(ffmat[1:3],radius=sqrt(ffi/no),
          main="Maths note for males",
          col=c(1:5))
```





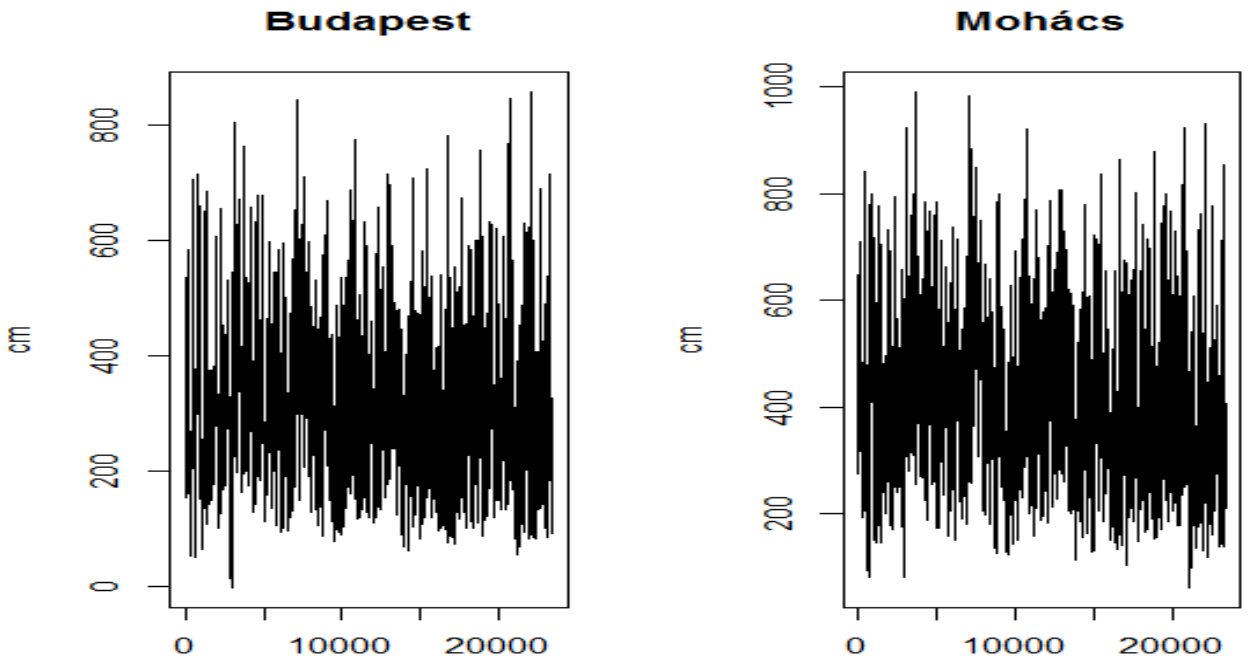
Let us observe that the ratio of the radii is the square root of the ratio of the sample sizes, so the plot is area-proportional. The 0 frequencies had to be removed in order to omit their labels. However, it

is worth noting that the researchers argue that the average reader cannot determine the relation between the arcs, so the use of the bar chart is much more recommended.
Now let us switch to another data set, which gives an insight to real research problems. The time series of daily water level measurements of the Danube at Budapest and Mohács will be investigated. In order to avoid problems caused by missing data, we deal with the period between January 1, 1946 and December 1, 2009 as earlier (especially during the world wars) there were months with completely missing observations.

```
bud <- read.table("D:\\oktatas
                  \\BudapestNapi.lst")
moh <- read.table("D:\\oktatas
                  \\MohacsNapi.lst")
bud <- bud[bud[,1]>1945,]
moh <- moh[moh[,1]>1945,]
bud <- bud[,4]
moh <- moh[,4]
```

If we utilize that the observations come from neighbouring days (i.e. we have a time series data) and plot the data as a function of time, we get the next figure, which is not too informative.

```
par(mfrow=c(1,2))
plot(bud,type="l",xlab="",
         ylab="cm",main="Budapest")
plot(moh,type="l",xlab="",
         ylab="cm",main="Mohács")
```

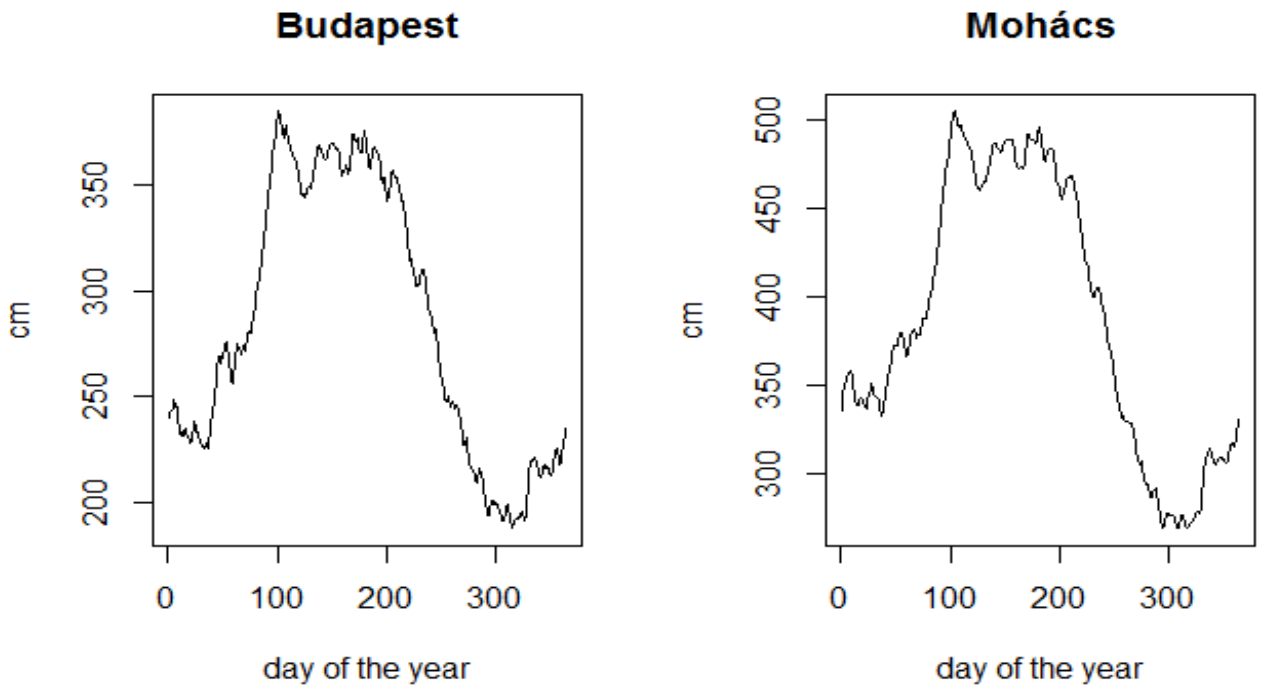


The problem is caused by the too many data points. Besides the extremely high or extraordinary law values not many details can be identified. It is much more interesting to investigate the annual cycle of the water levels. Simply the daily averages can be used for this purpose. As we have previously omitted the leap days from the data, these can easily be computed, as follows.

```
par(mfrow=c(1,2))
r <- c(0:63)
budev <- bud[1:365]; mohev <- budev

for (i in 1:365)
  {budev[i]=mean(bud[r*365+i]);
   mohev[i]=mean(moh[r*365+i])}
summary(budev)
summary(mohev)

plot(budev,type="l",
```

```
xlab="day of the year",
ylab="cm",main="Budapest")
```

```
plot(mohev,type="l",
          xlab="day of the year",
          ylab="cm",main="Mohács")
```



Here all the details can be easily verified. The long lasting sequence of higher values from mid-spring is true for both stations. Other patterns are also very similar. The data from Mohács are not only larger, (this is mainly due to the specific location of the water gauge) but exhibit larger variance.

2.3. Graphical tools for checking normality

While investigating the distribution of our data, quite often the question of normality arises. We shall come back to this important issue in Section 3, where we use the tools of the mathematical statistics. But now we check this assumption by simple diagrams.

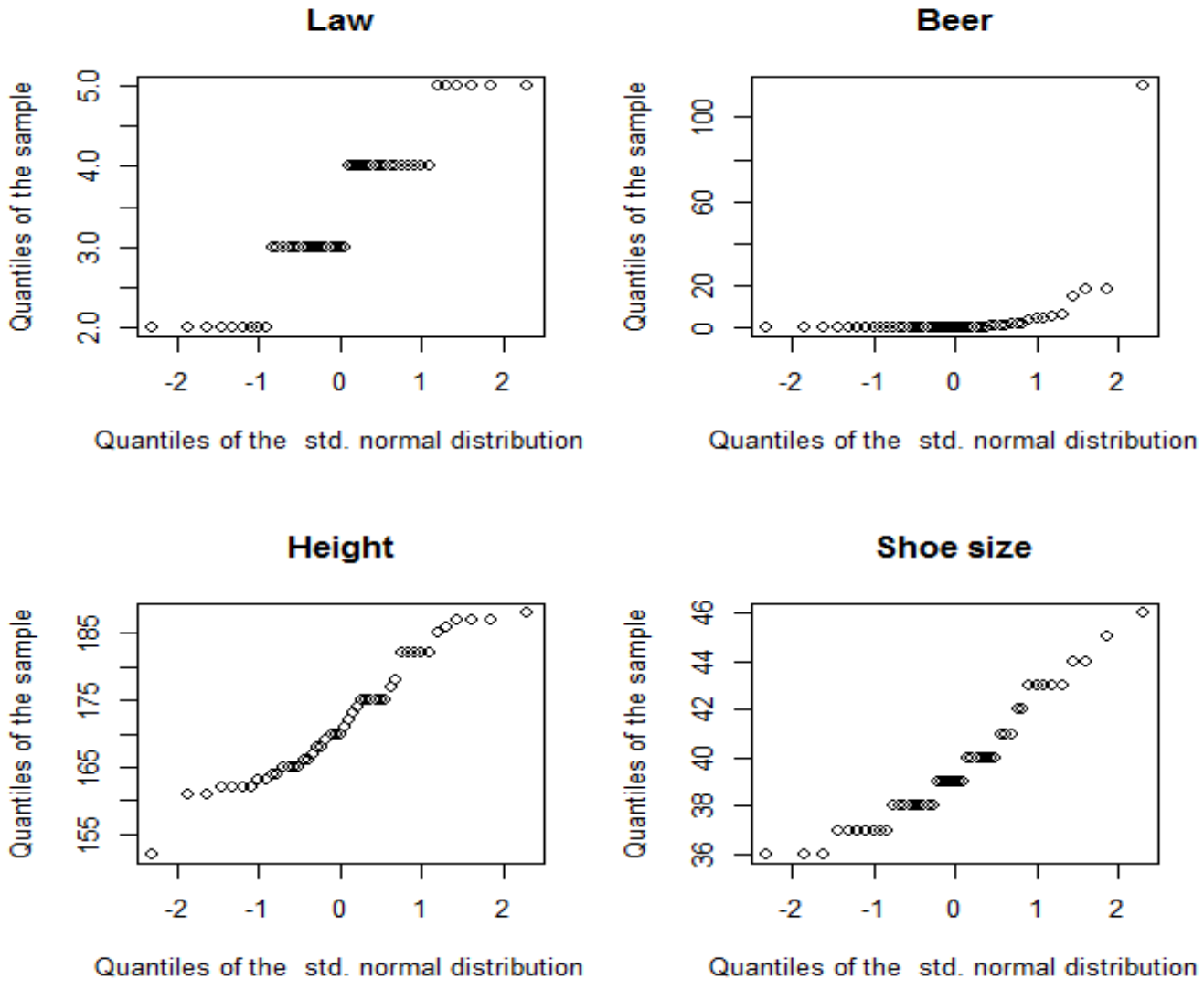
The histogram itself is a self-explaining figure of the distribution, but the choice of the classes may cause substantial effects on the results. There are more objective methods for the same purpose. First we present the so-called QQ-plot. Here we plot the following points:

$$\left(\Phi^{-1}\left(\frac{i}{n+1}\right), x_i^{(n)}\right)$$

i.e. we compare the quantiles of the assumed distribution with the empirical quantiles. As the normal distribution is a location-scale family, there is no need for parameter estimation here. If the sample follows a normal distribution, then the points of the diagram should be near to the line, which goes

through the point $(0, \bar{x})$ and having steepness equal to the standard deviation $sd(x)$. It is common to show this line on the diagram, together with the points. The disadvantage of this method is that in its simple form it does not give information on the significance: what deviations can still be considered as random. To make the evaluation easier one may present a QQ-plot, which belongs to a simulated normal sample of the same size as our data.

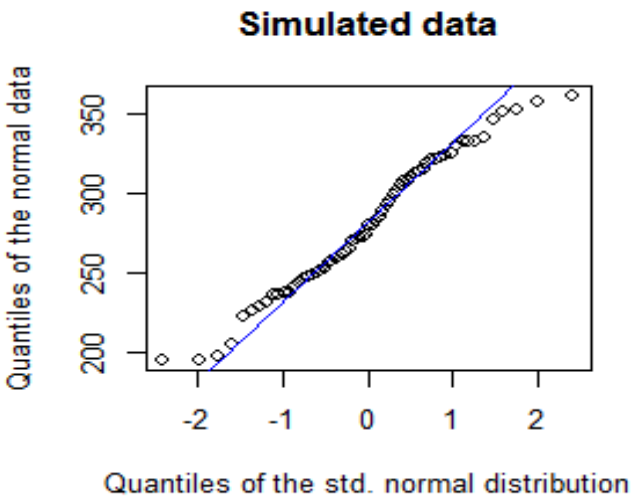
```
par(mfrow=c(2,2))

xlab <- paste("Quantiles of the",
              "std. normal distribution")

qqnorm(diak[,2],main="Law",
 xlab=xlab,
 ylab="Quantiles of the sample")

qqnorm(diak[,3],main="Beer",
 xlab=xlab,
 ylab="Quantiles of the sample")

qqnorm(diak[,5],main="Height",
 xlab=xlab,
 ylab="Quantiles of the sample")

qqnorm(diak[,6],main="Shoe size",
 xlab=xlab,
 ylab="Quantiles of the sample")
```

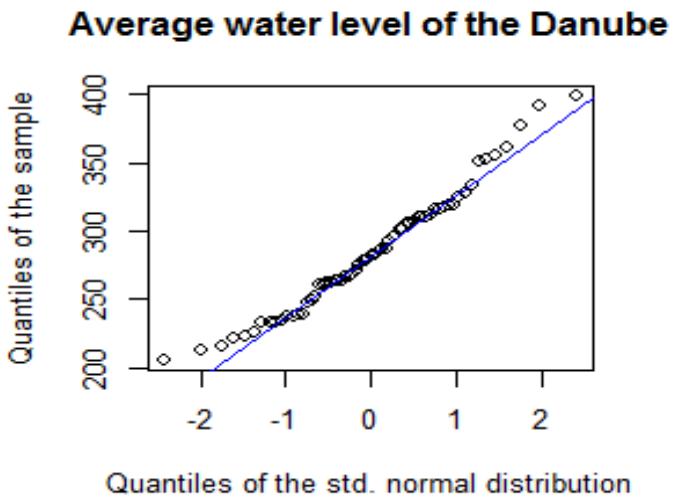


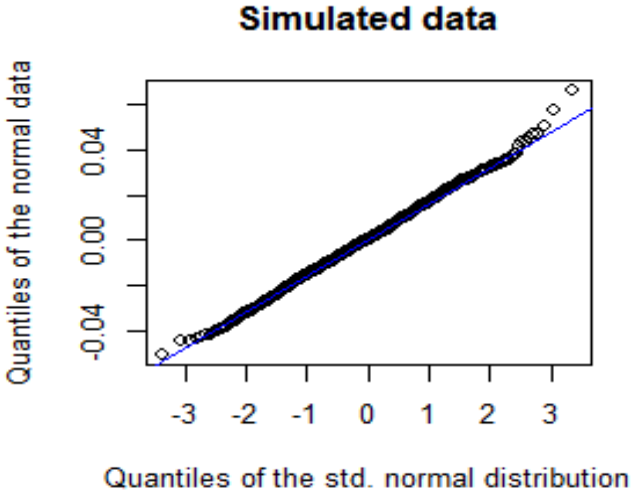
For the beer consumption the outlier destroys the fit completely (which is of course not good anyway, due to the skewness of the distribution). For the otherwise symmetric law-notes, the observations have only four distinct values, this makes the normality hopeless. In case of shoe size the relatively frequent small values look unusual in the case of a normal distribution. Body height is often modelled by normal distribution, but our data do not support this assumption. Of course, such small data sets quite often produce unusual behaviour. About the question, what difference is still typical, one may get information by simulations.

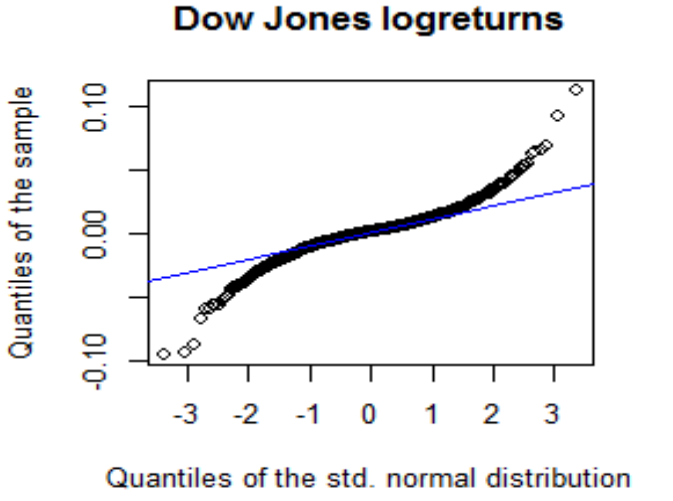
This is shown in the next figure, where the left panels show simulated data from normal distribution, while on the right panel there is the real data. The first row shows the annual average water levels, while the second gives the daily logreturns, based on closing values of the Dow Jones stock index data from the New York stock exchange, which can be freely downloaded from the webpage of the Yahoo. Here we used the daily log-returns $\log(X_{t+1}-\log(X_t)$, based on the daily closing values X_t. In order to avoid the problems with large data sets, here we just show the results for the data from 2005 till 2010.

```
par(mfrow=c(2,2))

xlab <- paste("Quantiles of the",
              "std. normal distribution")
ylab <- "Quantiles of the normal data"

r <- c(1:365)
budev2 <- bud[1:64]; mohev2=budev2
for (i in 1:64)
  {budev2[i] <- mean(bud[r+365*(i-1)]);
   mohev[i] <- mean(moh[r+365*(i-1)])}
y <- rnorm(64,mean(budev2),sd(budev2))

qqnorm(y,
  main="Simulated data",
  xlab=xlab,
  ylab=ylab)
qqline(y,col=4)

qqnorm(budev2,
  main=
   "Average water level of the Danube",
  xlab=xlab,
  ylab="Quantiles of the sample")
qqline(budev2,col=4)

dj <- read.table("D:\\oktatas\\aring
            \\nasdaq_djia_2005_2010.txt")
djl <- dj[,1]
y <- rnorm(length(djl),mean(djl),sd(djl))

qqnorm(y,main="Simulated data",
  xlab=xlab,
  ylab=ylab)
qqline(y,col=4)

qqnorm(djl,main="Dow Jones logreturns",
  xlab=xlab,
  ylab="Quantiles of the sample")
qqline(djl,col=4)
```







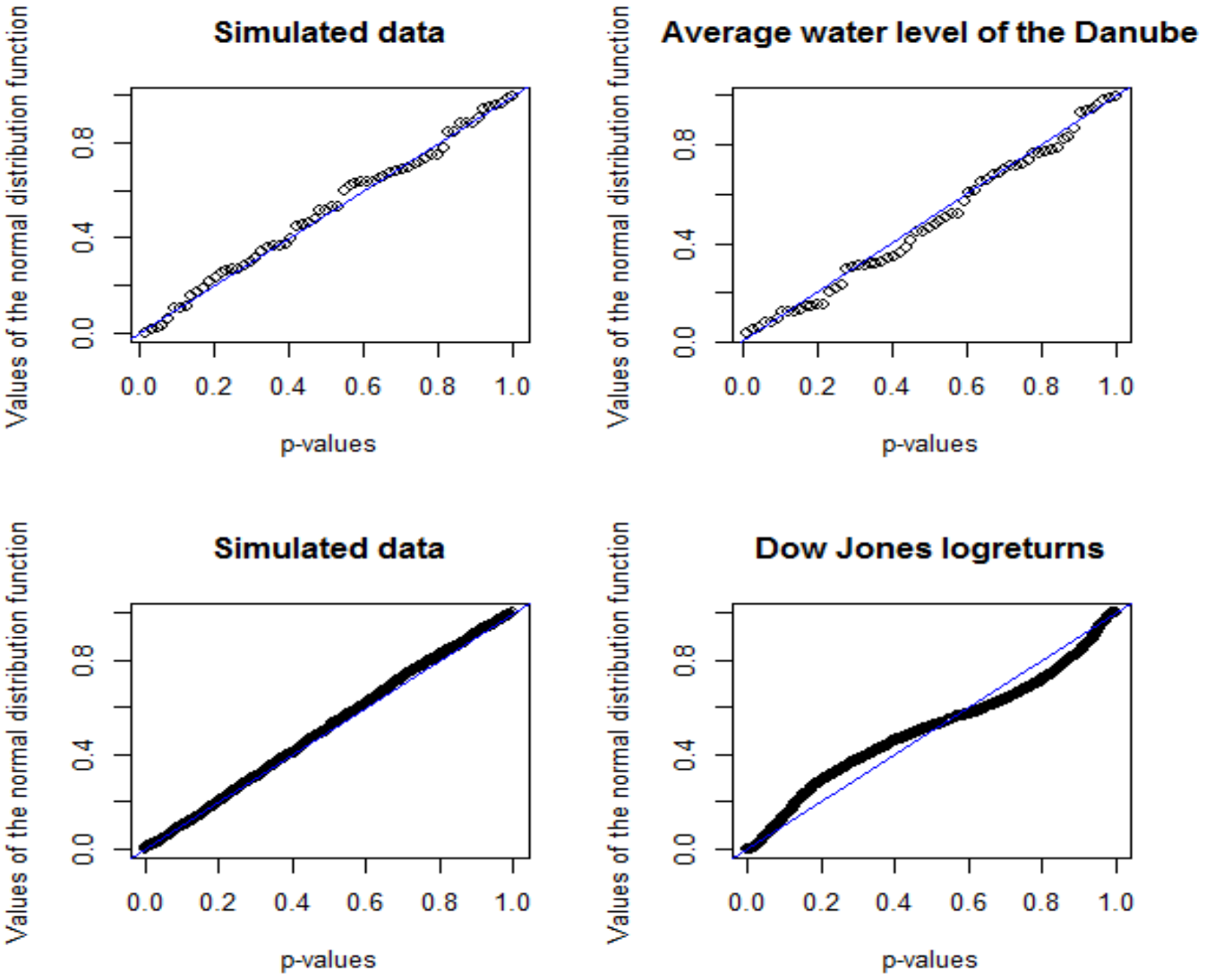


It can be seen, that while the average water level is near to the normal distribution (its theoretical background is the central limit theorem), the logreturns are far from being normal. This fact caused (and still causes) a lots of problems for those dealing with financial data.

Besides the QQ-plot diagram the PP-plot is also quite often used. Here the empirical distribution function of the sample is compared to the normal distribution function's value at the sample points. If the fit is good, these points lie near to a line.

```
par(mfrow=c(2,2))

y <- rnorm(64,mean(budev2),sd(budev2))
ylab <- paste("Values of the normal",
              "distribution function")
xlab<-"p-values"

plot(c(1:length(y))/(length(y+1)),
     pnorm(sort(y),
        mean(budev2),sd(budev2)),
     main="Simulated data",
     xlab=xlab, ylab=ylab)
abline(0,1,col=4)

plot(c(1:length(y))/(length(y+1)),
     pnorm(sort(budev2),
        mean(budev2),sd(budev2)),
     main=
     "Average water level of the Danube",
     xlab=xlab, ylab=ylab)
abline(0,1,col=4)

y <- rnorm(length(djl),mean(djl),sd(djl))
plot(c(1:length(y))/(length(y+1)),
     pnorm(sort(y),
         mean(djl),sd(djl)),
```

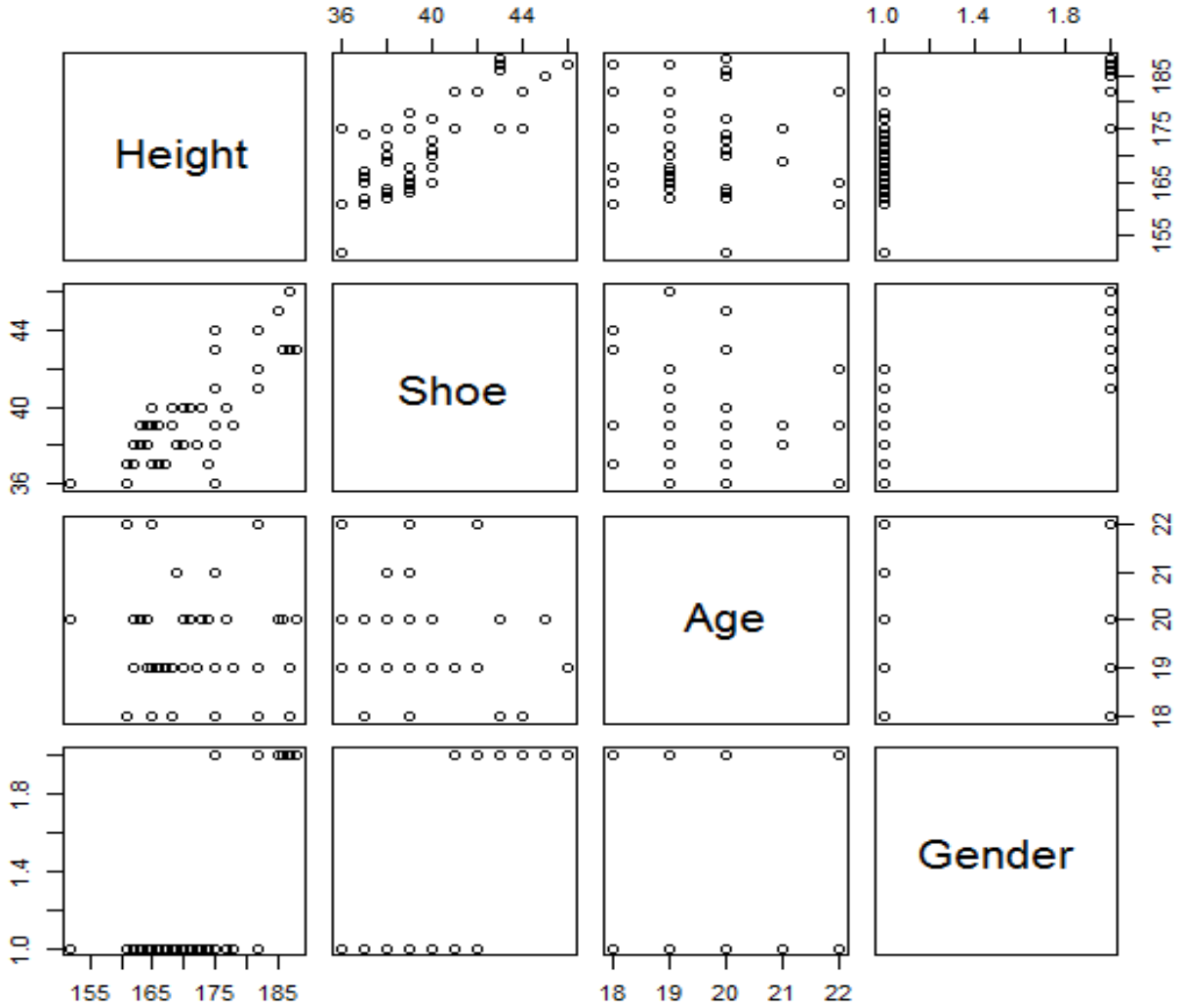
```
     main="Simulated data",
     xlab=xlab, ylab=ylab)
abline(0,1,col=4)

plot(c(1:length(y))/(length(y+1)),
     pnorm(sort(djl),
         mean(djl),sd(djl)),
     main="Dow Jones logreturns",
     xlab="xlab, ylab=ylab)
abline(0,1,col=4)
```



2.4. Checking bivariate dependence

The most natural first step is the pairwise plot of the variables. In order to keep the figures readable, we just consider the last four variables of our student data in the next simple command:

```
par(mfrow=c(1,1))
plot(diak[,5:8])
```



First we see from the plot that the categorical variables are coded by **R**, so they can also be plotted in the coordinates. There is a quite natural strong relationship between the height, the shoe size and the gender. Age looks as being independent from all other variables. We shall come back to these questions in the section, dealing with independence tests.

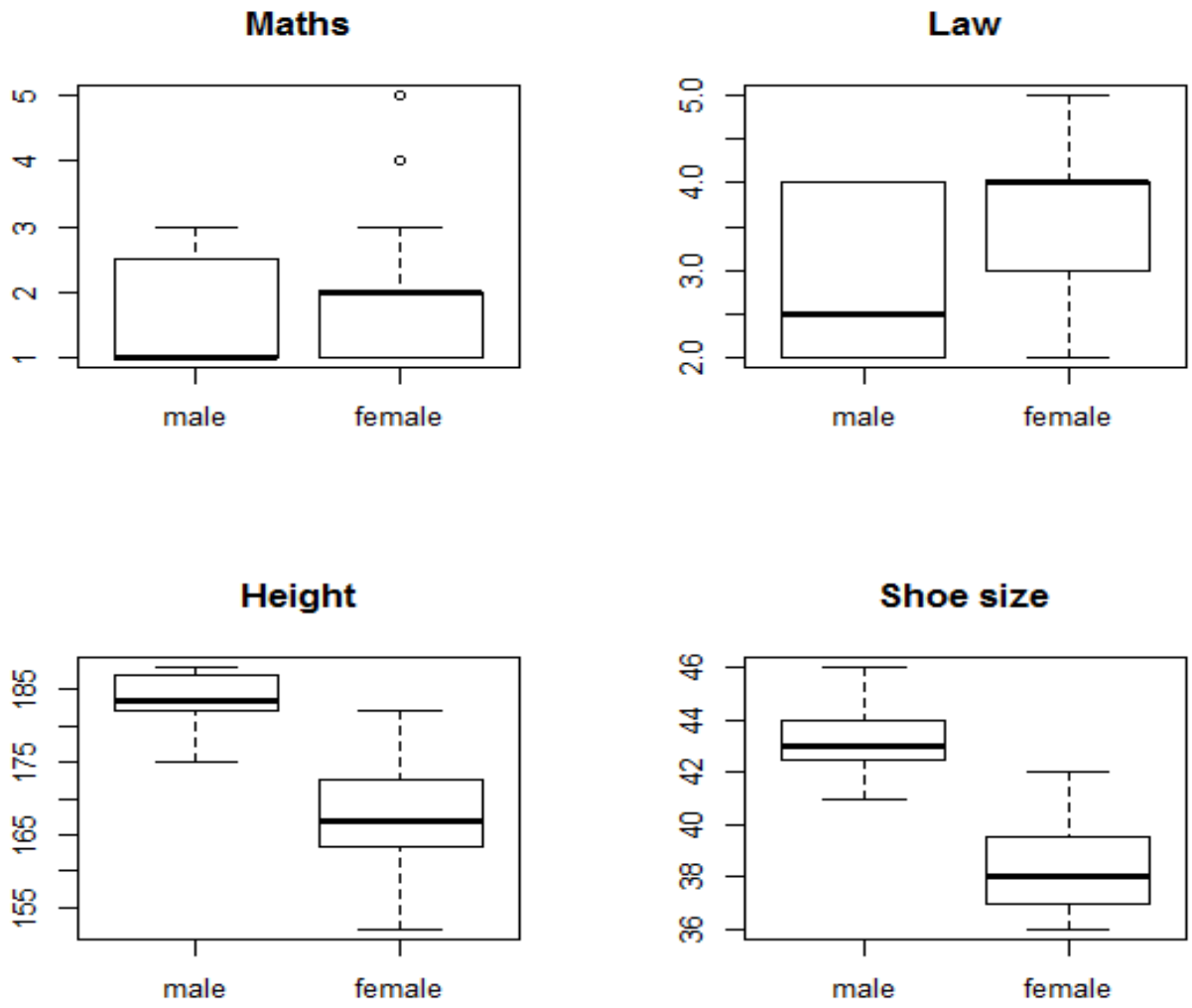
There is however, a tool worth considering in **R**. To apply this, let us investigate the shoe size data. On a simple plot we cannot check exactly how many times have the different values occurred. We see that each size from 36 to 42 has occurred at least once, but we cannot evaluate the frequencies for the females for example. We can present these on our plot if we use not the original values but their perturbed versions on the diagram. And this is not a problem if we consider the original meaning of the shoe sizes, since the observed integer values are rounded versions of a continuous variable (the size of the foot).

```
par(mfrow=c(1,2))
cipj<-jitter(diak[,6],factor=1.3)
plot(as.numeric(diak[,8]),cipj,
     xlab="Females      Males",
     ylab="shoe size",
     main="The shoe sizes ",axes=F)
axis(2);axis(4)
hist(diak[,6],
     xlab="",ylab="frequencies",
     main="Shoe size",
     breaks=c(1:12)+69/2)
rug(cipj)
```

Let us investigate some important points of the code! Jitter adds random perturbations to the observations. In order to keep the consistence of the figures, we have saved the perturbed values before adding in to the plot. In the first figure we had to convert the values of gender to numbers, otherwise we would have got a simple boxplot. This conversion would have resulted in strange decimals on the axes, which were first omitted, then redrawn to the two sides of the plot. The new feature of the second plot is the "rug' below the x axis, which now shows the perturbed values.

The dependence on a dichotomous variables or on categorical variables with at most a few values is best shown by a boxplot, as follows.

```
par(mfrow=c(2,2))
v<-list()
v$"male"<-diak[diak[,8]=="M",1]
v$"female"<-diak[diak[,8]=="F",1]
boxplot(v, main="Maths")

v<-list()
v$"male"<-diak[diak[,8]=="M",2]
v$"female"<-diak[diak[,8]=="F",2]
boxplot(v, main="Law")

v<-list()
v$"male"<-diak[diak[,8]=="M",5]
v$"female"<-diak[diak[,8]=="F",5]
boxplot(v, main="Height")

v<-list()
v$"male"<-diak[diak[,8]=="M",6]
v$"female"<-diak[diak[,8]=="F",6]
boxplot(v, main="Shoe size")
```



These diagrams show the dependence on gender for four variables. On this diagram the median is shown by the thick black line, the box shows the range between the quartiles, the whiskers range from the minimum till the maximum, if its length is not larger than 1.5-times the difference between the quantiles – in this default case. If the range of the data is larger, the outliers are represented by individual circles (as in case of females' maths notes: 4 and 5 seem to be outliers in this sense).

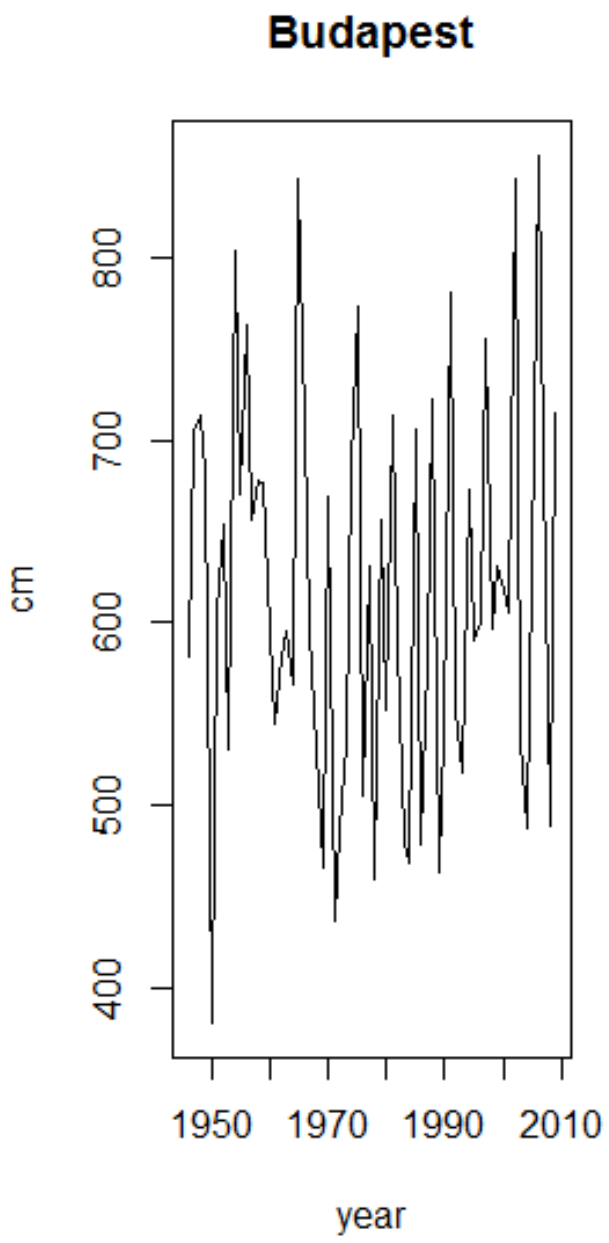
The results show that the women have better results, but they have smaller height and shoe sizes. This gives an answer for the seemingly unusual behaviour of the correlation matrix.

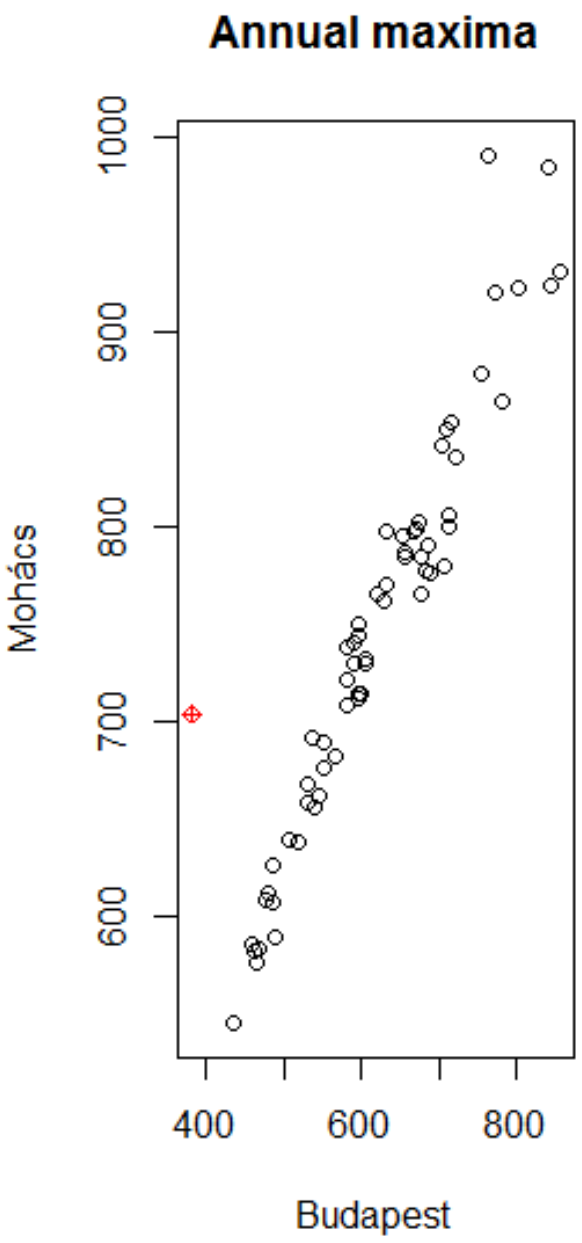
It is worth noting that for the default 8 digits we just use 3, which is enough in most of the cases and enormously improves the readability of the table.

```
> round(cor(diak[,c(c(1:3),c(5:7))]),3)

        Maths  Law   Beer  Height Shoe   Age
Maths   1.000  0.52 -0.16 -0.117 -0.207  0.055
Law     0.516  1.00 -0.22 -0.209 -0.362  0.232
Beer   -0.156 -0.22  1.00  0.285  0.371  0.007
Height -0.117 -0.21  0.28  1.000  0.790 -0.175
Shoe   -0.207 -0.36  0.37  0.790  1.000 -0.234
Age     0.055  0.23  0.01 -0.175 -0.234  1.000
```

For the analysis of the flood data another important direction is the investigation of the extremes. This can easiest be done by the investigation of the annual maxima. These values are are calculated and plotted by the following commands.

```
par(mfrow=c(1,2))
r <- c(1:64)
ev <- 1945
nap <- c(1:365)
budmax <- bud[1:64];
mohmax<-budmax

for (i in 1:64)
    {budmax[i] <- max(bud[(i-1)*365+nap])
mohmax[i] <- max(moh[(i-1)*365+nap])}

summary (budmax)
summary(mohmax)

plot(ev,budmax,type="l",xlab="year",
     ylab="cm",main="Budapest")
plot(budmax,mohmax,xlab="Budapest",
     ylab="Mohács",main="Annual maximuma")
points(budmax[5],mohmax[5],col=2,pch=10)
```





On the first panel we just show the data from Budapest. It is an important statistical question if the visible upward trend is significant or may just be a random effect. Another question is - based on the plot on the second pane, where the scatterplot of the annual maxima is shown - whether one of the data from year 1950 (marked red) may be erroneous, as the difference between the two sites is much larger than for any other years.

III. Tests of hypotheses

Hypothesis testing with the help of the R programming environment

3.1. Hypothesis testing

Hypothesis testing is one of the most typical methods in statistics beside parameter estimation and model building. In case of the hypothesis testing we decide – based on our sample – if the $\mathtt{H}_0$ hypothesis can be accepted for the distribution of our observations. Our assumption is that one of the disjoint hypotheses $\mathtt{H}_0$ and $\mathtt{H}_1$ is true for the distribution of our sample elements.

As hypothesis testing is based on random samples, it may be erroneous. It may happen that we reject that a sample from $\mathtt{H}_0$ is actually from $\mathtt{H}_0$ and it may accept a random sample coming from $\mathtt{H}_1$ as one belonging to $\mathtt{H}_0$. An erroneous decision of the hypothesis test is called an error of type I - or type II, depending on whether the true distribution of the sample belongs to $\mathtt{H}_0$ or to $\mathtt{H}_1$, as follows:

- An error is of *first type/type I* , if the sample is from $\mathtt{H}_0$ and the hypothesis test rejects the $\mathtt{H}_0$;

- An error is of *second type/type II*, if the sample is from $\mathtt{H}_1$ and the hypothesis test accepts the $\mathtt{H}_0$.

It can easily be seen that based on random samples one cannot make surely correct decisions even having arbitrarily many observations – except of very unusual cases.

But it is also true that under wide circumstances there is a procedure, which ensures that it has the *minimal* probability of *type II* error, *given* the probability of *type I* error. We shall call these methods *optimal*.

The classical definition of a hypothesis test is by the help of a critical region (meaning that $\mathtt{H}_0$ is rejected if the observations fall into this region), which is *level* α if it has probability at most α under the distributions belonging to $\mathtt{H}_0$, and it is *optimal* if it has maximal probability under every distribution in $\mathtt{H}_1$ among the level α sets (we give formal definitions in the next paragraphs). It is important to note, that α is usually small (at most 0.05), so the rejection of the $\mathtt{H}_0$ is informative in the sense, that it is erroneous only with a probability not larger than α. On the other hand if we accept $\mathtt{H}_0$ it is not as informative, as it tells us only the 'lack of evidence for rejection' – which is a much weaker statement than the proof of $\mathtt{H}_0$.

Let us have $X_1(\omega) = x_1, \ldots , X_n(\omega) = x_n$ observed values from the $(X_1, ..., X_n)$ random vector. We may construct our method based on an n-variate $d()$ function (test statistic) which has a univariate image: $d(x_1, ..., x_n) \in \mathbb{R}$. We intend to find a critical region $K \subset \mathbb{R}$, for which the probability

$$P(d(X_1, ..., X_n) \in K)$$

is small ($<= \alpha$) if the distribution of the observed X variables lies in $\mathtt{H}_0$, and it is large if this distribution is in $\mathtt{H}_1$.

Let $\mathcal{K}_\alpha$ be the set of such $K \subset \mathbb{R}$ sets, for which if X has a distribution in $\mathtt{H_0}$, then $P(X \in K \leq \alpha)$. A critical region $K \in \mathcal{K}_\alpha$ is called *optimal* if there is no $K' \in \mathcal{K}_\alpha$ for which $P(X \in K) \leq P(X \in K')$ would hold under a distribution in $\mathtt{H_1}$.

We shall deal with hypothesis pairs $(\mathtt{H_0}, \mathtt{H_1})$ and test statistic $d()$ for which the distribution of the statistic $d(X_1, ..., X_n)$ is known for distributions in $\mathtt{H_0}$. Let us denote the distribution function for the element in $\mathtt{H_0}$, which is nearest to $\mathtt{H_1}$ by G_0.

For the hypothesis pairs and statistics under investigation it will also be true that the optimal K (or at least the critical region we shall apply) will be either a half-line or the union of half lines (i.e. the complement of a finite section).

If we look for the critical region in the form of a half-line, then it can only be the $[G_0^{-1}(1-\alpha), \infty)$ or the $(-\infty, G_0^{-1}(1-\alpha)]$ as these are the half-lines which have probability α under the distribution G_0.

The situation is not as unanimous if we want to get the critical region as the union of two half-lines. In this case we usually choose a symmetric one, where both of the half-lines have probability $\alpha/2$, i.e. we get the set

$$K = (-\infty, G_0^{-1}(\alpha/2)] + [G_0^{-1}(1-\alpha/2), \infty).$$

This K is however, usually not optimal with respect to the definition above.

We can see that in the above-mentioned 'simple' cases to decide if the statistic is in the critical region, it is enough to give the quantiles of the distribution function $G_0(x)$. When we compare the actual value of the statistic to the appropriate one from $G_0^{-1}(\alpha)$, $G_0^{-1}(1-\alpha)$, $G_0^{-1}(\alpha/2)$ or $G_0^{-1}(1-\alpha/2)$, then we can decide if the statistic lies within the critical region.

If we use the computer for hypothesis testing – in the classical case, in accordance to the above-mentioned procedure – the computer expects two information from us: the data and the level α. As before, α denotes the maximal probability of the error of type I.

On the other hand, the program quite often does not tell us the decision about $\mathtt{H_0}$, just gives two numbers: the value of the statistic and the α- (or other, related) quantile of the distribution of our test statistic under $\mathtt{H_0}$. In the case, when the critical region is a (possibly infinite) interval or its complement, then this is just enough to the decision: first we create the critical region and then compare the value of the statistic to the quantile and either accept or reject $\mathtt{H_0}$.

One can see, that there is a step left in this classical procedure, which can possibly be automatized: the comparison of d and the g_α quantile. This step is simplified by the *p-value*.

Here we use the well-known fact that if we substitute a random quantity to its own distribution function, then we get a number with $\mathcal{U}([0,1])$ (uniform) distribution. This means that under $\mathtt{H_0}$, $G_0(d(X_1, ..., X_n)) \sim \mathcal{U}([0,1])$. This can be interpreted as

$$p - \text{value} = G_0(d),$$

i.e. the value of the test statistic, substituted into the a distribution function of the test statistic under the null hypothesis. This means that it is a number, which has $\mathcal{U}([0,1])$ distribution, if $\mathtt{H_0}$ is true.

The p-value is enough on its own to decide the hypothesis for any level α! Based on the fact that under $\mathtt{H_0}$ the p-value is uniformly distributed over the interval a $[0,1]$, it is enough to find the subinterval of length α within $[0,1]$, into which the p-value typically falls if $\mathtt{H_1}$ is true. This is obviously a set into which the statistic falls with probability α if $\mathtt{H_0}$ is true. So we have got a critical region with first type error probability α and with minimal probability of the type II error.

As the p-value is a monotonic transformation of the d statistic, if the critical region for d is based on half-lines, then the critical region for the p-value will coincide with the ends of the $[0,1]$: either $[0,\alpha]$, $[1-\alpha, 1]$ or $[0, \alpha/2] + [1-\alpha/2, 1]$.

This implies that typically the small and the large p-values form the critical region. Thus, based on the p-value, for the decision about the acceptance of $\mathtt{H_0}$ it has to be checked that the p-value is neither too small nor too large.

However, quite often the program uses the specific information on the hypotheses and transforms the statistic so that only the small p-values form the critical region. Anyway, it means that to the decision we do not need the quantiles and the test statistic.

The p-value of a statistic can be understood as the probability that we get a statistic value larger or equal the given one, based on another similar random sample from the $\mathtt{H}_0$.

There are misunderstandings about the meaning of the p-value. Some interpret the small p-value as 'the probability of the $\mathtt{H}_0$ is small under the given sample'. But this interpretation is completely erroneous. It is true that we base our decision on a probability, but this is not the probability of $\mathtt{H}_0$, but the probability of getting an even more extreme statistic compared to the one we actually got. The question about the probability of $\mathtt{H}_0$ is not even meaningful in our methodology, as we have not defined probabilities over the hypotheses!

Mathematically the p-value is just a similar statistic as the u, t or F which we shall introduce next. The only difference is that we have transformed the statistic on a way, that its distribution is uniform under $\mathtt{H}_0$. Such a statistic is more comfortable than a usual u, t or F etc. statistic, as the critical set to a given p-value can directly be got based on the desired level. We just have to consider, which are the typical values of the p-value under $\mathtt{H}_1$, and these should be covered by an interval of length α.

In the next part we present several such hypothesis tests, for which the usual test statistic follow a normal, t or F distribution under the hypothesis $\mathtt{H}_0$. In the classical approach these distributions looked so important, that quite often the methods have got their names from the given distribution. In case of the approach with the p-value, the problem is simplified to a statistic having uniform distribution. Even so, the tests have kept the names based on their traditional test statistic.

3.2. Hypothesis tests for the mean and the variance

In the next we show how to investigate hypotheses about the unknown mean and variance of the sample, with the help of the program **R**.

The u-test is a typical introductory example in the statistics literature. Not because of its frequent applications, but because its properties can be treated easily and on an exact manner. The u-test can be applied in cases, when we have either a large sample, or we can suppose that the individual observations have a normal distribution with a known variance σ^2. In case of the u-test under the above conditions, we must decide whether the expected value of the observations is equal to a given value m ($\mathtt{H}_0$ hypothesis) – taking into consideration the $\mathtt{H}_1$ hypothesis as well. Here we use the fact that if we have n observations and their average is $\bar{x}$, then supposing that the expected value of our observations is indeed m, then the statistic $u = (\bar{x} - m)/(\sigma/\sqrt{n})$ has standard normal distribution. However, the needed condition of the 'known variance' is very strong and is rarely true. If we replace σ^2 in the formula with one of its estimators, based on our sample like $s^2 = \sum_{j=1}^{n}(x_j - \bar{x})^2/(n-1)$, the corrected empirical variance, then the distribution of $d = (\bar{x} - m)/(s/\sqrt{n})$ is not standard normal, but the t_{n-1} distribution, which is substantially different from the normal in case of $n < 20$.

In the sequel we deal with hypothesis tests for the mean and the variance, as follows:

a) There is no function in the base system of **R**, which can perform the u-test. So if we want to apply it, we have to program it.

We introduce in some detail, how can the u-test be performed, when having a sample with known variance and if we may suppose that either the sample is large enough, or we can suppose for the different hypotheses that the individual observations have a normal distribution.

b) As a second example we show, how can we test by the classical F-test the equality of the unknown variances of the two samples. This problem can be solved by the command `stats::var.test()`.

c) The t-test is applicable for comparing the mean values under less strict conditions than it was needed in case of the u-test, as here we do not need to know the variances. For different versions of the t-test we shall apply the command `stats::t.test()`.

This function has to be parametrized in case of the typical problems as follows:
- in case of the one-sample t-test:
  `t.test(x,mu = m)`;
- in case of the two-sample t-test with equal variances:
  `t.test(x,y,var.equal = TRUE)`;
- in case of the two-sample t-test with different variances:
  `t.test(x,y,var.equal = FALSE)`;
- for the paired-sample t-test:
  `t.test(x,y,paired = TRUE)`.

The type of the alternative hypothesis H_1 can always be given with the same parameter `alternative`, which has the following possible values: `"two.sided"`, `"less"`, `"greater"`.

The $1-\alpha$ coverage probability of the confidence intervals can be set by the parameter `conf.level`, which has a default value of `0.95`.

3.2.a The u-test: equality of the mean in case of known variance

That test may be called u-test, where based on the data we can calculate a real number, which has standard normal distribution if the null hypothesis is true.

The u-test is usually applied in the case if we have normally distributed observations. As we usually consider a linear transformation (when calculating the standardized version) it is also needed that the standard deviation (or equivalently the variance) should be known – or at least that we can suppose it to be known.

We investigate two problems, which may be solved by the u-test:
- i) In the first case the question is if it is true that the expected value of the observations equals to a given m. This is the classical u-test.
- ii) In the second case the question is if the expectations of two groups of observations are equal. This is the so-called u-test for two samples.

i) The classical u-test

Let the independent $(x_1, ..., x_n)$ observations come from a normal distribution $\mathcal{N}(\mu, \sigma)$ and let their average be $\bar{x}$. This implies that the

$$d = \frac{\sum_{j=1}^{n} x_j/n - \mu}{\sigma/\sqrt{n}} = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

statistic follows the standard normal distribution.

Exercise. Let us suppose that the weight of 12 packs of wood were the following (in kilograms):

$$9.8, 9.7, 10.1, 10.4, 9.5, 10, 9.9, 10.1, 10, 10.1, 9.8, 9.9$$

Can it be accepted that the expected weight of the packs is 10 kg, if we suppose that the filling weight of the packs is independent and normally distributed with a standard deviation of $\sigma = 0.1$ kg?

Solution. Let us perform the test at the level of 5%. The average of the observed values is $\bar{x} = 9.94$ kg. The standard deviation of the average is $0.1/\sqrt{12} \approx 0.0289$ by the assumptions. Thus if the expected value of the weights of the packs is indeed 10 kg, then the statistic $d \approx (9.94 - 10)/0.0289 \approx -2.02$ is the observed value of a standard normal distribution. As the standard normal distribution falls with 95% probability into the ± 1.96 interval, and the value of d falls into the $(-\infty, -1.96) + (1.96, \infty)$ critical region, which has a probability of 5%, we have to reject at the level of 5% that the the expected weight of the packs is 10 kg.

We can do the same with the help of an **R** script as follows. Let the variable z contain the 12 observations. We must decide between the following pair of hypotheses, at the level of 5%:

H_0: the expected weight of the packs is 10 kg

H_1: the expected weight of the packs is $\neq 10$ kg

The program:

```
z<-c(9.8, 9.7,9.9,10.4,9.5,10,
9.9,10.1,10 ,10.1,9.8,10.1)
s<-.1;m<-10
d<-(mean(z)-m)/(s/sqrt(12))
q<-qnorm(.975)
p<-(1-pnorm(abs(d)))*2
```

The three values we have got:
$d = -2.020726$, $q = 1.959964$ and $p = 0.04330814$.

The variable p that was calculated last has a value of $\approx 4\%$. It is the probability that a number having standard normal distribution deviates further away from 0 than our $d \approx -2.02$. Thus it is the p-value of our test.

As we have already explained – if the H_0 is true, i.e. the expected weight of the packs is 10 kg – the p-value is uniformly distributed over the interval $[0, 1]$. Thus, if e.g. we want to decide on the level of 5%, then we must find an interval of length 0.05 within the interval $[0, 1]$ into which the p-value typically falls if the alternative hypothesis is true. As in our case the alternative hypothesis is that the expected weight is not 10 kg, and in this case `d` has typically a large absolute value, implying that the p-value is typically small if the alternative hypothesis is true.

Thus in our case the critical set, based upon the p-value is the interval $[0, 0.05]$, so we have to reject the H_0 if the p-value is small. The p-value is $\approx 4\%$, which falls into the critical region for the p-value. Thus we reject the H_0 i.e. the statement that the expected weight of the packs is 10 kg.

Now we were able to reject the null hypothesis, while we have run such a procedure, where the probability of the type I error was $\alpha = 0.05$ i.e. 1/20. Thus we knew that our method rejects the H_0 on average once in 20 such cases, when the sample comes from the H_0 (in our case this means that in the pack the average wood amount is 10 kg), only we have got by chance an average which differs substantially from the expected value.

If we applied a hypothesis test on the level of α = `1%` or = `3%` then the critical region for the p-value would have been `[0,.01]` or `[0,0.03]`. These do not contain the p-value of our sample – compared to these limits the p-value is not small –, so we would have accepted the null hypothesis at the levels of `1%` and `3%`.

The corresponding quantiles to these levels are `qnorm`(.995) $\approx$ 2.57 and `qnorm`(.985)) $\approx$ 2.17 Thus of course the critical regions $(-\infty, -2.57] + [2.57, \infty)$, $(-\infty, 2.17] + [2.17, \infty)$ which correspond to `d` do not contain our value `d` ≈ -2.02.

The last calculations show that if we choose a procedure that rejects erroneously the H_0 only in `3` or `1` percent of the cases, then we must accept the hypothesis that the expected weight is`10` kg. Then why do we not use these 'better', `3` or `1` percent levels?

The reason is that if we improve (decrease) the level, this automatically implies the shrinkage of the critical set. And this has the consequence that the procedure will be less critical to observations coming from the alternative hypothesis. Mathematically: the probability of type II error increases. Of course the level should be chosen according to the severity of the first type error. If the rejection implies the halting of a whole factory for example, e.g. in industrial process control, then it is common to apply low levels (and to ensure the quality – which corresponds to H_0 – by complementary procedures).

In our case if such packs would have been shipped, which have expected weight of `9.9` kg and we still would decide by `12` measurements and the above `d` whether we accept the H_0 hypothesis, that the expected weight is `10` kg then
if the level α of the hypothesis test is
5%, then the prob. of type II error is 6.6%
3%, then the prob. of type II error is 9.8%
1%, then the prob. of type II error is 18.7%
for the given element of `9.9`kg of the alternative hypothesis.

I.e. if we apply for example a test of level `1%` for testing if the expectation is `10`, then in almost one fifth of the cases a `12` element sample, which has expectation `9.9` will be accepted as coming from a population having expectation `10` in spite of having supposed a low variance (s.d.=0.1) for the individual observations.

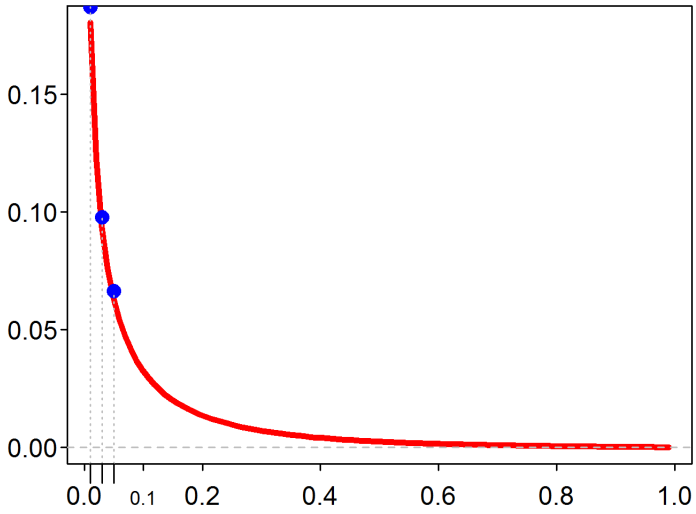
The above results can be calculated by the script

```
m1<-(9.9-10)/(0.1/sqrt(12))
q5<-qnorm(1-.05/2)
q3<-qnorm(1-.03/2)
q1<-qnorm(1-.01/2)
pnorm(q5,m1)-pnorm(-q5,m1)
pnorm(q3,m1)-pnorm(-q3,m1)
```

`pnorm(q1,m1)-pnorm(-q1,m1)`

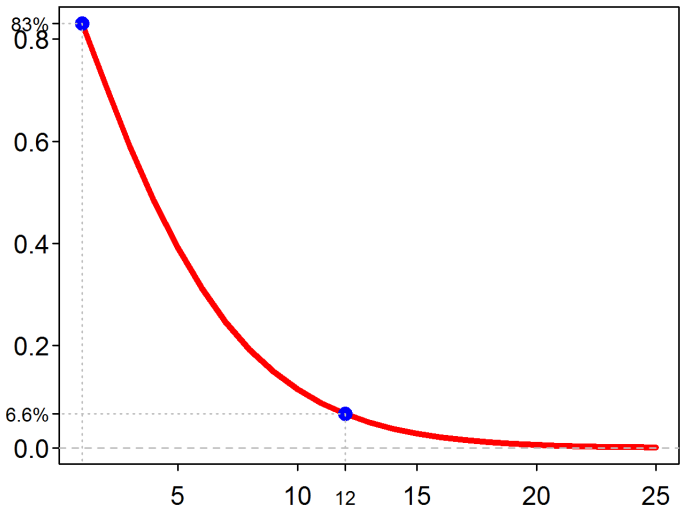
Here `m1`$\approx$`-3.46` is the standardized difference (`-0.1`) of a pack with expected value `9.9`.

The next figure shows the changes of the type II error probabilities (vertical axis) depending on the chosen type I error probability (horizontal axis) if we have a sample of `12` elements, coming from the normal distribution from H_1 with expected value `9.9`, and if the standard deviation of the observations is `0.1`.



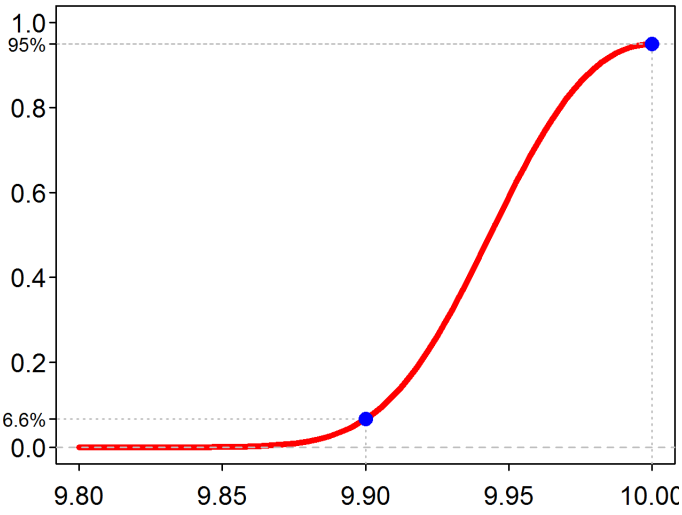
We have plotted the type II error probabilities only for type I error probabilities larger than 1/100, as it is true that the curve equals 1 for $\alpha = 0$, but it decreases very quickly from here. The blue dots on the curve are the type II error probabilities we have calculated previously.

The second curve shows the type II error probabilities (vertical axis) for given $\alpha =$ `0.05` type I error probability and expectation $\mu =$ `9.9` as a function of the sample size (horizontal axis) which varies in the range `1,...,25`.



If we have just a single observation, then the `5%`-critical region is the complement of the section $\pm$`1.96` $*$ `0.1` $\approx [-0.2, 0.2]$, into which a random observation from $\mathcal{N}(0.1, 0.1)$ falls with probability `pnorm(.2,.1,.1)-pnorm(-.2,.1,.1)`$\approx$ `0.8299`. That is the reason why is the value of the curve at `1` observation approximately `0.83`. The other dot on the curve is the previously computed `6.6%` type II error probability for a sample having `12` elements.

The third curve shows the type II error probabilities (vertical axis) for a given sample size of `12` as a function of the expected value of the sample (horizontal axis). The type II error probability is obviously the highest if we have $\mu =$ `10`: here it equals to the complement of the level of the procedure, `0.95`.



It can be seen, how quickly decreases the type II error probability as the difference between the actual expectation and `10` increases. Here we see again, that for $\mu = 9.9$ the procedure with $\alpha = 5\%$ has a type II error probability of `6.6%`.

ii) The u-test for the averages of two samples

Let the first set of observations be $x_1, ..., x_n$, and the other $y_1, ..., y_m$. Let us suppose that these are observed values of the independent $X_1, ..., X_n$ and $Y_1, ..., Y_m$ random quantities, where $EX_1 = ... = EX_n = \mu_x$ and $EY_1 = ... = EY_m = \mu_y$ further $DX_1 = ... = DX_n = \sigma_x$ and $DY_1 = ... = DY_m = \sigma_y$. Under these conditions, if the expected value of the two groups is equal, i.e. if $\mu_x = \mu_y$, then the distribution of the statistic

$$d = \frac{\bar{x} - \bar{y}}{\sqrt{\sigma_x^2/n + \sigma_y^2/m}}$$

is standard normal.

Thus, if we want to investigate whether the equality of the two expectations can be accepted at level α, then we must check whether the d defined previously belongs to the critical set

$$K = (-\infty, \Phi^{-1}(\alpha/2)] + [\Phi^{-1}(1-\alpha/2), \infty).$$

Furthermore, if our alternative is that the expected value of the first group is larger than that of the second, then this question can be answered affirmatively if d is larger than $\Phi^{-1}(1-\alpha)$. Finally, if our alternative is that the expected value of the second group is larger, then $(-\infty, \Phi^{-1}(\alpha)]$ is the critical region.

If we want to decide based on the p-value, then the p-value of the two-sided alternative $\mathtt{EX} \neq \mathtt{EY}$ is $2*(1-\Phi(|\mathtt{d}|))$, the p-value to the alternative $\mathtt{EX} > \mathtt{EY}$ is $1-\Phi(\mathtt{d})$, and that of to $\mathtt{EX} < \mathtt{EY}$ is $\Phi(\mathtt{d})$.

Let us consider the previous example again; suppose that the first sample is the first 5 out of the 12 observations, and the second is the last 7. Furthermore let us suppose again, that the standard deviation is 0.1 in both groups. Then the procedure above can be carried out for the following hypothesis-pair

$\mathtt{H_0}$: $\mathtt{EX} = \mathtt{EY}$
$\mathtt{H_1}$: $\mathtt{EX} < \mathtt{EY}$

and at the level $\alpha = .01$ as follows:

```
z<-c(9.8, 9.7,9.9,10.4,9.5,10,
9.9,10.1,10 ,10.1,9.8,10.1)
(xa<-mean(z[1:5]));(ya<-mean(z[6:12]))
(d<-(xa-ya)/(.1*sqrt(1/5+1/7)))
qnorm(.01)
```

The results:
The two averages: 9.86 and 10.
The value of the d statistic: -2.39.
The upper limit of the 1%-critical region: -2.33.

It means that we have to reject on the level $\alpha = 1\%$ the hypothesis about the equality of the two expectations to the alternative that the expectation of the second group is larger as the statistic $d =$-2.39 belongs to the critical set $(-\infty, -2.33]$.

If we had considered the two-sided alternative, then as qnorm(0.005)=-2.575829, the critical region would have been $(-\infty, -2.58] + [2.58, \infty)$ which does not contain $d =$-2.39, so we had to accept the hypothesis of the equality!

But on the other hand as the critical values belonging to the 5% level and the one-sided alternatives are: qnorm(0.05)=-1.64 and qnorm(0.95)=1.64, finally the two-sided are qnorm(0.025)=-1.96 and qnorm(0.975)=1.96, so we have to reject the hypothesis of the equality in case of any common alternative.

These decisions can be also read from the three p-values, being 0.0168, 0.0084 and 0.9916 respectively, which are obviously following from each other.

Summary

The value of the test statistic $d =$-2.39 should be an observed value of a standard normal variate in case of $\mathtt{H_0}$. We see that its absolute value is too large to consider it as a typical value under $\mathtt{H_0}$ (unless we want to be extremely sure about our rejection – this was shown for the results in case of $\alpha = 0.01$). Its value much more refers to the alternative hypothesis. These results can be expressed by the previously calculated quantile values and obviously by the computed p-values as well.

3.2.b Testing the equality of variances by F-test

Let us supoose that we have the observations $x_1, ..., x_n$ and $y_1, ..., y_m$. We know that they are normally distributed and independent. Let us suppose that within the two samples the variances of the observed random quantities are constant. However, it is questionable whether the variances of the two samples are equal. So let our research question be the equality of the variances and the alternative be that the first sample has a larger variance.

It means that we consider the following two hypotheses:

$$\mathtt{H_0} : \sigma_x = \sigma_y$$
$$\mathtt{H_1} : \sigma_x > \sigma_y$$

If $\mathtt{H_0}$ is true, then the ratio of the corrected empirical

variances:

$$d = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2/(n-1)}{\sum_{j=1}^{m}(y_j - \bar{y})^2/(m-1)}$$

has the $F_{n-1,m-1}$ distribution.

Based on this observation we can construct a hypothesis testing method for a given level α as follows. We have to reject the hypothesis of the equality if for the calculated d statistic $d > F^{-1}_{n-1,m-1}(1-\alpha)$, where $F^{-1}_{n-1,m-1}(1-\alpha)$ is the α-quantile of the F-distribution with parameters $n-1, m-1$.

Using the previous data, we can continue the computation in **R**, considering the first 5 elements of our data as measurements from x, and the last 7 elements as measurements from y.

```
z<-c(9.8, 9.7,9.9,10.4,9.5,10,
9.9,10.1,10 ,10.1,9.8,10.1)
(d<-var(z[1:5])/var(z[6:12]))
qf(.95,4,6)
```

The value of d is 8.475, and the 0.95-quantile is 4.53, which means that we have to reject the equality hypothesis of the two samples. This is shown by the p-value as well, which is by the formula `1-pf(d,4,6)` just about `0.012`.

We may do the same investigation using a suitable parametrization of the `var.test()` function in the `stats` package

```
var.test(z[1:5],z[6:12],1,"greater")
```

The result of the call is the following:

```
F test to compare two variances

data:  z[1:5] and z[6:12]
F = 8.475, num df = 4, denom df = 6, p-value = 0.01207
alternative hypothesis:
true ratio of variances is greater than 1
95 percent confidence interval:
1.869344      Inf
sample estimates:
ratio of variances
8.475
```

We can see that we have got a somewhat more detailed result, compared to our own calculations.

- Among the parameters of `var.test()` there is a number `1`, meaning that we intend to test the equality of the variances. If the question would have been that the variance of the first group equals 1.5-times the variance of the second, then we should have written 1.5 here.

- Seemingly among the results of the `var.test()` function the same statistic is appearing twice: the 8.475 is once the value of the `F` statistics and later as the `ratio of variances` – this is the `d` value, which we have calculated previously. The confidence interval refers also to this ratio, the lower limit of the interval is given by the formula `d/qf(.95,4,6)`.

- The `F` value equals with the `d` value in our case, as we wanted to test the equality of the variance ratio to, `1`. If the test would have been e.g. about the equality of the variance ratio to `r=1.5` (i.e. the variance of the first group is `r`-times the variance of the second group), then the `F` value would be `1/r` times the ratio of the variances – and of course the p-value would be the corresponding probability – while the ratio of the variances and its confidence interval would remain the same.

A simulation example

Let us suppose that we intend to show that

'The variance is reduced to one third
if we repeat each measurement three times.'

If we can indeed perform three measurements for a value to be measured and consider their average as a single result, then the variance of these measurements will exactly be the third of the variances of the original measurements. As if X_1, X_2, X_3 are independent variables with identical variance σ^2, then

$$D^2(\overline{X}) = D^2\left((X_1 + X_2 + X_3)/3\right) =$$

$$= \left(D^2(X_1) + D^2(X_2) + D^2(X_3)\right)/9 = \sigma^2/3$$

Let us now prove the statement via simulations.

Let us suppose that the errors of the original measurements are standard normal and let the function `gen()` generate their error, while the procedure, based on the repeated measurements have errors given by the function `gen3()`. In this example we have samples of size `n=19` and `n3=16`.
The simulation and the test can be run as follows:

```
rm(list=ls())
set.seed(123)
gen<-function() rnorm(1)
```

```
gen3<-function() mean(rnorm(3))
n<-19;n3<-16;
x<-NULL;y<-NULL;
for(i in 1:n) x<-c(x,gen())
for(i in 1:n3) y<-c(y,gen3())

var.test(x,y,3)
```

The result of the program is the following:

```
F test to compare two variances
data:  x and y
F = 1.3803, num df = 18, denom df = 15, p-value = 0.5336
alternative hypothesis:
true ratio of variances is not equal to 3
95 percent confidence interval:
1.483231 11.043005
sample estimates:
ratio of variances
4.141046
```

We can easily check if the script has indeed used the previously written formulas by the following program:

```
var.r<-var(x)/var(y);
(f.val<-var.r/3); # F.value
2*min(p<-pf(f.val,n-1,n3-1),1-p)# p.value
a<-.95;b<-1-(1-a)/2 # conf.int:
var.r/c(qf(b,n-1,n3-1),qf(1-b,n-1,n3-1))
  var.r # ratio of variances
```

In our case we chose the alternative hypothesis $\sigma_x/3 \neq \sigma_y$, as this was logical in this example. It is however worth checking, how would the p-value of the F-statistics and the confidence interval of the variance ratio change, if we consider one-sided alternatives.

We may calculate the p-value for the three different alternatives (the variance of the second group is not equal, larger or smaller than that of the first) using our own formula or the program `var.test()`:

```
# equal: 0.53
2*min(p<-pf(f.val,n-1,n3-1),1-p)
var.test(x,y,3,"two.sided")$p.value

# greater: 0.27
1-pf(f.val,n-1,n3-1)
var.test(x,y,3,"greater")$p.value

# less: 0.73
pf(f.val,n-1,n3-1)
var.test(x,y,3,"less")$p.value
```

Calculating the confidence interval for the three different alternatives (the variance of the second group is not equal, larger or smaller than that of the first) using our own formula or the program `var.test()`:

```
# equal: [1.48, 11.04]
c(var.r/qf(b,n-1,n3-1),
var.r/qf(1-b,n-1,n3-1))
var.test(x,y,3,"two.sided")$conf.int

# greater: [1.76, Inf]
c(var.r/qf(a,n-1,n3-1),Inf)
var.test(x,y,3,"greater")$conf.int

# less: [0, 9.39]
c(0,var.r/qf(1-a,n-1,n3-1))
var.test(x,y,3,"less")$conf.int
```

The interpretation of the simulation results:

In case of the hypotheses:
$\mathtt{H_0}$: $\sigma_x/3 = \sigma_y$
$\mathtt{H_1}$: $\sigma_x/3 \neq \sigma_y$
As the ratio of the two empirical variances turned out to be `4.14`, the value of the F statistic is `4.14/3=1.38`. As the two samples have `19` and `16` elements, this ratio has $\mathtt{F_{18,15}}$ distribution, if $\mathtt{H_0}$ is true. Thus the two-sided p-value is `0.5335` and the `95%` two-sided confidence interval for the `4.14` is `[1.48, 11.04]`.

We have got that the p-value of the F statistic is large, so the probability of getting larger or smaller variance ratios than the variance ratio of our sample is large if $\mathtt{H_0}$ is true, so the null hypothesis is not to be rejected.

We come to the same decision under the two one-sided alternatives as well (it means that the null hypothesis is not to be rejected, i.e. it can be accepted that the variance of the first sample is three times that of the second sample). As it can be seen from the results of the above scripts, the p-value under the right-sided alternative is `0.27` and the left-sided p-value is obviously its complement: `0.73`. So both p-values are so large that they do not support the rejection of the null hypothesis.

3.2.c The *t-test:* equality for the expectation in case of the unknown variance

We describe the testing methodology for four different problems. In all cases we shall use the `t.test()` function in the package `stats`. The test has practical importance if we have relatively few observations, as in case of many observations we usually make just a negligible error, if we approximate the distribution of the statistics by the standard normal distribution.

One sample t-test:

```
t.test(x,mu = m)
```

Let the observed vector be $x_1, ..., x_n$, a realization of the $X_1, ..., X_n$ independent, normally distributed random variables with the same expectation and variance. The question is whether the expected value of the observations is equal to a given value m.

The problem can be solved by the help of the following statistic:

$$d = \frac{\bar{x} - m}{\sqrt{\frac{\sum_{j=1}^{n}(x_j - \bar{x})^2}{n-1}/n}}$$

as if $X_j \sim \mathcal{N}(m, \sigma)$ with the given m and an unknown σ, then the distribution of the above statistic is t_{n-1}. One can observe that in the denominator of the statistic under the square root we have divided the corrected empirical variance by n.

The problem can be solved by the `t.test()` function in the `stats` package of the **R** environment as follows:

```
z<-c(9.8, 9.7,9.9,10.4,9.5,10,
9.9,10.1,10 ,10.1,9.8,10.1)
t.test(z,mu=10)
```

The parameter `mu=10` is the expected value in the null hypothesis. The default alternative is that the expectation $\neq 10$, so if we have other alternatives, then it has to be taken care of. The result:

```
One Sample t-test
data:  z
t = -0.8731, df = 11, p-value = 0.4012
 alternative hypothesis:
 true mean is not equal to 10
 95 percent confidence interval:
 9.794622 10.088711
 sample estimates:
 mean of x
 9.941667
```

We can see that the t-value is small in absolute value, (`t = -0.8731`) so the corresponding p-value is large `p-value = 0.4012`. The 95% confidence interval for the expected value of the observations contains the hypothetical m =10 expectation. Thus there is no need to reject the null hypothesis, i.e. that the expected value of the observations is 10.

Let us change the alternative:

```
t.test(z,mu=10,alternative="two.sided")
t.test(z,mu=10,alt="less")
t.test(z,mu=10,alt="greater")
```

The first is the same as before, only we have explicitly written the default parameters. In the next two calls we have abbreviated the name of the parameter, it is allowed as it can be identified this way as well.

Only the p-value and the confidence interval changes together with the alternative. The table of the results:

alternative	p-value	conf.int
two-sided	0.4012	(9.79, 10.09)
less	0.2006	(-Inf, 10.06)
greater	0.7994	(9.82, Inf)

Thus there is no need to reject the null hypothesis, i.e. that the expected value of the observations is 10 under any alternative hypotheses.

Two-sample t-test in case of equal variances:

```
t.test(x,y,var.equal = TRUE)
```

Let the first sample be $x_1, ..., x_n$ and the second $y_1, ..., y_m$. Let us suppose that the observations are independent, normally distributed, with unknown standard deviation σ.

Under these conditions, if the two samples have equal expected values, then the statistic

$$d = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{\sum_{j=1}^{n}(x_j - \bar{x})^2 + \sum_{\ell=1}^{m}(y_\ell - \bar{y})^2}{n+m-2}\left(\frac{1}{n} + \frac{1}{m}\right)}}$$

has t_{n+m-2} distribution. The first term in the denominator of the statistic under the square root is an

unbiased estimator for the common variance of the samples. As this is the

$$\sum_{j=1}^{n}(x_j-\bar{x})^2+\sum_{\ell=1}^{m}(y_\ell-\bar{y})^2$$

sum of the squared differences of the observations from the sample mean, divided by $n+m-2$. This variance estimator is multiplied by $1/n+1/m$ under the square root, as it can easily be seen (e.g. term by term) that this way we get an unbiased variance estimator of the numerator.

Continuing the earlier example, we can interpret and solve it with the following command:

```
t.test(z[1:5],z[6:12],var.equal=TRUE)
```

We get the answer:

```
Two Sample t-test
data: z[1:5] and z[6:12]
t = -1.0366, df = 10, p-value = 0.3243
alternative hypothesis:
difference in means is not equal to 0
95 percent confidence interval:
-0.4409225  0.1609225
sample estimates:
mean of x mean of y
9.86     10.00
```

The fact that the 95% confidence interval for the differences of the means contains the 0, implies that there is no need to reject the hypothesis of the equality. This is supported by the p-value as well. And a little practice helps to see it immediately from the value of the statistic. A t distribution with 10 degrees of freedom does not differ very much from the standard normal, so a number with absolute value around 1 cannot fall into the critical region.

We may save the result of the above `t.test()` to a variable, e.g. to `W`. If we investigate this variable with usual methods like `class(W)`, or `str(W)` etc., then we see that it is a list with 9 elements, of a class `htest` i.e. 'hypothesis test'. It turns out, however that the elements do not contain much more information than the ones already printed out.

Two-sample t-test in case of different variances:

```
t.test(x,y,var.equal = FALSE)
```

Let the first sample be $x_1,...,x_n$ and the second $y_1,...,y_m$. Let us suppose that the observations are independent, normally distributed, with unknown and possibly unequal standard deviation σ_x and σ_y. In the formula for d used earlier, the equality of the variances of the two samples was needed (as we supposed that the squared differences estimate the same quantity). If the standard deviations are unequal, then we cannot compute this way. However, there is another natural estimator for estimating the standard deviation of the differences of the mean. The denominator contains the corrected empirical variance of the sample variances s_x^2 and s_y^2, in more detail:

$$s_x^2=\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}\qquad s_y^2=\frac{\sum_{j=1}^{m}(y_j-\bar{y})^2}{m-1}.$$

Thus

$$d=\frac{\bar{x}-\bar{y}}{\sqrt{s_x^2/n+s_y^2/m}}.$$

It can be seen that the variances do not cancel out after taking the expectation of d, like in the case of equal variances. This implies that the distribution of this statistic is not a t distribution, even if the two expectations are equal.

The method is called t-test in spite of this fact, as the distribution of the statistic under the null hypothesis is approximated by the t-distribution, which has the nearest standard deviation to the standard deviation of the statistic.

In the statistical practice we get usually a t-distribution that a normally distributed quantity is divided by an empirical estimator for the standard deviation. This estimation is usually a square root of averaged squared sums. Thus in practice the t-distributions have integer degrees of freedom. But the t-distributions can be defined for any positive real number as their parameter. This allows us to find the optimum in the set of all positive numbers, resulting in the fact that the t-distributions which is used for approximating the standard deviation of d, does usually not have an integer as its 'degree of freedom'. This we shall see in the next example, where among the results of the t-test, non-integer degree of freedom appears.

Modifying the earlier example so, that we do not suppose the equality of the samples, we can apply the following command:

```
t.test(z[1:5],z[6:12],var.equal=FALSE)
```

The structure of the result is similar to the earlier one:

```
Welch Two Sample t-test
data: z[1:5] and z[6:12]
t = -0.8943, df = 4.681, p-value = 0.4148
alternative hypothesis:
difference in means is not equal to 0
95 percent confidence interval:
-0.5508027  0.2708027
sample estimates:
mean of x mean of y
9.86     10.00
```

We have 'Welch Two Sample t-test' in the title, as this version of the t-test, where samples with possibly different variances are compared is also called Welch-test. The averages of the samples have not changed. Our decision about the hypothesis, on the 5% level is again the acceptance of the null hypothesis about the equality of the expected values of the samples.

The rejection of the equality was not to be expected, seeing the result of the equal variance case. The approximated degree of freedom is always smaller than the value belonging to the case of the equal variances, which is now $n+m-2=10$. The approximated degree of freedom (which depends not only on the sizes, but on the empirical variances of the samples, too) is $df = 4.681$, and the variance of the t-distributions with smaller degree of freedom is larger, as the variance of the t_k distribution is $k/(k-2)$ (it is finite only for $k > 2$).

The formula for the degree of freedom is

$$\frac{\left(\widehat{\sigma_{\bar{x}}^2}+\widehat{\sigma_{\bar{y}}^2}\right)^2}{\frac{\widehat{\sigma_{\bar{x}}^2}^2}{n_x-1}+\frac{\widehat{\sigma_{\bar{y}}^2}^2}{n_y-1}}.$$

This is computed by the next sequence of commands, for the data we have used.

```
x<-z[1:5]
y<-z[6:12]
nx<-length(x);nx1<-nx-1
ny<-length(y);ny1<-ny-1
vx<-var(x)/nx
vy<-var(y)/ny
(vx + vy)^2/(vx^2/nx1 + vy^2/ny1)
```

The paired t-test:

```
t.test(x,y,paired = TRUE)
```

Let the bivariate observations (x_1, y_1),...,(x_n, y_n) be such that the members of the n pairs are typically dependent and correspond to a similar phenomenon, but the n sample pairs should be independent and have the same bivariate distribution.
In this case the differences $x_1 - y_1, ..., x_n - y_n$ are identically distributed and independent.

Let us suppose that the differences of the pairs have normal distribution.

The original question is, whether the expected value of the elements of the pairs is equal. But this question is obviously equivalent to the one, which states that the differences within the pair have expectation 0. The advantage of this approach is twofold: first, we usually reduce the variance by taking the pairwise differences (e.g. if the pairs refer to the same individual, then the between-individual variance is filtered out). Second, the conditions of the two-sample t-test are not fulfilled for the original samples, because of the assumed dependence between the pairs.

This hypothesis has already been investigated by the one-sample t-test (there a more general approach allowed for any given value m). We can thus solve the problem with the following statistic: If $u_j = x_j - y_j$, $j = 1, ..., n$ then the procedure of the first t-test can be applied to u:

$$d = \frac{\bar{u}}{\sqrt{\frac{\sum_{j=1}^{n}(u_j-\bar{u})^2}{n-1}/n}}.$$

This d value has t_{n-1} distribution, if the two components of the pairs have the same expected value.

Let us use the earlier data set again, like observations 1 and 7 , 2 and 8 etc. would form a pair.

This data can be understood e.g. as follows: we have checked only 6 packs, but every pack was measured twice, and the first measurements were written into the first row, and the second measurements to the second row.

Under these assumptions the paired t-test can be performed in **R** by the `t.test()` function of the `stats` package as follows:

```
z<-c(9.8, 9.7,9.9,10.4,9.5,10,
     9.9,10.1,10 ,10.1,9.8,10.1)
t.test(z[1:6],z[7:12],paired = TRUE)
```

The result:

```
Paired t-test
data:  z[1:6] and z[7:12]
t = -1.19, df = 5, p-value = 0.2875
alternative hypothesis:
difference in means is not equal to 0
95 percent confidence interval:
 -0.368677  0.135344
sample estimates:
mean of the differences
-0.116667
```

This means that although the second measurements are lighter on average by 0.117kg, we do not have to reject the equality of the two expectations. Probably because the standard deviation of the measurement differences is large, as seen from the width of the confidence interval (≈ 0.5).

3.3. Goodness of fit

In quite a few cases we have to find answers to questions, which do not fit into the methods defined in the previous part. This situation arises, when we cannot describe the null hypothesis with a few conditions for some parameters. Such questions are

- goodness-of-fit,
- the test for the homogeneity of two samples,
- test of independence.

In this part we investigate the first problem. The most frequent of its applications is the normality-check, as we have seen that the normality is quite often needed to the application of other methods.

But let us start with the general definition. We call goodness-of-fit the hypothesis test, where the null hypothesis is a given distribution. This means that we must decide, whether it can be accepted that the observations came from a given distribution. In most of the cases we do not have an exact guess for the distribution of the observed quantity, so our hypothesis is just about the family of the distributions it might belong to. But these problems can be simplified to the goodness-of-fit hypothesis for a given distribution, if the original family is a parametric one. As it is a possible way first to estimate the unknown parameters of the distribution and then check whether the observed distribution of the sample fits to this member of the hypothesized distribution family. It is true that this way we perform the estimation and the goodness-of-fit based on the same data set, which has an effect to the distribution of the test statistics. But there are proven methods to incorporate this into the decision, at least for certain statistics. First we shall deal with the most common method, the so-called χ^2-test.

3.3.a The case of known parameters

In its original form the test is suitable for discrete distributions only - but we shall see later, how to deal with continuous distributions. The general form of the χ^2 statistic is:

$$\mathtt{d} = \sum_{\mathtt{k}} \frac{(\mathtt{observed_k} - \mathtt{expected_k})^2}{\mathtt{expected_k}},$$

where k runs through all the possible cases.
First let us consider the easiest problem, where the supposed distribution is uniform on 2 points. A natural example is gender – let us go back to our data about the students. This can be solved in **R** by the simple command `chisq.test` as follows:

```
chisq.test(table(diak[,8]))
```

The result (recall that among the 47 students there were only 12 men):

```
X-squared = 11.255, df = 1,
p-value = 0.000794
```

We can see that the p-value is very small (below 0.001), so we can reject the hypothesis that in this student population the gender distribution is uniform with confidence. This means that the difference in our sample was not caused by pure chance. The result (χ^2-test object) contains the following slots:
- statistic: the value of the statistic;
- parameter: the degree of freedom;
- method: the applied method:
  (e.g. goodness-of-fit);
- observed: the observed vector of frequencies;
- expected: the expected vector of frequencies;
- residuals: the Pearson-type residuals:
  (observed - expected) / sqrt(expected).

We have seen the first three components when running the test, the others have the following structure:

```
> chisq.test(table(diak[,8]))$observed
F  M
35 12
> chisq.test(table(diak[,8]))$expected
F    M
23.5 23.5
> chisq.test(table(diak[,8]))$residuals
F         M
2.372269 -2.372269
```

So we see the observed and the expected frequencies and the calculated residuals. Their sum of squares gives the value of the test statistic. This detailed list of results is especially useful, if we are interested e.g. to find out, which cells cause the large value of the statistic. There is an interesting option, worth noting. As the χ^2 test statistic has just asymptotically a χ^2 distribution, and the convergence depends not only on the sample size, but also on the frequencies of the single cells, it may be worth using the simulated p-values:

```
  chisq.test(table(diak[,8]),
  simulate.p.value=T)

  X-squared = 11.255,
df = NA, p-value = 0.001999
```

The p-value has become larger by approximately 20% (note that the actual value depends on the random numbers in the simulation), but it of course does not change our decision of rejecting the null hypothesis, i.e. it is very likely that the deviation from the uniform distribution was not caused by pure chance.

The next example is also about a fully specified null hypothesis. Mr. Somebody has registered each day, how many advertisement phone calls he received. Summarizing the results of 100 days, he got the following table. He has received on

15	21	40	14	6	4	days
0	1	2	3	4	5	calls

This means that on the `100` days he has received altogether `187` calls and e.g. there were `4` such days when he received `5` calls. He has the hypothesis, that he is disturbed by 5 advertising companies, each of them call him independently from each other with a daily probability `33%`. Can the hypothesis be accepted?

We may formulate the hypotheses:
H_0 : the daily number of calls is `Binom(5,0.33)`
H_1 : the daily number of calls is *not* `Binom(5,0.33)`

If the calls follow indeed the distribution of H_0, then the probability of `k=0,...,5` calls per day:

$$p_k = \binom{5}{k}\left(\frac{1}{3}\right)^k\left(\frac{2}{3}\right)^{5-k}.$$

Thus during the `100` days the number n_k of such days, when he receives exactly `k` calls, for $k = 0, ..., 5$ is the following:

0	1	2	3	4	5
13.50	33.25	32.75	16.13	3.97	0.39

We can see that under the hypothesized model the expected number of days with `5` calls does not reach `1`, which is too low to apply the chi-squared distribution of the test statistic. But if we join the days with `4-5` calls, then their joint expected frequency exceeds `4`, which is considered generally enough to the application of the test. The empirical and expected frequency of the days in this case is:

	0	1	2	3	4-5
empirical:	15	21	40	14	10
expected:	13.50	33.25	32.75	16.13	4.36

These can be compared by the χ^2 statistic at the level of 5%. The value of the χ^2 statistic in this case:

$$\mathtt{d} = \sum_{\mathtt{k}} \frac{(\mathtt{observed_k} - \mathtt{expected_k})^2}{\mathtt{expected_k}} = 13.84$$

The suitable χ^2 critical value for 5-1=4 degrees of freedom at the level of 95% is 9.49, which is smaller than our value of the statistic.

- Thus the hypothesis must be rejected.
- The calls by the advertising companies do not follow the Binom(5,0.33) distribution.

The above results can be got in **R** by the following commands (unfortunately there is no general built-in simple function for this application of the χ^2-test):

```
x<-0:5 # daily calls
nx<-c(15,21,40,14,6,4) # number of days
sum(nx)  # 100 total number of days
sum(x*nx)# 187  total number of  calls

# expected number of days (binom dist).
round(v<-dbinom(0:5,5,.33)*100,2)

# joining days 4 and 5
nx[5]<-nx[5]+nx[6];(nx<-nx[1:5])
v[5]<-v[5]+v[6];v<-v[1:5]
round(v,2)

(d<-sum((nx-v)^2/v)) # 13.84
qchisq(.95,4) # 9.49
```

3.3.b The case of estimated parameters

Mr. Somebody, who gave 0.33 as the parameter of the binomial distribution, has not used his notes. Since, if we consider that he has received 187 calls during the 100 days, then this result is most likely under the binomial distribution with repetition parameter 5 if its probability parameter is 187/(5*100)=0.374.

Let us investigate our data for this hypothesis as well.

Assuming the binom(5,0.374) distribution, from the 100 days, the number of such days, when he receives exactly k calls, for $k = 0, ..., 5$ is the following:

0	1	2	3	4	5
9.61	28.72	34.31	20.50	6.12	0.73

The expected number of days with 5 calls is still too low, so we join the days of 4-5 calls again. The empirical and expected frequency of the days in this case is:

empicial:	15	21	40	14	10
expected:	9.61	28.72	34.31	20.50	6.86

The value of the χ^2 statistic for this table is d=9.54, which is still larger than the critical value of 9.49 from above. And moreover we must compensate that we have estimated the probability parameter of our hypothesized distribution from the data. So the critical value has to be taken from the χ^2 distribution with 3 degree of freedom! This means that the critical value in our case is the 95% quantile of the χ^2_3 distribution, which is 7.81. So the data does not fit to the binomial distribution even in case of an estimated parameter.

Remark

The p-value of the statistic is 2.3%. This means that if we perform the hypothesis test at the level of 1%, then we can accept the hypothesis of binomial distribution. However, in this case we apply a procedure which

- rejects a sample with 100 elements coming from a binom(5,0.384) distribution only with a probability of 1/100, but
- it accepts such distributions, which are actually not binomial, with much higher probability – i.e. this test has higher type II error probability.

3.3.c The case of the Poisson distribution

Mr. Somebody could have thought that his data follows a Poisson distribution. This can be based on the assumption that the random time intervals between subsequent calls are independent, and have identical exponential distribution. If we estimate the parameter of the Poisson distribution, which coincides with the expected value, then we get from our data that $\widehat{\mu}$=1.87. If we consider the frequencies as Poisson($\widehat{\mu}$), then the days with calls 0,...,5+ have expected frequencies in 100 days as the following table shows.

0	1	2	3	4	5+
15.41	28.82	26.95	16.80	7.85	4.17

If we compare this table with the empirical frequencies by the χ^2-test, then we get d=9.37, which is smaller than the critical value, based on the χ^2_4 distribution for the level of 5%, which is 9.49. This implies that the hypothesis of the Poisson distribution can be accepted at the 5% level.

We can compute the above results in **R** by the help of the following commands:

```
x<-0:5# number of daily calls
nx<-c(15,21,40,14,6,4)# obs.freq.

(m<-sum(x*nx)/100) # expectation

v<-dpois(0:4,m)*100 # exp.freq.
round(v<-c(v,100-sum(v)),2)# the last

(d<-sum((nx-v)^2/v)); qchisq(.95,4)
```

Remark

We had to apply the quantile of the χ^2 distribution with 4 degree of freedom because we have compared the frequencies of 6 cells, but one parameter of the fitted distribution was estimated from the data, and 6-1-1=4.

3.3.d The case of continuous distributions

However, in most of the cases the binning of our data is not so obvious as it was for discrete distributions. If we want to fit a continuous distribution (typically the normal), then we must bin the data. In this case we must take care on the fact that into every interval must fall some values (generally this means at least five). This can be achieved the easiest if we choose equiprobable intervals with respect the distribution to be fitted. This ensures that at least the theoretical (expected) frequencies will be equal in the classes.

The test can be constructed based on the following steps (for the number of intervals there is no unique optimal value: some programs use $\log_2(n)$, others simply use 10 as default value with the condition that the classes with observed frequencies below 5 are joined).

- The determination of the number of classes (n).
- The determination of the endpoints by using suitable quantiles.
- The computation of the number of observations in the classes.
- The estimation of the parameters of the fitted normal distribution.
- The computation of the expected frequency of the classes under the estimated normal distribution.
- Computation of the test statistic.
- Determination of the p-value.

Our student data set is rather small, so we work with the smallest number of classes, that gives positive degree of freedom after having estimated the parameters of the normal distribution. As it has 2 parameters, the minimal number of classes is four.

```
  n <- 4;j <- 5
  # number of intervals and the variable
  q <- c(1:n)/n
  h <- c(-Inf,qnorm(q))
  mu <- mean(diak[,j])
  sig <- sd(diak[,j])
  h <- h*sig+mu
  # these are the limits
nu <- rep(0,times=n)
  for (i in 1:n)
nu[i]<-
  sum((diak[,j]<h[i+1]) & (diak[,j]>=h[i]))
  # expected frequency:
  np <- sum(nu)*rep(1/n,times=n)
  chi<-sum((nu-np)^2/np)
  # the p-value:
p <- 1-pchisq(chi,n-3)
p
```

The result is $p = 0.23$, and it shows that the height can be accepted as normally distributed, in spite of its obvious mixture–type appearance (as the height of men and women is different).

The above reasoning implies that the χ^2 test is suitable for fitting continuous distributions, but the method is not ideal: the results may depend on the binning. We shall come back to this issue later.

3.4.The test of independence

The probabilistic independence of two random quantities mean that the distribution of one does not depend

on the actual value of the second. The test of independence as a method is the statistical investigation of the question about the probabilistic independence of two or more random quantities.

In this part we shall deal with the test of independence, when the observed units can be categorized with respect of two (or more) points of view, and each unit can have finitely many possible types according to all of the points of view. The observations for such units are typically summarized in bivariate contingency tables. So we shall deal with contingency tables and investigate if the properties of the two points of view are independent.

Let us suppose that the variables have r and c possible values respectively. The data is given corresponding to the possible value pairs in a table of size $r \times c$, where the element in the ith row and jth column is $n_{i,j}$. This means that we had $n_{i,j}$ such cases, when for the observed unit the first variable had its ith possible value and the second had its jth possible value.

There are quite a few methods for checking the independence of the two classifying variables. We have some classic measures, including the correlation and multivariate techniques like different correspondence models. Here we show just the most classical method, the χ^2 independence test.

The χ^2 independence test is in fact a special goodness-of-fit with estimated parameters. Since

- first we estimate by the sample, what are the probabilities of the possible values of the first property and then the same for the types with respect of the second property.
- Then – supposing that the two properties are independent – we say that the joint occurrence of the pairs must be equal to the product of these two estimated probabilities, multiplied by the number of observations. Thus we give an estimation for the values of the frequency table, under the hypothesized independence of the two properties.
- Finally, we investigate if the difference between the estimated and the observed frequency table is not too large to the acceptance that these two are essentially equal.

The procedure is called χ^2 independence test as the distance between the two tables is measured with the help of the χ^2 statistic, which is indeed approximately χ^2 distributed if the independence holds true and the number of observations is not too small in each cells.

Some notations, helping the description:

The total number of observations

$$n = n_{+,+} = \sum_{i=1}^{r} \sum_{j=1}^{r} n_{i,j}.$$

If $n_{i,+} = \sum_j n_{i,j}$ and $n_{+,j} = \sum_i n_{i,j}$, i.e. $n_{i,+}$ and $n_{+,j}$ are the row and column sum of the frequency table, respectively, then an estimator of the row-wise distribution:

$$(n_{1,+}/n, ..., n_{r,+}/n),$$

and the same for columns:

$$(n_{+,1}/n, ..., n_{+,c}/n).$$

Thus, if for $n = n_{+,+}$ observations the two properties are independent from each other, then the expected number of observations falling into the cell (i, j) can be estimated by

$$\widehat{n_{i,j}} = n \, \frac{n_{i,+}}{n} \, \frac{n_{+,j}}{n}.$$

Thus to decide about the independence of the two properties we must check if the observed $n_{i,j}$ table is not too far from the table $\widehat{n_{i,j}}$, got by the independence hypothesis.

It is known that the

$$d = \sum_{i,j} \frac{(n_{i,j} - \hat{n}_{i,j})^2}{\hat{n}_{i,j}}$$

distance between the two tables can be approximated by the $\chi^2_{(r-1)(c-1)}$ distribution, where r and c are the number of rows and columns, respectively, if the independence holds true and the number of observations is not too small in each cells.

First we show, how can we carry out this hypothesis test by the help of the basic formulas of **R**. Next we turn to the question of using the `chi.sq()` function of package `stats`.

So let us suppose that it is given the `D` frequency table. Our task is to investigate the following pair

of hypotheses for the two categorical variables, upon which the table is based:

H_0: are independent from each other
H_1: are *not* independent from each other

3.4.a Calculations by the basic operations

Let us use the data, which is the first example in the help of the function `chi.sq()`.

```
rm(list=ls())
D <- cbind(c(762, 327, 468),
c(484, 239, 477))
dimnames(D) <- list(
     party  = c("Democrat",
                "Independent",
                "Republican"),
     gender = c("F","M")  )
D
(n<-sum(D))
```

As we see, the table contains gender-wise US-data of party-preferences, based on altogether 2757 observations.

The following commands calculate the value of the above `d` statistic and the corresponding χ^2_2 critical value. We use the χ^2_2 distribution, as the size of the table is 3×2, i.e. in case of independence the degree of freedom of the approximating χ^2 distribution to `d` is `2=(3-1)(2-1)`.

```
rp <- rowSums(D)/n
cp <- colSums(D)/n
E <- n*rp%o%cp
(d <- sum((D-E)^2/E)) # 30.07
qchisq(0.95,2) # 5.99
```

As we see from the result of the calculations, the value of `d` is `30.07` and the threshold value for accepting H_0 is `5.99`. Thus the value of the statistic fall into the level `5%` critical region, so by the χ^2 test we must reject the independence of the gender and the party-choice.

The result is not surprising, if we check the table thoroughly.

```
            gender
party         F   M
Democrat    762 484
Independent 327 239
Republican  468 477
```

As it can be seen, there are many more women (column `F`) then men, but the Republican party was chosen by more men than women.

The above-mentioned reason for non-independence in this form is just a first guess. As from the point of independence not the frequencies, but the relative frequencies and their ratio is relevant. More exactly we must investigate that e.g. it is true that in the three rows of the table the women/men ratio is approximately equal. Or equivalently the fact that the two distributions over three possible values, defined by the data of the first and the second column can be considered as equal.

3.4.b Application of `chisq.test()`

We have seen above, how to check the independence for our table manually, but for the more detailed results (e.g. residuals) the statistic of the `stats::chi.sq()` function might be useful. The χ^2 test of independence can be run by the `chisq.test` command, if previously a contingency table was already created from the data.

If we run the χ^2 test of the `stats` package with the parametrization `chisq.test(D)`, then we just get the following lines:

```
Pearson's Chi-squared test
data:  D
X-squared=30.07, df=2, p-value=2.954e-07
```

In this printout only the p-value of the `d` statistic is new. But we can calculate this ourselves by the `1-pchisq(d,2)` command. The printout is so short because from the object of type `htest` given as a result of the `chisq.test()` the method `print.htest()` writes only these parts.

The next commands save the object, containing the result of the `chisq.test()`. Later we show, how to compute the elements of this object. Finally, we show how to use the elements of this result object to analyse the frequency table.

We repeat some earlier commands in order to increase the clarity of our explanations.

```
w <- chisq.test(D)
class(w);str(w,give.a=FALSE)
n<-sum(D)
rp<-rowSums(D)/n;cp<-colSums(D)/n
```

```
E<-n*rp%o%cp
V<-n*(rp*(1-rp))%o%(cp*(1-cp))

# chi2 statistic
(d<-sum((D-E)^2/E))

# p-value of the statistic
1-pchisq(d,prod(dim(D)-1))

w$observed-D
w$expected-E

# components of the chi2 statistic
w$residuals-(D-E)/sqrt(E)

# the standardized errors per cell
w$stdres-(D-E)/sqrt(V)
```

The result of the `str(w)` structure is the following print:

```
[1] "htest"
List of 9
$ statistic: Named num 30.1
$ parameter: Named int 2
$ p.value  : num 2.95e-07
$ method   : chr "Pearson's Chi-sq ...
$ data.name: chr "D"
$ observed : num [1:3, 1:2] 762 327 ...
$ expected : num [1:3, 1:2] 704 320 ...
$ residuals: num [1:3, 1:2] 2.199 ...
$ stdres   : num [1:3, 1:2] 4.502 ...
```

So the earlier calculations present every part of the result object.

The results of the computations can be explained as follows. In the result object the element

- `$observed` is the same as the processed `D` table
- `$expected` is the same as the table, which we got from the table `D` by estimating the number of observations in the cells by assuming the independence of the two components, which create the table
- `$residuals` are squared and summarized for getting the statistic `d`, used for independence test i.e. giving the χ^2 statistic. This table is an important tool to decide, which cells of the table show substantial deviations from the hypothesized independence. These cells might have decisive contribution for the statistic `d` to exceed the acceptance threshold.
- `$stdres` contains the standardized values, assuming that the individual cells of the table follow the binomial distribution with parameters `n` as observations and $\widehat{n_{i,j}}/n$ probability – which is the case if the independence holds.

Let us check, what do the `$residuals` and `$stdres` tables give in our case.

The data in the `$residuals` table printed by the `round(w$residuals,3)` command:

	gender	
party	F	M
Democrat	2.199	-2.505
Independent	0.411	-0.469
Republican	-2.843	3.239

It can be seen that the largest additive component of the `d` statistic is provided indeed by the republican men. If we let print out the `w$residuals**2` table as well, then it can be seen that this cell, together with the cell of the necessarily under-represented republican women give almost two third (`18.57`) of the total `d` statistic (`30.07`).

It should be mentioned that as it can be seen from the results of the `chisq.test(D[-3,])` statistic, the sub-table formed from the first two rows of the original table can be considered as being independent (p-`value=0.19`). Thus, if we investigate just the question of gender-dependence of preferences only for democrats-independents, then the independence of the party-preference and gender can be accepted.

The data of the `$stdres` table printed by the `round(w$stdres,3)` command:

	gender	
party	F	M
Democrat	4.502	-4.502
Independent	0.699	-0.699
Republican	-5.316	5.316

Here we have divided (standardized) the difference between the observed frequencies `D` and the estimated frequencies under independence `E` with the standard deviation we get for the elements of `D`, as the standard deviation of the binomial distribution, which has as probability parameter the product of the two estimated probabilities defining the cell.

This can equivalently calculated as $\sqrt{n}$ times the product of the standard deviation of the two Bernoulli random elements of the respective row and column. As if the probability of the property in the ith row of the table is $p_{i,+}$, the same for the the property in the jth column is $p_{+,j}$, then the standard deviation of the indicator of these properties

is $\sqrt{p_{i,+}(1-p_{i,+})}$ respectively $\sqrt{p_{+,j}(1-p_{+,j})}$, and the standardization factor is $\sqrt{n}$ times the product of these standard deviations.

Thus the elements of the `$stdres` table follow approximately the standard normal distribution. The numbers of the previous table, which are larger than `3` implies that for the republican cells, but also for the two democrat cells there is a substantial deviation from the independence.

Some remarks about the table `$stdres`:

- From the above print it is clear that in the rows there are numbers with the same absolute value, but with different signs. This is always necessary in case of tables with only two columns. We could see the same for tables with just two rows for the columns. For tables with more rows and columns the results are usually different – even that is not true that the sum of the elements in rows or columns is `0`, as seen in our cases (for the columns).

In case of two cells the same absolute value is caused by the fact that the probabilities of an observation falling into the first or second column are complement to each other. This implies that the sum of the two columns of the error matrix `D-E` is 0, and that causes that in the matrix `V` the error variances are the same within a row.

- When evaluating the normalizing matrix `V`, we must take into account that $n_{i,j}$ is the sum of n independent $\texttt{Bernoulli}(\texttt{p}_{\texttt{i},+}\texttt{p}_{+,\texttt{j}})$ random quantities, where these Bernoulli variables are the product of a $\texttt{Bernoulli}(\texttt{p}_{\texttt{i},+})$ and a $\texttt{Bernoulli}(\texttt{p}_{+,\texttt{j}})$ quantities, but the formula $D^2(XY) = D^2(X)D^2(Y)$ is valid even for independent variables X and Y only if these have 0 expected values, like `D-E` in our case.

Let us see some examples from our student data base as well.

```
table(diak[,6],diak[,8])
chisq.test(table(diak[,6],diak[,8]))
chisq.test(table(diak[,6],
diak[,8]),simulate.p.value=T)
```

This is a typical case, when the use of simulated p-values is recommended, due to the lot of cells, which are empty or contain just a few observations. Of course both options reject the independence of shoe size and gender. The printed result:

```
    F M
36 3 0
37 7 0
38 9 0
39 7 0
40 7 0
41 1 2
42 1 1
43 0 5
44 0 2
45 0 1
46 0 1
X-squared = 40.864, df = 10,
p-value = 1.193e-05
X-squared = 40.864, df = NA,
p-value = 0.0004998
```

In the next example we investigate the dependence between beer consumption and gender.

```
table(diak[,3],diak[,8])
chisq.test(table(diak[,3],diak[,8]))
chisq.test(table(diak[,3],
diak[,8]),simulate.p.value=T,B=10000)
```

The result:

```
      F  M
0    28  3
1     1  3
2     1  2
3     1  0
4     2  0
5     1  0
6     0  1
15    1  0
18    0  2
115   0  1
X-squared = 25.297, df = 9,
p-value = 0.002659
X-squared = 25.297, df = NA,
p-value = 3e-04
```

Now the difference between the p-values of the two methods is even larger. Let us see, by how much they differ, if we apply the recommended join of categories. The number of replicates has been increased to 10000 in order to ensure that there will be indeed higher values among the simulated data than that of the observed sample.

```
sor_2<-diak[,3]>0
table(sor_2,diak[,8])
chisq.test(table(sor_2,diak[,8]))
chisq.test(table(sor_2,
diak[,8]),simulate.p.value=T,B=10000)
```

```
sor_2  F  M
FALSE 28  3
TRUE   7  9
X-squared = 9.7141, df = 1,
p-value = 0.001829
X-squared = 12.039, df = NA,
p-value = 0.0011
```

The p-values are near to each other, they are between the two values, seen in the table above. These look as being real for this 2×2 table.

If we want to get an even more exact result, we may apply the so-called Fisher-exact test, which gives the exact probability for getting the observed or even larger value under the null hypothesis.

```
fisher.test(table(sor_2,diak[,8]))
```

The result is more detailed in this case:

```
p-value = 0.001058
alternative hypothesis:
true odds ratio is not equal to 1
95 percent confidence interval:
2.110881 81.838252
sample estimates:
odds ratio
11.18708
```

The p-value is very near to the ones previously seen. It is worth noting, however that for such small sample sizes the test is conservative: it gives slightly higher p-values, as the current observed value has positive probability, which is counted to the rejection region.

We may see a simple example for this phenomenon.

Let us suppose that out of 8 boys and 8 girls some do sport regularly. Let us see, what are the p-values for different gender–distributions.

```
p<-rep(0,times=4);
for (i in 1:4)
{x1<-c(i,8-i);
x2<-c(8-i,i);
p[i]<-round(
fisher.test(rbind(x1,x2))$p.value,3)}
p
```

```
[1] 0.010 0.132 0.619 1.000
```

It can be seen that there is no test for the common first type error probability of $\alpha = 0.05$, after $\alpha = 0.01$ (which rejects, if the distribution is 0 vs. 8) we immediately get $\alpha = 0.13$. Mathematically the randomization may be a solution, but in practice it may be preferable to take new sample elements in order to decide about these questions in the borderline cases, like the two-sample plans in industrial statistics.

3.5.The test of homogeneity

The third application area of the χ^2 test is testing homogeneity. To apply this method, we need just two vector of frequencies, and the method is the same as in case of the independence test. So as a last remark, we rather show the following feature of **R**, which allows us to define the distribution with the help of frequencies, just the fact has to be indicated that these values must be transformed into probabilities.

```
x <- c(39,37,30,28,2,25)
p <- c(20,20,20,20,10,20)
chisq.test(x, p = p, rescale.p = TRUE)
```

The result has the well-known format of the χ^2 test:

```
X-squared = 16.879, df = 5,
p-value = 0.004735
```

3.6. Closing remarks to the chi^2 test

The χ^2 distribution of the `d` statistic is just an approximation even in the case of independence. The statistic was not created as the squared sum of independent normal variables, but dependent binomials. Thus the approximation of the `d` statistic with the χ^2 distribution can only be accepted if we have enough observations in the cells.

The notion of 'enough' was determined by simulations, and the result was *that the number of expected observations*

- should be at least `4` if the level is `5%`,
- should be at least `6` if the level is `1%`,

The `stats::chi.sq()` program was written on a way that the program gives a warning if there is a cell, where the number of expected observations is less than 4. In this case the already introduced simulated critical-value calculation is recommended, as we have done it already - e.g. analogously to the next command.

```
chisq.test(table(diak[,8]),
simulate.p.value=T)
```

3.7. Statistics based on ranks

The Wilcoxon signed-rank test can also be applied to the comparison of two paired samples, or to test the symmetry with respect to a given value. It can also be considered as an alternative to the t-test in cases, when there are outliers in the data, causing that the sample cannot be considered as normally distributed. This results in a low power of the t-test.

The Wilcoxon test does not consider the numeric value of the sample elements, only their rank in the ordered sample.

To be more specific, let the sequence of the differences from a paired sample be $X_i - Y_i$ (or the sequence $X_i - \mu$ if we test the symmetry with respect to μ). Let us suppose that there are no 0 differences (i.e. that there are no ties in the sequences). Ordering the sequence $|X_i - Y_i|$, let us compute the sum of ranks of those, which were positive originally. (The rank is the index in the ordered sample.) Computing the same for the negatives, the difference of these two statistics is denoted by W.

Under H_0 (the samples have the same distribution) we can give the limit distribution of W as normal with expectation 0 and variance

$$n(n+1)(2n+1)/6.$$

The test can be used with different options for small sample sizes the most important one is the choice exact=T , which results that the p-values are based on the exact distribution under H_0 (which can be computed by recursion) rather than on the limit distribution.

If our task is to test the symmetry of the drunken beer amount, then this method is not suitable, as it cannot be carried out if we have multiple values among the sample elements. In this case the ranks of the equal values is substituted by their average, but the many equal values may substantially distort the variance of the statistic. It can be taken into account for example by using the methods described in the book of Hollander-Wolfe (1973).

```
wilcox.test(diak[,3], mu=mean(diak[,3]))
```

The results are self-explaining: while the t-test cannot prove the asymmetry, here we got it with high significance.

```
V = 190, p-value = 4.172e-05
```

For the case of paired observations, there is just one difference: we give two samples and then there is no need for specifying the mu.

Let us recall the other data set we have already used in the first Section about annual maxima of water level in Budapest and Mohács.

The application of the Wilcoxon test here is quite natural, as here we cannot suppose the normality of the observations. We may observe that here it may be worth applying the one-sided alternative (even without knowing the data, we may suppose that water levels are higher downstream, due to the natural additional affluents, at least if the gauges are set at comparable levels).

In the first command we have not applied any corrections, but in this case it was not realistic, as in practice the 0 points of the gauges are chosen arbitrarily. In the second try we have checked the differences from the medium water level and the result was just not significant as the following lines show.

```
wilcox.test(mohmax,budmax,alternative="g")
W = 3287.5, p-value = 1.766e-09

wilcox.test(mohmax-median(moh),
budmax-median(bud),alternative="g")
W = 2368.5, p-value = 0.06362
```

The Wilcoxon test, applied to the case of two independent samples is often called Mann-Whitney test. In this case we compare all the pairs $(X_i, Y_j)_{i=1,j=1}^{n,m}$ (or order the joined sample (X_i, Y_j) with $n+m$ elements) and the following formula gives the test statistic:

$$\sum_{j=1}^{m} \#\{X\{'s\} < Y_j\}$$

which has the following parameters under H_0: the expected value is $nm/2$ and the variance $nm(n+m+1)/12$.

Let us use the beer consumption variable of our data set once more and check if we can get significant results for the gender differences by the known tests.

```
fs<-diak[diak[,8]=="F",3];
ns<-diak[diak[,8]=="M",3]
wilcox.test(fs,ns)
```

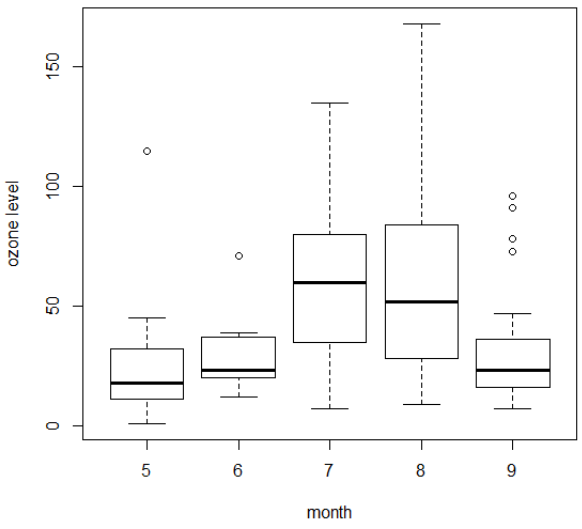
The result:

```
W = 94.5, p-value = 0.0008862
t.test(fs,ns)
t = -1.348, df = 11.055, p-value = 0.2046
```

These results also support the Wilcoxon test: while the t-test was not able to detect the difference between the beer consumption of the genders, it gave a clearly significant result. But this can of course only be true for the non-normally distributed samples, as it is well-known that for normal samples the t (u) tests are the most powerful.

One of the sample data sets of **R** also shows an interesting application:

```
boxplot(Ozone ~ Month,
data = airquality,
xlab="month",ylab="ozone level")
wilcox.test(Ozone ~ Month,
data = airquality,
subset = Month %in% c(5, 8))
```



Here we have used the interface called `formula`, taking care that exactly two groups were considered.

If we have more, non-normally distributed groups, then we may use the Kruskal-Wallis-type nonparametric analysis of variance. The above figure also supports that there is indeed a substantially looking difference between the months.

3.8. ROC curve

In many cases (especially in the medical or the social sciences) it is customary to measure the goodness of the methods by the so-called ROC-curve. This approach can also be used for defining the most appropriate critical region.

In order to understand the notion, let us suppose the following simple hypothetical situation, where the doctor knows exactly which patients are ill (e.g. by another established diagnostic method). These patients are called positive (in accordance to the medical terminology), and their status is denoted by 1. The negative cases are denoted by 0.

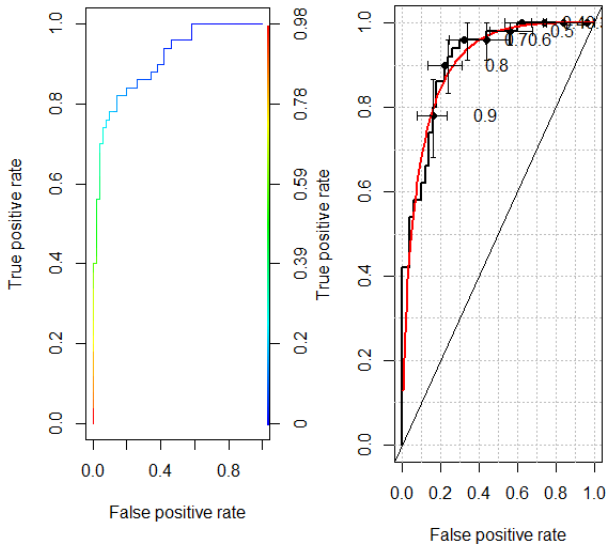
We suppose that the method under investigation orders to each patient a probability that he or she is positive. In a statistical hypothesis setting this is practically $1 - p$ (where p is the p-value). Based on these probabilities, a classifier can be defined as a threshold (those patients are declared positive, who have higher probability than the threshold). In the next part we give some typical graphical methods for presenting the results.

The ROC (Receiver Operating Characteristic) curve relates the proportion of correctly positively classified cases (True positive rate, TPR) to the proportion of erroneously positively classified cases (False positive rate, FPR). It can be expressed as:

$$(FP/N; TP/P)$$

where P and N denote the number of positive and negative cases, respectively. Each point on the curve represent a decision rule, out of which we can choose according to our preferences. It is obviously the best to choose a procedure as near to the upper-left corner of the unit square as possible, as this misclassifies the least number of cases.

It is seen on the figure, where the first panel shows just the curve itself – for a simple simulation scheme –, with a nice colouring scheme, while the second is based on the built-in function, and it also includes the diagonal (which belongs to an uninformative decision system) and a smooth curve, which is the theoretical ROC curve in the case, when both of the populations have normal distribution with the parameters of the observed samples.



To produce the first curve we need the package `ROCR`, which contains several other useful tools, which we shall use in the next paragraphs. The second curve was got by the package `verification`, which allows for calculating the simulated confidence regions, as shown by crosses on the previous figure.

```
nn<-50
x <- matrix(0,100,20)
t <- rep(0,times=100)
seed <- 12345
val <- c(rep(0,times=50),rep(1,times=50))
for (i in 1:50) x[i,] <- rnorm(20)
for (i in 1:50) x[i+50,]<-rnorm(20,0.5,1)
for (i in 1:100)
  t[i] <- t.test(x[i,])$p.value

library(ROCR)
pred <- prediction(t,val,
label.ordering=c(1,0))
perf <- performance(pred,"tpr", "fpr")
sum(t[val==0]<0.5)/nn
sum(t[val==1]<0.5)/nn
```

```
library(verification)
# CI by bootstrapping and fitted curve

par(mfrow = c(1,2))
plot(perf,colorize = TRUE)
roc.plot(val,1-t,main = NULL,
         xlab = "False positive rate",
         ylab = "True positive rate",
         CI = T, n.boot = 100,
         plot = "both", binormal = TRUE)
```

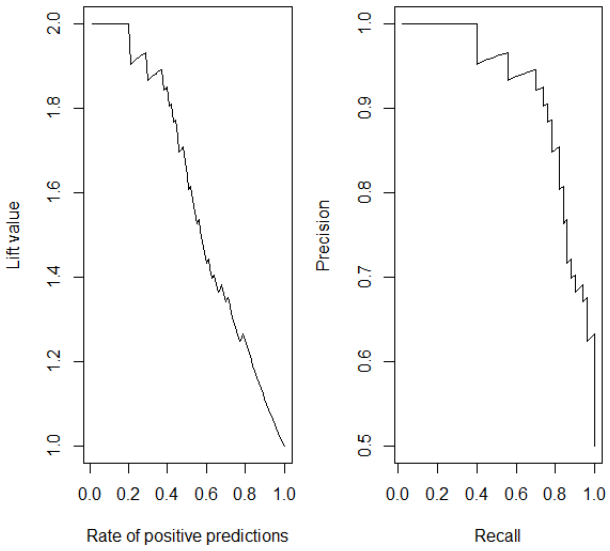
Accuracy is a similar notion, defined by the formula

$$\frac{TP+TN}{P+N}.$$

Finally the lift values are also shown, which is formally the lift of such a probability that we categorize an observation as positive if it is indeed positive: $P(\hat{Y}=1|Y=1)/P(\hat{Y}=1)$, which can be estimated as

$$\frac{TP/P}{(TP+FP)/(P+N)}.$$

It is analogous to the commonly used diagram of precision vs. recall (this is defined as $\frac{TP}{TP+FN}$), as it is seen on the second panel of the next figure.



The figure was created by the following code.

```
par(mfrow = c(1,2))
plot(performance(pred,
     measure = "lift", x.measure="rpp"))
plot(performance(pred,
     measure = "prec", x.measure="rec"))
```

3.9. Simulations

Also for the parametric tests seen earlier, such questions may emerge, for which there are no theoretically sound answers.

- Is the sample size big enough in order the u/t test to be applicable?

- What is the real level of the test if the normality is not exactly fulfilled?

In the next paragraphs we consider such a case, when the sample is not normally distributed, to be more specific, it has exponential distribution. The sample size varies between 20 and 2000, and we investigate the first type errors. In the simulations we recommend to set the seed of the random number generator at the beginning, as this way the results will be reproducible.

```
n <- c(20,40,80,150,300,500,1000,2000)
k <- 10000
ered <- matrix(10,k,length(n))
seed<-12345
for (i in 1:length(n))
  {for (j in 1:k)
     {xdat<-rexp(n[i]);
      ered[j,i]<-t.test(xdat-1)$p.value}}

er5<-rep(0,times=length(n))
for (i in 1:length(n))
  er5[i]<-sum(ered[,i]<0.05)/k
round(er5,3)

er1<-rep(0,times=length(n))
for (i in 1:length(n))
  er1[i]<-sum(ered[,i]<0.01)/k
round(er1,3)
```

The code above generates 10000 simulated standard exponential samples to each of the sample sizes and stores the p-values of the t-tests. It is easy to determine the empirical first-kind error probabilities,

based on these values.

It can be seen that for smaller sample sizes there is a substantial difference from the nominal (expected) p-values and this deviation decreases to below 10% only for sample sizes larger than 500.

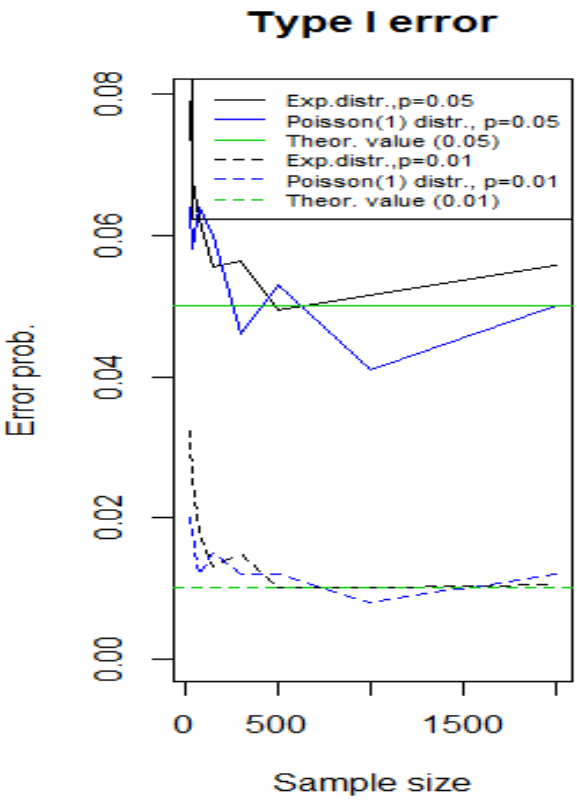
In the next part we investigate the power function: besides the case of $\lambda = 1$ (not only for the exponential, but also for the Poisson case), which corresponds to H_0, also distributions with $EX = 1.1$ are considered.

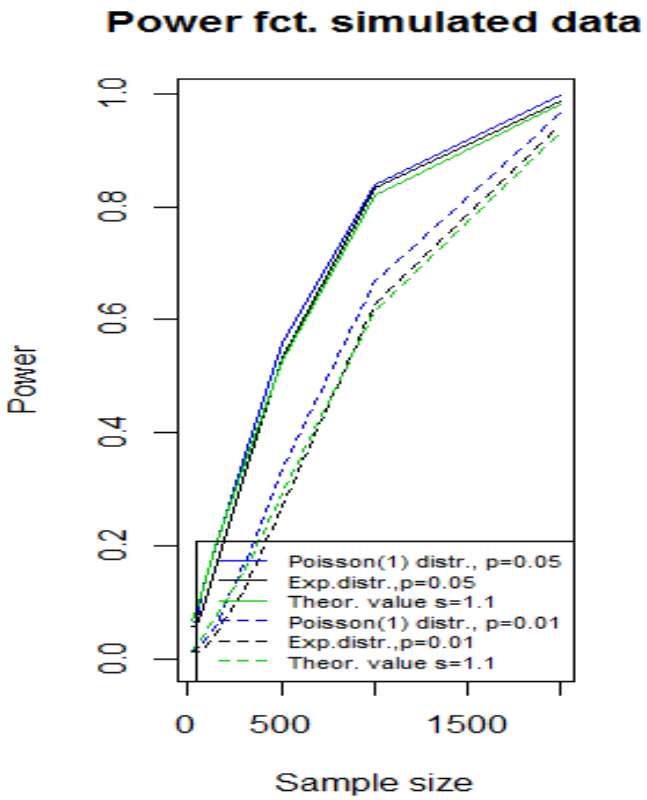
First we give the code, then the resulting figure.

```
ered<-matrix(10,k,length(n))
seed<-12345

for (i in 1:length(n))
  {for (j in 1:k)
    {xdat<-rexp(n[i],1/1.1);
     ered[j,i]<-t.test(xdat-1)$p.value}}

ero5 <- rep(0,times=length(n))
for (i in 1:length(n))
  ero5[i] <- sum(ered[,i]<0.05)/k

ero1 <- rep(0,times=length(n))
for (i in 1:length(n))
  ero1[i] <- sum(ered[,i]<0.01)/k

# Poisson
n<-c(20,40,80,150,300,500,1000,2000)
k<-1000
ered <- matrix(10,k,length(n))
seed <- 12345

for (i in 1:length(n))
  {for (j in 1:k)
     {xdat <- rpois(n[i],1);
      ered[j,i]<-t.test(xdat-1)$p.value}}

erp5 <- rep(0,times=length(n))
for (i in 1:length(n))
  erp5[i] <- sum(ered[,i]<0.05)/k

erp1 <- rep(0,times=length(n))
for (i in 1:length(n))
  erp1[i] <- sum(ered[,i]<0.01)/k

ered <- matrix(10,k,length(n))
seed <- 12345

for (i in 1:length(n))
  {for (j in 1:k)
     {xdat <- rpois(n[i],1.1);
      ered[j,i]<-t.test(xdat-1)$p.value}}

erop5 <- rep(0,times=length(n))
for (i in 1:length(n))
  erop5[i]<-sum(ered[,i]<0.05)/k

erop1 <- rep(0,times=length(n))
for (i in 1:length(n))
  erop1[i] <- sum(ered[,i]<0.01)/k


par(mfrow=c(1,2))
erelm <- ero5
for (i in 1:length(n))
  erelm[i] <-
  power.t.test(n=n[i],delta=0.1,sd=(1.1),
               sig.level=0.05,
               type="one.sample",
               alternative="two.sided",
               strict = TRUE)$power
erelm1 <- ero5
for (i in 1:length(n))
  erelm1[i] <-
  power.t.test(n=n[i],delta=0.1,sd=(1.1),
               sig.level=0.01,
               type="one.sample",
               alternative="two.sided",
               strict = TRUE)$power

plot(n,er5,type="l",ylim=c(0,max(er5)),
     main="Type I error",
     xlab="Sample size",
     ylab="Error prob.")
lines(n,er1,lty=2,col=1)
lines(n,erp5,lty=1,col=4)
lines(n,erp1,lty=2,col=4)
abline(h=0.05,col=3)
abline(h=0.01,col=3,lty=2)

legend("topright",
       c("Exp.distr.,p=0.05",
         "Poisson(1) distr., p=0.05",
         "Theor. value (0.05)",
         "Exp.distr.,p=0.01",
         "Poisson(1) distr., p=0.01",
         "Theor. value (0.01)"),
       lty=c(1,1,1,2,2,2),
       col=c(1,4,3,1,4,3),cex=0.7)
```

```
plot(n,ero5,type="l",ylim=c(0,max(ero5)),
     main="Power fct. simulated data",
     xlab="Sample size",ylab="Power")
lines(n,ero1,lty=2,col=1)
lines(n,erop5,lty=1,col=4)
lines(n,erop1,lty=2,col=4)
lines(n,erelm,col=3)
lines(n,erelm1,col=3,lty=2)
```

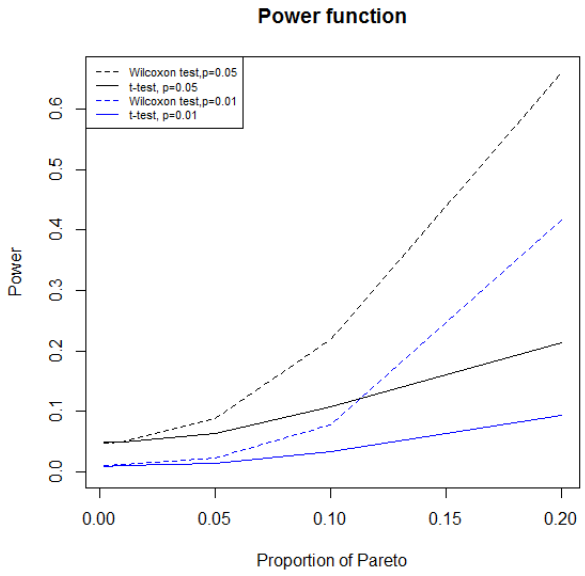
```
legend("bottomright",
       c("Poisson(1) distr., p=0.05",
         "Exp.distr.,p=0.05",
         "Theor. value s=1.1",
         "Poisson(1) distr., p=0.01",
         "Exp.distr.,p=0.01",
         "Theor. value s=1.1"),
         lty=c(1,1,1,2,2,2),
         col=c(4,1,3,4,1,3),cex=0.7)
```





The theoretical value was got from the `power.t.test` command of **R**. It can be seen from the figure, that for the exponential case we might need even 1000 observations in order to reach the expected levels, for the Poisson case fewer are enough to it. The really large deviations belong to the cases with small sample sizes and small error probabilities: in this case we may get even 2-3 times more false rejection probability.

In the next part we investigate, in which cases it is worth applying the Wilcoxon test instead of the two-sample t-test. One of the samples is normal, with expectation $\mu = 1.6$ and variance $\sigma^2 = 0.06$, the other is a mixture of the same normal distribution and a Pareto distribution with shape $\alpha = 3$. In the simulation we investigate the effect of increasing the Pareto-part on the power functions of the two tests. We may conclude that as we move away from the normal case, the more effective the Wilcoxon test is as the next figure shows.



Finally, let us see the program
that generated the above output.

```
al<-3
p <- c(0.002,0.01,0.05,0.1,0.2)
k <- 10000
n <- 500
eredt <- matrix(10,k,length(p));
eredw <- eredt

seed<-12345
for (i in 1:length(p))
  {for (j in 1:k)
    {xdat2 <- rnorm(n,al/(al-1)+.1,
                    sqrt(6/100));
     xdat1 <- rnorm(n,al/(al-1)+.1,
                    sqrt(6/100));
     xdat1[1:(n*p[i])]<-
                (1/runif(n*p[i]))^(1/al);
     eredt[j,i]<-
     t.test(xdat1,xdat2)$p.value
     eredw[j,i] <-
       wilcox.test(xdat1,xdat2)$p.value}}

ert<-rep(0,times=length(p))
for (i in 1:length(p))
  ert[i] <- sum(eredt[,i]<0.05)/k

erw<-rep(0,times=length(n))
for (i in 1:length(p))
  erw[i] <- sum(eredw[,i]<0.05)/k

ert1<-ert
for (i in 1:length(p))
  ert1[i] <- sum(eredt[,i]<0.01)/k

erw1<-ert
for (i in 1:length(p))
  erw1[i] <- sum(eredw[,i]<0.01)/k

par(mfrow=c(1,1))
plot(p,ert,type="l",ylim=c(0,max(erw)),
         main="Power function",
         xlab="Proportion of Pareto",
         ylab="Power")
lines(p,erw,lty=2,col=1)
lines(p,ert1,lty=1,col=4)
lines(p,erw1,lty=2,col=4)

legend("topleft",
       c("Wilcoxon test,p=0.05",
         "t-test, p=0.05",
         "Wilcoxon test,p=0.01",
         "t-test, p=0.01"),
       lty=c(2,1,2,1),
       col=c(1,1,4,4),cex=0.7)
```

IV. Linear regression in **R**

The regression models explain the value of one target variable by the values of some explanatory variables. The role of statistics in regression modelling is three-sided: first fitting the model, then checking the model validity, finally interpreting the meaning of the constructed model. In this short section we discuss only the linear regression models fitted by method of least squares. This model fitting technology is comfortable, it has a nice geometrical interpretation, and the related statistics have a natural descriptive explanations. Besides a short theoretical introduction we will explain the use of the `stats::lm()` method, and comment its results.

4.1. Linear regression using the `stats::lm()` function

We demonstrate the regression modelling at first on the data sets `LifeCycleSavings` and `trees`. The application of these data sets facilitate that either are element of `datasets` package, which is a basic supplementary package of the **R** system. Hence to use the data sets of this package it is not necessary to install or load the `datasets` package. Namely the `datasets` package is one of the `27` packages installed together with the **R** core program, and it is also one of the `7` packages which are loaded at the start of the graphical **R** console, the `Rgui`. The `?LifeCycleSavings` and `?trees` commands provide facilities to gain more information on the ancestry and explanation of these data sets.

The `trees` is a `data.frame` of size 31×3. Its three variables are '`Height`', '`Girth`' and '`Volume`', for `31` black cherry trees. This data set originates from the handbook of the Minitab statistical program system, (Minitab Student Handbook, 1976, pp 274).

The `LifeCycleSavings` is a data set (data.frame) with `50` observations on `5` variables. For each of 50 countries there are 5 variables, which are the following: "sr" – aggregate personal savings rate, "pop15" – % of population under 15, "pop75" – % of population over 75, "dpi" – real per-capita disposable income, "ddpi" – growth rate % of dpi in the sixties. The data set was collected by AG. Sterling in 1977, and it was first published by DA. Belsley et al., 1980. The primal intention of the data collection was the examination of F. Modigliani's 'Life Cycle Hipotesis' from 1975: "The willingness for saving depends on the age structure of the country."

4.1.1. Three simple examples for the application of the regression model

At first we concentrate on the fundamental application methodology of the `lm()` function and on the simplest results of it. At the moment we do not investigate the validity of the model.

Example 1.: The case of a single regressor variable

Copy the following command in the **R** console!

```
lm(sr~pop15,data=LifeCycleSavings)
```

The systems' response after the edited version of our command is just the following:

```
(Intercept)        pop15
     17.497       -0.223
```

The concise interpretation of these results:

In the investigated countries
the savings rate (`sr`*) as the function*
of the population percent under `15` *(*`pop15`*)*
is roughly `17.5-pop15/5` *percent of the disposable income.*

Namely the observed savings rate *decreases* when the proportion of young people increases. In a fictional country without youth the savings are one sixth ($\approx$ `17.5`) of the income. When the proportion of youth is approximately equal to 1/3, then the savings rate is one tenth ($\approx$ `17.5 − 33.3/5`) of the income.

Look at the dependence of the savings rate on the proportion of old people, by means of the following command!

```
lm(sr∼pop75,LifeCycleSavings)
```

The new result:

```
(Intercept)        pop75
      7.152        1.099
```

The meaning of this is that the savings rate *increases* with the growing proportion of aged people in the population. In a society without elderly the savings rate is around 7.2%. And each +1 percent old, generate around +1.1 percent higher savings rate.

This result agrees with sociological ideas. But we have only two independent models, and also lack the statistical validation of the built models at the moment. The results until now means only, that the two foregoing models are separately the best linear functions of the two population indicators, approximating the disposed savings rate. The examination of the common impact of the two variables necessitates the construction of a common model, based on the two indicators.

Example 2.: Multiple linear regression

If we intend to explain the savings rate by the two population indicators (`pop15`, `pop75`) *together*, then the results from the two one-dimensional model effects inevitably give a misrepresentation. *We must put the two explanatory variable in the regression model at once* !

Consider the essential part of the response to the

```
lm(sr∼pop15+pop75,LifeCycleSavings)
```

command:

```
Coefficients:
(Intercept)        pop15        pop75
    30.6277      -0.4708      -1.9341
```

The result is surprising: the 1% increase in the proportion of youth *decreases* the savings rate by half percent, the 1% increase in the proportion of elderlies *decreases* the savings rate by two percent!

This means, that if we take both population indicators in our model at once, then the value of the target variable decreases by the increase of either explanatory variables!

Let us explain this more precisely. The savings rate in a country is approximately `30-pop15/2-pop75*2` percent by the resulting formulae of our regression model. Namely if we put the two variables together in the model, then the size of the effect of the proportion of young people is the double as in the case of single variable regression model! And the effect of the proportion of elderlies is not an increase, but a decrease!

Example 3.: Polynomial regression model

The preceding inconsistent results suggest an additional investigation of the influence of the proportion of population over 75. So we may get the idea of modelling the savings rate by a polynomial function of `pop75`. Notice, that the polynomials are linear functions *in its parameters (the coefficients)*. Hence the `lm()` command also supports the fit of polynomial models.

Let us use the

```
lm(sr∼pop75+I(pop75^2),LifeCycleSavings)
```

command! The important part of the results:

```
(Intercept)        pop75   I(pop75^2)
    5.9658       2.3997      -0.2608
```

We see that by this polynomial model the linear effect of the proportion of elderly is again positive and it has increased in absolute value, and the effect of the *square of the proportion of aged* has a clean negative effect! But this model building is nothing else then guessing. It has neither economical nor sociological foundations. Is is purely a result of a calculation.

In the case of our other data set, – the `trees` data – the reason for the existence of the polynomial regression model is more natural. The appearance of the dependence of the tree volume on the girth size in a form of cubic polynomial is logical.

Isn't that so?

Let us begin the analysis of the `trees` dataset!

Let our hypothesis be that the dependence between the girth size and the tree volume can be approximated by the incomplete polynomial

$$y \sim a + b \cdot x + c \cdot x^3.$$

It lacks the quadratic term.

This model is fitted by the following `lm()` command:

```
lm(Volume~Girth+I(Girth**3),trees)
```

The essential part of the results:

```
(Intercept)          Girth     I(Girth^3)
  -3.919278       1.322529       0.006092
```

This result shows, that the volume of the tree can indeed be represented by a polynomial having positive coefficient of the third power of the girth size.

But if we take a thinner tree, like one with a girth of 2.8 inches, then the volume of the tree is negative by the received formula ...

If we want that the model fulfills the zero intercept condition also, then it is attainable by the `(M<-lm(Volume~-1+Girth+I(Girth**3),trees))` command. The new coefficients are:

```
     Girth  I(Girth^3)
0.896578    0.006735
```

The numerical difference between the last two models is notable only in the interval $[0, 10]$.
But we can check by the command

```
summary(M),
```

that in this model all the coefficients and the whole model are all statistically significant.
It is an accomplished model!

4.1.2. The graphical interpretation of the simple regression model

In order to demonstrate the regression models, by a backstep, at first we describe the dependence of the *volume* and the *girth size* by the simplest polynomial:

$$y \sim a + b \cdot x.$$

In our case this formula means the following:

$$\text{volume} = \text{constant}_1 + \text{constant}_2 * \text{girth.size} .$$

The above scheme can be interpreted and fitted in **R** for the `trees` dataset by the commands:

```
y<-trees$Volume
x<-trees$Girth
(M<-lm(y~x))
```

This command sequence defines the two used variables for convenience, the `trees$Volume` and the `trees$Girth`, as `y` and `x` respectively, and saves the resulting object of the fit command `lm()` to the `M` variable.

For a better understanding the nature of the data and the fitted model let us plot the data set:

```
plot(x,y,pch=20,col="red")
```

By the `coef()` command we get the coefficients of the fitted model from the `M` object, and we add a blue line to our image, which corresponds to the regression function:

```
abline(coef(M),col="blue")
```

The red points represent the trees and the blue line represents the regression of the tree volumes on the tree girth sizes. This blue line is the line, which is the nearest to the points among all lines, if we use quadratic distance from the data points, measured vertically.

We show the predicted values in blue by the following command:

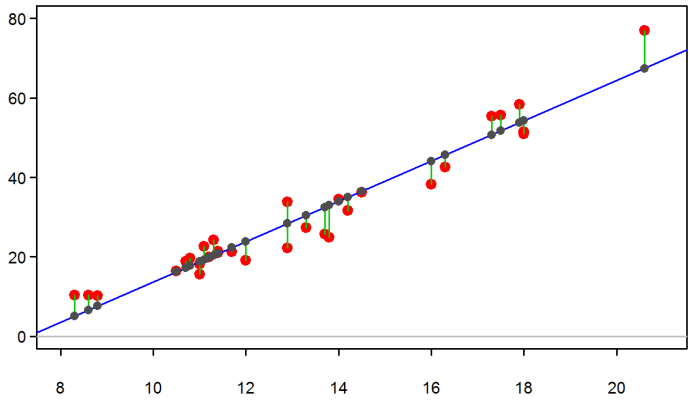
```
points(x,predict(M),pch=8,col="blue")
```

Here we used the `predict()` method. This program uses the `M` model variable only, and without a new data set, it calculates the value of the fitted model in the points of the original design matrix.

We can draw in green the distances, which were mi-

nimised in squared-sum form by the
`segments(x,y,x,fitted(M),col="green")`
command.

Here we used that the `fitted(M)` call gives just the same result as the `predict(M)` before.

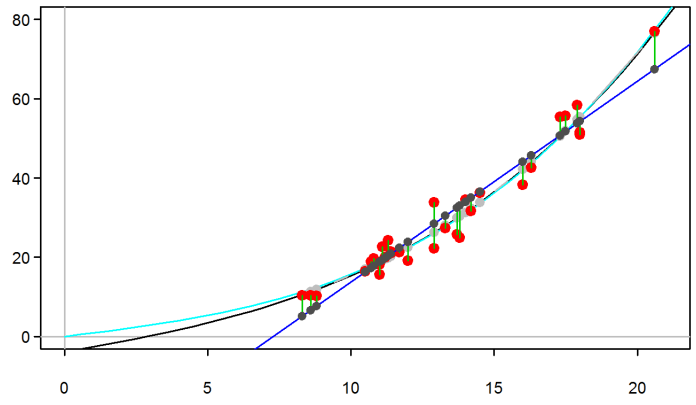


Let us add to our figure the graph of the two former approximating polynomials of degree three as well!

```
M3<-lm(y~I(x^2)+I(x^3))
a3<-coef(M3)[1]
b31<-coef(M3)[2];b33<-coef(M3)[3]
lines(x0,a3+b31*x0+b33*x0**3)

M4<-lm(y~-1+I(x^2)+I(x^3))
b41<-coef(M4)[1];b43<-coef(M4)[2]
lines(x0,b41*x0+b43*x0**3,col="cyan")
```

The first command section fits and draws in black the polynomial of degree 3 without a quadratic term. The second adds to the plot the result of the zero-intercept version of the same polynomial in cyan.

The seven commands from above supplement the earlier plot, and the result is the following:



Visibly the third degree polynomials provide a better fit.
But it is really so?
How can we justify it statistically?

4.2. Practical regression fitting

Till now we investigated different functions from the parametrized function set:

`lm(y∼1+x+I(x**2)+I(x**3))`.

The regression functions were chosen from the subsets of this four parametric, third degree polynomial set. The last model was the two parameter function `lm(y∼-1+x+I(x**3))`, before it we fitted the three parameter model `lm(y∼1+x+I(x**3))`, and the first was the two parameter `lm(y∼x)` model.
The full third order model, by four parameters surely fits our data better than all other models with fewer parameters. Namely the parameter set of the latter are subsets of the full model.

At the same time, as it can be seen on the preceding plot, the two and three parameter third order polynomial regression lines are not too different. The only question is: is it worthwhile to use one or two additional parameters? The improvement in the fit by one or two extra parameters is really remarkable? It seems easier to use a third order model! But apparently in the region of measurements the linear model follows the measurements with equal accuracy as the higher order models. Is it worthwhile to take a more complicated third order model?

Consequently the fundamental question is:

How can we resolve the problem of model validity?

4.2.1. The four basic questions

The general problem of model validity is decomposable into the following four elementary questions:

Question 1
The problem of error distributions. Are the errors indeed realisations of independent random variables having normal distribution with expected value 0, and identical variance?

Question 2
The possibility of finding a broader, essentially better model.

Question 3
The significance of parameters. It may be that several parameters only accidentally differ from zero. Investigate the possibility of a narrower model, which is essentially just as good.

Question 4
The problem of overfitting. We fit a model to a concrete data set. It is feasible that the built model describes the present data set, but how will it work for other data sets?

Outline of the answers

The first question.

This mission is insolvable.

Without repetition there are no answers to the question of the dependence of two phenomena. Hence there is no way to justify the independence of the errors, for example of the first and second observations. The solidity of our answer is also weakened by the doubtfulness of the condition of zero expected value for each of the error variables. And it is similarly difficult to prove that our observations have identical dispersion.

The only relatively easily solvable question to test is the supposed normality of the observation errors, assuming that they are values of independent, identically distributed variables, with zero expected value and equal dispersion. Some available tests for normality are useful in the case, when one or other conditions are not surely fulfilled. These tests are sort of a portmanteau tests, which measure more types of departures from the supposed properties at once .

When we must drop the normality by a test of this portmanteau character, then it may happen that not the normality lacks, but the independence, or the zero expected value, or both.

The normality and the other properties of the observations are important because without normality and the other properties, the statistics used to answer other questions about the regression model do not have the usual distribution, and in the case of non-normality the common interpretation of the classical statistics becomes inappropriate.

The second question.

This problem is interesting, since a model with more explanatory variables is necessarily more accurate, but the parameter estimates of a model with more parameters are necessarily less accurate.

If we use very many parameters in our model, then the produced model is typically an interpolation of the values of the target variable. The saturated model fits the data without error. This interpolation character appears beyond a certain number of parameters, and such a model for random observations depends too much on the random noise of the actual measurements.

Fortunately when we investigate two models, where one is wider and the other is narrower, it is easy to check, whether the broader is substantially better or not. The obvious instrument is the standard deviation of the errors in the two models. By this idea we measure the explanation power of a model by the empirical value of the model error variance, more accurately: by the value of the residual sum of squares (`RSS`).

We accept the broader model if in that case the sum of squares of errors (the `RSS`) is significantly less then the sum of squares (`RSS`) in the narrower model.

In case of hierarchical models this technique limits the growth of the parameter set.

The *least* or *minimal model* consists of only one parameter: the mean value of the target variable. By the minimal model we assume, that the values of the target variable equal to a constant which does not depend on chance, plus a random error with zero expected value.

Comparing all models to the least model, – including the regression models – defines a hierarchical scheme. At first we always evaluate the relation of these two hierarchical models: the 'regression model' and the 'least model'. We investigate, whether the 'regression model' is significantly better than the 'least model' fitted to the same dataset or not.

The 'regression' and the 'least' model together forms the shortest model chain. Subsequently we elongate the chain by inserting more models between the maximal and minimal models. We leave as maximal model the regression model, and insert one or more models in the chain by subsequently deleting variables from the full model. Next we evaluate the adjacent pairs of the model chain, by the technique of pairwise comparisons. Finally we stop at the first, which is significantly worse then the preceding more abundant.

This stepwise constructed model is optimal only with respect to the chosen hierarchical model chain.

The third question.

The answer to this question is the simplest, but substantial.

It may happen that in the selected and fitted model there are parameters which values do not differ statistically from zero. The needless parameters damage the accuracy of other significant parameters. Hence it is important to erase these surplus parameters.

But this is one of the most pleasing work!

When the conditions are fulfilled, then the common

t-statistic is adequate. Hence we are able to delete the needless parameters one by one.

The fourth question.

The meaning of this question is for example the following. If anybody brings some *new* black cherry data, and we apply the model fitted to the original dataset, then if our model is a fair model, which isn't overfitted, then the fit of the model (the residual sum of squares) on the new dataset is approximately the same as to the original dataset.

Seldom exists an exact solution to a problem of this kind, because we rarely receive enough new, independent data from the same setting.

Hence we must simulate the existence of the new data. We must use tricks. There are many techniques and knacks. But if we have too few 'proper' data, then the reliability of these proxy methods is very low.

It is a plausible way to divide the data in two parts. And at first we fit the model to one part of the data. And afterwards we may validate the fitted model by the other part.

Another possible method is first fitting the model to the full data set. The fitted model decomposes every single observation in two parts: the predicted value and the estimated error. And afterwards we simulate new data by a random matching of the observed predictions and the errors.

In the control step we fit the previous model to this artificial data. And finally we compare the errors of the original and the simulated model.

Finally three further remarks related to model validity.

In the situation of model fitting two different problems may occur. In the first type of cases *we know* the subclass of the valid regression function, only the actual parameter values are unknown. In the second one, we choose the subclass *by the available data* and estimate its parameters as well.

In relation of the model validity we have to remark that the positive answers to the four previous questions *do not prove anything in a mathematical sense, they are simply results of an investigation.* A mistake is possible. The result of an investigation can state at best, that '*by the available data, by the tested models... our outcome is ...*'.

The size of our data set is always a relevant factor. And it is also important that the fitted model should be interpretable for the given data set.

In the case of our example data set `trees`, if we fit a model, which implies that the growth of the volume is quicker than a linear function of the girth, then this means that a greater tree has a relatively thicker girth. Consider, are we able to give one reasonable explanation to this behaviour, or not?

4.2.2. Answers to the basic questions with the help of the `lm()` procedure

The simplest way to answer the four question is
- first fit a model by the `lm()` procedure
- then evaluate the results, by the `summary()` function.

Let us use the black cherry dataset once more!

Use the model

$$V = a + b * G + e$$

where e is the value of the random error, which

are assumed to be independent and distributed normally, with zero mean and unknown, constant standard deviation.

The suitable command compressed in one line:

```
summary(lm(trees$Volume~trees$Girth))
```

The result of the function `summary.lm()` here consists of four sections:

1.) confirmatory print out of the calling sequence
2.) statistics of the estimated errors of the model
3.) the coefficients of the fitted model and their statistics
4.) the statistics evaluating the fitted model.

The actual results:

```
Call:
lm(formula = trees$Volume ~ trees$Girth)

Residuals:
    Min      1Q  Median      3Q     Max
-8.0654 -3.1067  0.1520  3.4948  9.5868

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
trees$Girth   5.0659     0.2474   20.48  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.252 on 29 degrees of freedom
Multiple R-squared: 0.9353,     Adjusted R-squared: 0.9331
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

Let us interpret the printed results backwards!

In the (last) fourth section
you find five statistics.

On the basis of these statistics it is possible to decide about the validity of the model.

* The '`residual standard error`' is the estimated standard deviation of the e errors. It is calculated from the residuals of the fitted model.

* The '`degrees of freedom`' means the degrees of freedom of the estimated residual standard error. In our case it is $29 = 31 - 2$, because there were 31 cases and we estimated 2 parameters.

* The '`multiple R-squared`' is the square of the empirical correlation of the target variable and of the approximating variable determined by the regression method. Its common notation is `R2`. By an other interpretation it is the explained proportion of the variance of the target variable. The '`adjusted R-squared`' is a modified value of `R2`. If we investigate two hierarchical models, then the adjusted version of `R2` isn't necessarily monotonically increasing, while `R2` is.

* The 'F`-statistic`' is essentially the quotient of two empirical variances. More precisely, it is a standardised ratio of two sums of squares: the sum of squares of the fitted values, and the sum of squares of the residuals. This statistics indicate whether the regression model is intrinsically better as the (narrower) constant-mean (minimal) model, or not. This statistic has an $F_{1,29}$ distribution when the minimal model is the valid model of the data.

* The last statistic is the 'p`-value`' of the F`-statistic`. It is a transformed version of the previous F`-statistic`. When the minimal model (the least) is the proper model of our data, then it is $\mathcal{U}([0,1])$ distributed. It is small when the regression model is the proper model and large in the case of the least model. At present it is small: the fitted regression model is significantly better than the minimal model.

The third section
is under the heading '`Coefficients`'

Here you find the estimations of model parameters (see the '`Estimate`' column) and the statistics to evaluate the results. The simplest approach is to consider only the p-values in the '`Pr(>|t|)`' column. The stars at the end of the rows mark the decision result of the hypotheses about the coefficient. The stars in our table signs that the calculated coefficients significantly differ from zero.

The second section
under the title '`Residuals`'.

Five statistics describe the distribution of the estimated residuals: minimum, maximum, median, the first and the third quartile. Broadly speaking it is the same as the result of the `fivenum()` command: the Tukey Five-Number summary.

The first section of print
gives the edited version of our call.

4.2.3. Answer in short

Verbal evaluation of the printouts of the `summary(lm())` procedure used for the black cherry data.

The used command:

```
summary(lm(trees$Volume~trees$Girth))
```

"*The calculated regression model is a valid model.*
In the last line the small p-value of the F-statistic shows that the error of the regression model is significantly smaller than the error of the minimal model. The estimations of the `Intercept` *and of the coefficient* `trees$Girth` *can be used, since the p-values of the t-statistic are small. It means that the two calculated parameters are significantly different from zero. The skewness, manifesting the difference of quartiles and the median value* `0.15`*, is negligible at a standard error* `4.3` *and at the range* `17.65`*, calculated from the minimum and maximum. So the asymmetry of the residuals is unimportant, hopefully the deviation of the distribution of the errors from the normal distribution is inessential.*"

"*Thus, by the* `trees` *data, in the case of black cherry trees, if we investigate the linear dependence of the volume of the wood material in the tree, on the girth size, we get the following approximating equation:*
`wood volume = -36.9 + 5*girth size + error`,
where the mean value of error is `0` *and its estimated standard deviation is* `4.3`."

Comment:
In the next point we deduce that both from a practical viewpoint and from physical considerations, — and statistically, too — the result of the
$\texttt{lm}(\texttt{trees\$Volume} \sim -1 + \texttt{I}(\texttt{trees\$Girth} * *2) + \texttt{I}(\texttt{trees\$Girth} * *3))$
command provides a better model.

4.2.4. The graphical evaluation of the model errors

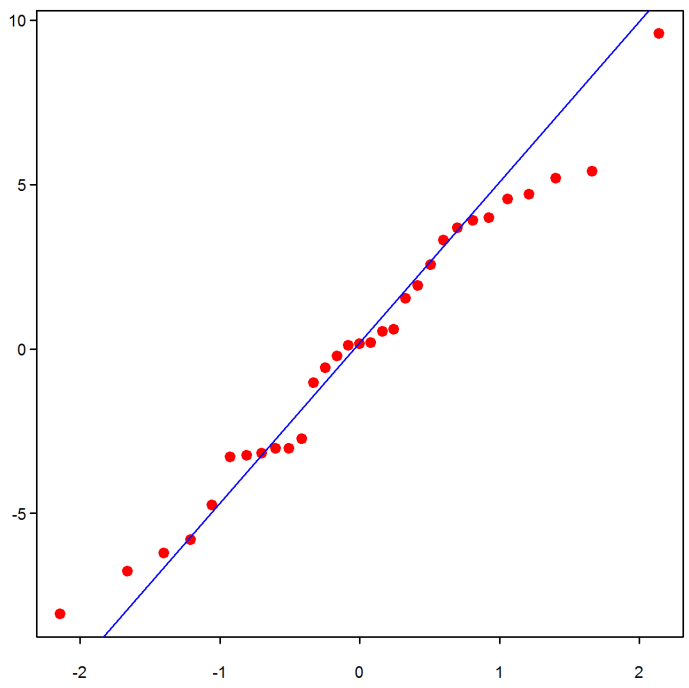
We introduce some simple graphical methods for analysing the residuals (i.e. the lengths of the line segments, shown by green dashes in the previous plot).

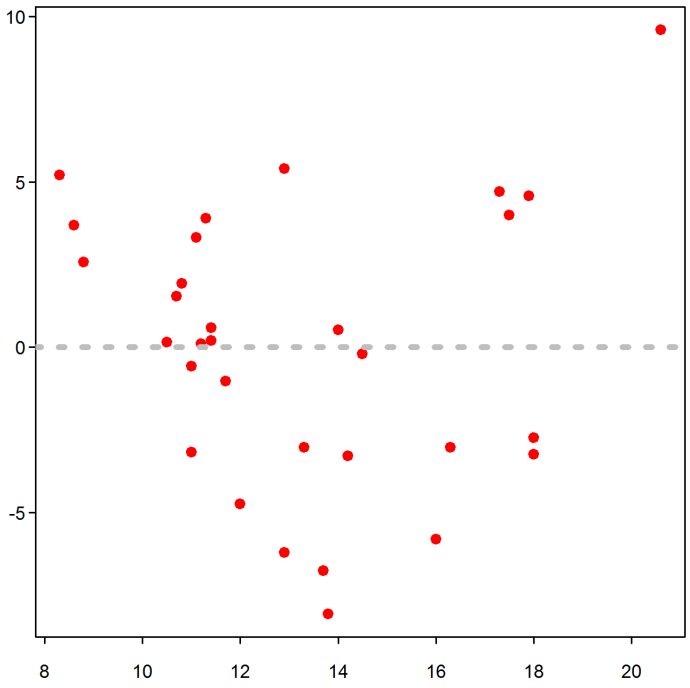
Let us consider the previously analysed cherry-tree model and save the fitted model to the variable `M` by the help of the
`M<-lm(Volume~Girth,trees)`
command!

First we check the normality of the residuals of the model by the command `qqnorm()`.

The previously in green plotted residuals can be got from the fitted model by the `residuals()` function.

So the following combination of commands:
```
qqnorm(residuals(M))
qqline(residuals(M))
```
gives the QQ plot of the residuals from the regression.





The line on the plot is the result of the second command. It is determined by the first and third quartile of the residuals. As it can easily be checked, this is the line through the 8th points from the left and the right, as we have 31 points on the plot.

When evaluating the figure, we must check if the points on the QQ plot deviate significantly from this line.

The x coordinates of the points are the quantiles of the standard normal distribution, while the y coordinates are the empirical quantiles of the residuals.

Based on the plot, we may get the conclusion that the hypothesis of the normality can be accepted, as the points essentially lie on a straight line.

In the univariate case like ours, it is a natural idea to create a two-dimensional plot, where the x coordinates of the points are the values of the explanatory variable, while the y coordinates are equal to the error of the regression in the corresponding points.

In our case we may get such a plot by the command lines

```
x <- trees$Girth; y <- residuals(M)
plot(x,y)
abline(h=0)
```

It can be seen that 'there is a problem' with the used model. For the intermediate values of the explanatory variable the errors are mostly negative, while for its small and large values these are rather positive. So in our case this figure-type looks useful for diagnostic purposes. It shows something that cannot be seen on the statistics of the `summary()` and which is hard to spot on the previous plot either.

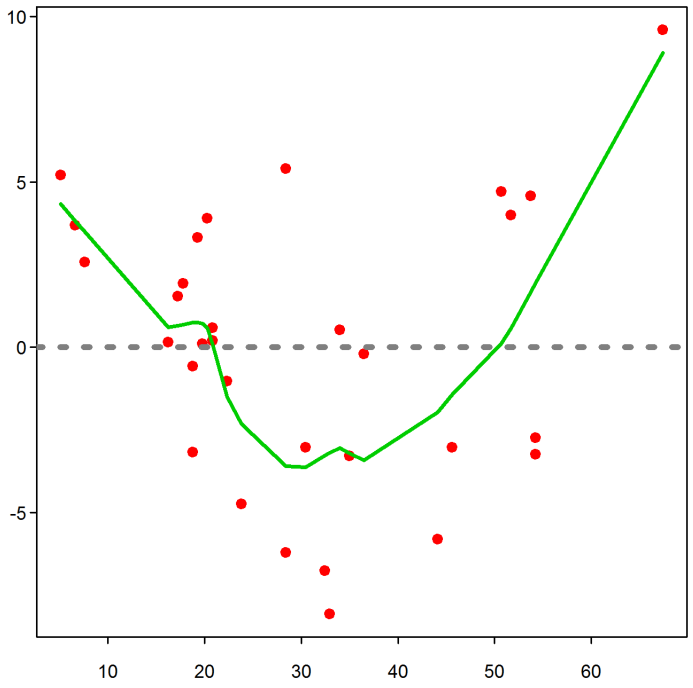
But it is not usual to use this plot.

Since in the multivariate case, when we have not one, but several explanatory variables, we would have as many figures as explanatory variables. And one cannot compare this figures, as it can easily be the case that the seemingly unexplained effect on a diagram is just the effect that is explained by the other variables.

But if we plot the points, for which the y coordinates are still the residuals, but the x coordinates are the forecasted values of the target variable – instead of the values of the explanatory variables –, we get a very useful diagram.

So let us consider the plot created by the commands

```
plot(fitted(M),residuals(M))
abline(h=0)
```



As we can see, in the univariate case (when we have only one explanatory variable), only the scale of the x axis has changed, as the forecasts are linear functions of the previously used values.

But if we use this diagram in the multivariate case, then the y coordinates (the heights) remain the same, only the x coordinates (including their order) change. This permutation usually does not correspond to the order of any of the explanatory variables. The shown diagram contains a green curve as well, which is created by the `lowess()` smoothing of the points. This part of the plot can be got by the following command:

```
panel.smooth(fitted(M),residuals(M))
```

The curve we got this way is sometimes called local regression. Unfortunately the detailed description of this interesting method is beyond the scope of this note, so we just give a short explanation.

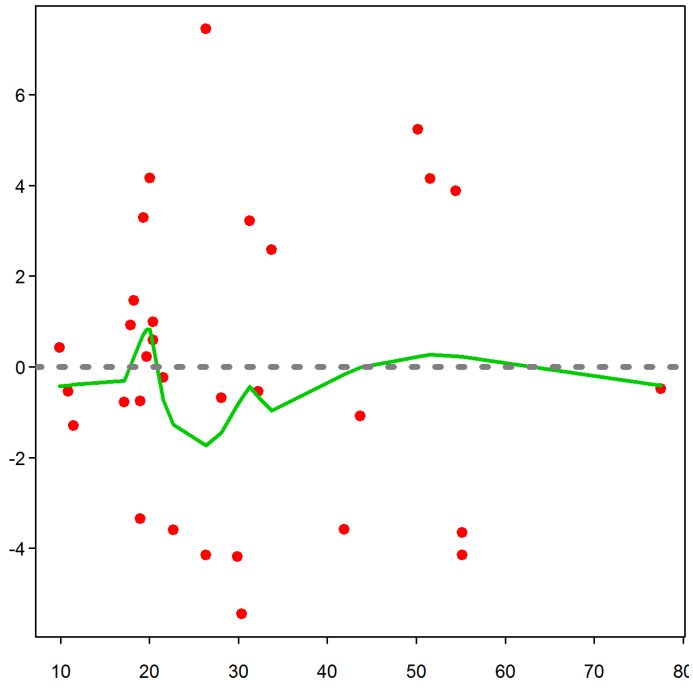
What we see as a line on the diagram, is actually a sequence of points, calculated at exactly the same x points, where the residuals are plotted. The points of this point sequence is got by fitting an approximation polynomial to the red points near to the actual point (base) – and then take the value of the fitted polynomial in the given base point.

When fitting the polynomial, the distance of the red points from the polynomial is weighted by a decreasing function of the distance of the red point and the actual base point (measured on the x axis).

This graphical interpretation of the residuals suggest, that it may be worth trying to approximate the wood material of the tree by a higher polynomial of its girth.

So let us calculate the regression of the wood material by a third order polynomial of its girth and consider the same diagnostic plot of the resulting errors.

```
y<-trees$Volume;x<-trees$Girth;
(M3<-lm(y~x+I(x**2)+I(x**3)))
plot(fitted(M3),residuals(M3))
abline(h=0)
panel.smooth(fitted(M3),residuals(M3))
```



The diagram shows that the model probably correctly estimates the volume of the wood material. The error (namely that the estimations are low for the average values and high for those differing from the average) seen on the previous plot has diminished.

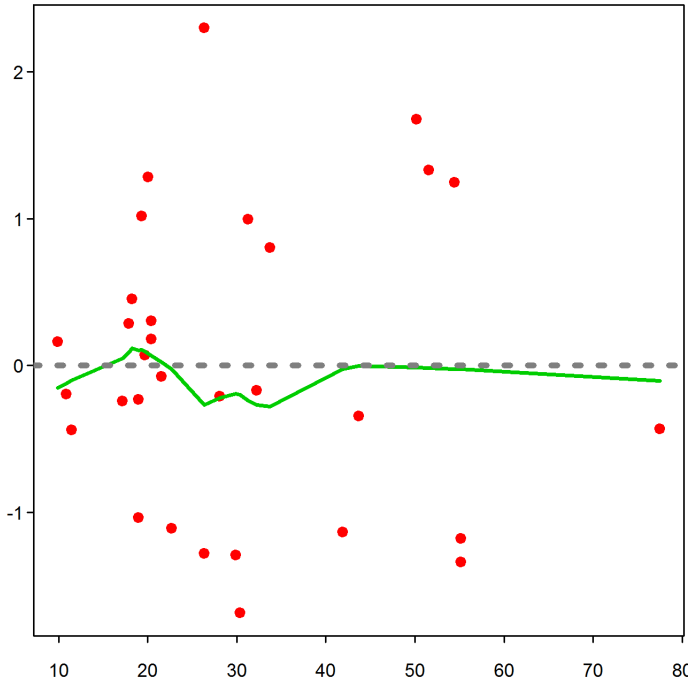
However, when investigating if the errors of the model have the same variance for all observations, then we see that the error variance is likely to be substantially larger in the middle of the data range than at the ends of it.

A final remark

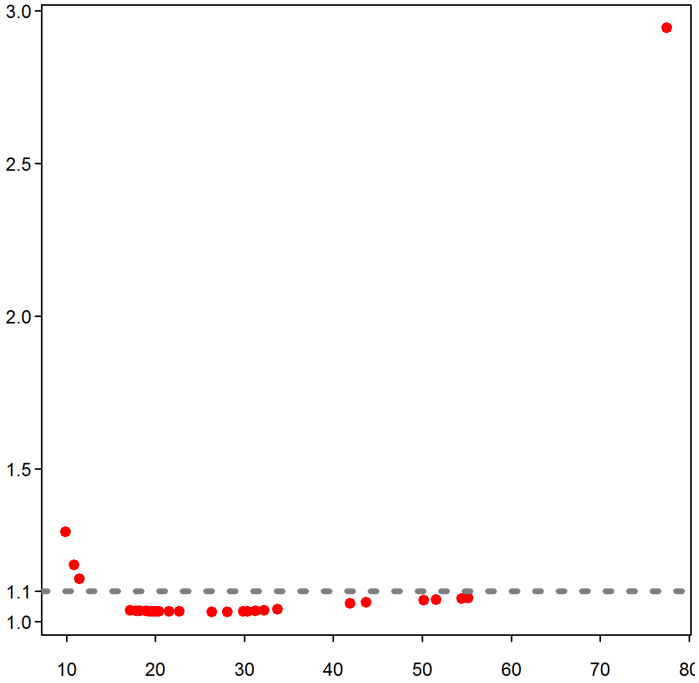
The errors shown on the previous plots are estimated values *with different variances*. Thus in case of more sophisticated investigations not the raw errors, but their standardized versions

$$r_{\texttt{STA}} = \frac{e}{s\sqrt{1-h}}$$

are to be plotted on the y axis of this diagnostic diagram (as we shall see when investigating the diagnostic statistics; h is the so-called leverage effect, compensating for the fact that residuals far away from $\bar{x}$ have a smaller variance, due to the applied least-square methodology).



This modification resulted in our case in just a very small difference from the previous plot, as it is seen from the next diagram.



The reason of this phenomenon is that the $\texttt{1}/\sqrt{\texttt{1}-\texttt{h}}$ statistic used at the modification (where `h` is the value of 'hat') differs from the rest only for the first 3 and the last 1 observation.

4.3. The statistics, evaluating the model fit

The fitted basic assumption of the $Y = X\beta + \varepsilon$ linear model is that the $\varepsilon_1, ..., \varepsilon_n$ errors, corresponding to the individual observations are:
- normally distributed;
- having zero expected values;
- having identical standard deviation.

In order to simplify the command lines let us extract the important variables from the source `data.frame` and save the fitted model and its summary statistics to the `M` and `ST` variables:

```
y<-trees$Volume;x<-trees$Girth;
(n<-length(y)); (p<-2)
(M<-lm(y~x)); (ST<-summary(M))
(m<-mean(y));(yh<-fitted(M))
```

4.3.1. The '`Residual standard error`'

We may think– based on our previous experiences – that the already introduced '`Residual standard error`', which is the fourth element of the `summary(M)` is the same as the result of `sd(residuals(M))`.

But in our case this gives instead of the expected `4.252` a value of `4.181` – as it is seen from `summary()` –, and the difference is larger than any possible rounding error.

The `sd()` function computes the empirical variance from the corrected empirical variance. This means that before computing the square root, the sum of squares of the difference between the observations and their average is divided by $n - 1$, where n is the number of observations. The reason for it is the fact that this way the estimation is computed from the unbiased estimator of the variance, assuming that these deviations are independent and the single linear dependence among them is that their sum is 0.

But in our case we must give an unbiased estimator for the error variance by two n dimensional points, where one of them is arbitrary (the y), but the other – while being n dimensional as well– lies in a p dimensional subspace (this is the $\widehat{y}$, the regression – i.e. projection – to the p dimensional subspace spanned by x)

Thus the line segment that is between our two points (y and $\widehat{y}$) and upon which length we intend to estimate the variance, is in reality only an $n-p$ dimensional line segment. So when calculating the length of it, we must take this dimensionality into account, thus it must be divided by $n - p$. In fact the same phenomenon is the reason why we should divide the squared distance by $n - 1$.

In our case we consider the residuals of a fitted two-parameter model, so the correct divisor for ensuring the unbiasedness of our estimator is $31 - 2 = 29$. Considering this fact, if we use for estimating the regression error standard deviation the formula

```
sd(residuals(M))*sqrt(30/29)
```

then the result coincides with the previously seen value of `ST$sigma=4.252`.

It is worth noting that while the square of the result is an unbiased estimator of the variance, the number itself is not an unbiased estimator for the standard deviation.

4.3.2. The '`Multiple R-squared`'

`R2` is indeed the squared correlation between the measurements and the forecasts as the value of

```
cor(y,fitted(M))**2
```

is `ST$r.squared=0.9353`, which is the same as the value in the table of `summary(M )`.

The value of `R2` can be got by the formula

```
var(fitted(M))/var(y)
```

as well. This is a consequence of the fact that we formed the ratio of the forecast vector and the original target vector and in these estimators the sums were divided by $n-1$ and these denominators cancel each other out. The result without the denominators is

```
s2pred<-sum((fitted(M)-m)^2)
s2orig<-sum((y-m)^2)
s2pred/s2orig
```

which explains why is R2 called *explained total-deviation proportion*, too.

In the last formula we have used that the mean of the fitted values is the same as the mean of the target variable i.e.
`round(mean(fitted(M))-mean(y),12)==0`).

It is worth noting that we also have
`round(mean(residuals(M)),12)==0`
(allowing for numerical inaccuracies). As the vector of the residuals is orthogonal to the $\underline{1} = (1, ..., 1)$ vector. Thus we have that the average of the residuals is 0. The same is true for the weighted average:
`sum(residuals(M)*x)==0`,
as the residuals are also orthogonal to the $\underline{x}$ vector.

If we have more explanatory variables, then the residuals are also orthogonal for their n dimensional vectors, so there are as many different zero-valued linear combinations of the residuals, as many explanatory variables.

This implies that the degree of freedom for the residuals of a regression with p parameters is exactly $n-p$. This can also be proven by the earlier argument: the residual after the projection is in the orthogonal subspace to the p dimensional subspace generated by the $(\underline{1}, \underline{x}, ...)$ vectors.

Caution!

The same formula, written by the `var()` functions can easily be misunderstood. It seemingly explains the alternative, but not exact name of the R2, being the explained variance-proportion. This name is in error, as we have already shown that from the two variances the one in the numerator is actually not an unbiased estimator of the variance of the forecasted values by the regression.

4.3.3. The 'Adjusted R-squared'

The formula of the adjusted R2 is

$$1-(1-\mathtt{R2})\frac{\mathtt{n}-1}{\mathtt{n}-\mathtt{p}},$$

which means in our case:
`1-(1-ST$r.sq)*(n-1)/(n-p)`
by the following logic.

We have already seen that R2 is the *explained deviation-proportion*. This means that $1-\mathtt{R2}$ can be understood as the *non-explained deviation-proportion*. As we had n observations and by this we have estimated p parameters, this non-explained deviation-proportion can be considered as based on $\mathtt{n}-\mathtt{p}$ 'free' observations. Thus, $(1-\mathtt{R2})/(\mathtt{n}-\mathtt{p})$ is the non-explained deviation-proportion for a single observation. But in fact we have $\mathtt{n}-1$ free observations, so the total non-explained deviation-proportion is $(\mathtt{n}-1)(1-\mathtt{R2})/(\mathtt{n}-\mathtt{p})$. This implies the previous formula.

The adjusted R2 is by definition smaller than R2. A negative value shows that the model is overfitted, i.e. that we have used too many parameters in the regression.

The adjusted R2 (abbreviated: adjR2) has the advantage over R2 that it is not monotonic.

If we increase the number of parameters in the model, then R2 is increasing, but the adjusted R2 not always, as in the denominator of the formula there is the p, which is the number of parameters. If p becomes as large as approximately n, then the multiplier of -(1-R2) in the formula of adjR2 can be as large as n-1, so the value of adjR2 can decrease to 2-n. On the other hand, if we approximate y by a constant, then p=1, and the value of adjR2 is equal to R2, thus non-negative. It proves that there is a p value (number of parameters), for which adjR2 is maximal.

So if we must decide which model to choose from a hierarchical increasing set of models, then it is a sound method to choose the largest model, for which the adjusted R2 is larger than for the next narrower model.

4.3.4. The '*F-statistic*'

Briefly, we may define the `F` statistic as the ratio of the variances of the minimal model and the regression model. But it is just approximatively valid. Namely, when the minimal model is true, then we intend to get a quantity with `F` distribution. To this we need the quotient of two independent variance estimators. But the current estimators are not independent.

However, when we take not the variance estimators of the models, but the total squared errors, then the total squared error, belonging to the regression model and the increase between the regression model and the minimal model are already independent. Thus, when we normalise the total squared error and the difference of the two total errors with their respective degrees of freedom, then if the minimal model is true, then their ratio

```
(s2pred/(p-1))/((s2orig-s2pred)/(n-p))
```

will have F distribution, as it can be seen from the result of the printout `summary(M)` command.

But the formula can be simplified, as in the `n` dimensional space the point `y` corresponding to the observations, the `yh` $:= \hat{y}$ point, corresponding to the regression estimator of the target variable and the (`m` $:= \bar{y}, ..., \bar{y}$) point of the space diagonal forms a right angled triangle.

Thus the Pythagorian theorem implies that the

```
sum((y-m)**2)-sum((m-yh)**2)
```

equals to

```
sum((yh-y)**2).
```

Thus the difference between the squared total sum of the `m` minimal model and the `yh` regression model is `sum((yh-y)**2)`, which is in fact the squared total error of the regression model, and its degree of freedom is `p-1`.

Thus in our case the increment has $p-1 = 1$ degree of freedom, the degree of freedom of the squared error of the regression model is $n-2 = 29$, and the given, by the `summary(M)` printed `ST$fstat` statistic has a value of $= 419.4$, which can also be calculated as the result of the command

```
sum((yh-m)^2/(p-1))/sum((y-yh)^2/(n-p))
```

4.3.5. Interpreting the '*p-value*'

The last element in the printout of `summary(M)` is the *p*-`value`, which is based on the fact that the distribution of the F statistic is $F_{1,29}$ if the minimal model is true. Its value can be got by the `1-pf(ST$fstat[1],1,29)` command. In our case it gives 0 – as the used functions have restricted punctuality.

If the minimal model is true, then the *p*-`value` has $\mathcal{U}([0,1])$ distribution. However, if not the minimal model is true, then the `Fstat` value is large, so the *p*-`value` will be small.

It can be seen that we get an optimal method e.g. on the 5% level, if we choose as critical set if the *p*-`value` is in $[0, 0.05]$. I.e. we reject the validity of the minimal model if the *p*-`value` is smaller than 5%.

However, it is important to interpret the rejection of the null hypothesis correctly. We cannot say that the probability of the null hypothesis is small, just that the data show too big deviations from the ones expected under the minimal model, towards the alternative hypothesis.

Thus the probability that in case of the validity of the minimal model, taking also the alternative hypothesis into account, the observed deviation (expressed as the `Fstat` value) would be greater or equal the current one, is too small.

4.3.6. The '`Coefficients`' table

In the 4-column '`Coefficients`' table the estimated parameters of the fitted models and the statistics for their checking are to be found.
The four columns are the following:

1. the estimators for the coefficients of the regression;
2. the estimated standard errors of the estimated coefficients;
3. the t-values of the estimated coefficients;
4. the p-values of the t-statistic.

To the interpretation of the 'Coefficients' column, let us briefly recall the general results for the regression, based on the least squares method!

Let us write the values of the target variable to the Y vector with n rows. and the values of the explanatory variables, completed with a column containing a vector of 1s to the X matrix of size $n \times p$. Then the regression model is $Y = X\beta + e$, where e is a vector of length n, containing the actual values of an independent, $\mathcal{N}(0, \sigma)$ distributed ε error vector.

Using the notations above, the estimator of β by the least squares method, based on Y and X is $\hat{\beta} = (X^T X)^{-1} X^T Y$. The covariance of $\hat{\beta}$ is $\sigma^2 (X^T X)^{-1}$ and $\hat{\sigma}^2 = ||\hat{Y} - Y||/(n-p)$ is a variance estimator, independent from $\hat{\beta}$. From this follows, that if one of the components of β is zero, then its estimator, normalised with its own standard deviation, has the t_{n-p} distribution.

Based on the reasoning above, one can test by the p-values in the fourth column of the table the hypothesis that the estimated parameter has a significant deviation from 0. The second column contains the estimated standard error of the estimators in the first column, while in the third column the ratio of the two numbers from the first and the second column can be found.

These briefly summarised statements about the printouts can be checked by a short sequence of commands:

```
X <- cbind(1,x);
Y <- as.matrix(y)
n <- nrow(X);
p <- ncol(X)
cn <- c("Summary","Calculated")

est <- cbind(ST$coe[,"Estimate"],
(b <- solve(t(X)%*%X)%*%t(X)%*%Y) )
colnames(est)<-cn

est # coef estimators -----------------

Yh <- X%*%b
V<- solve(t(X)%*%X)*sum((Y-Yh)^2)/(n-p)
# vcov(M) # would be the same
ste <- cbind(ST$coe[,"Std. Error"],
(se <- sqrt(diag(V))))
colnames(ste)<-cn

ste # standard errors -----------------

tva <- cbind(ST$coe[,"t value"],
(tval <- b/se))
colnames(tva)<-cn

tva #  t-values ----------------------

pva <- cbind(ST$coe[,"Pr(>|t|)"],
2*pt(abs(tval),n-p,lower=FALSE))
colnames(pva) <- cn

pva # $p$-values of the estimators ------
```

The statistics of the fourth part can be checked just the same way as we have checked the statistics of the third part. E.g. by using the following commands:

```
# --
# a "Residual standard error"

ST$sigma
sqrt(sum((Y-Yh)^2)/(n-p))

# --
# a "Multiple R-squared"

ST$r.squared

(R2 <- cor(Y,Yh)^2)
var(Yh)/var(y)
m <- mean(y)
sum((Yh-m)^2)/sum((y-m)^2)

# --
# az "Adjusted R-squared"

ST$adj.r.squared
1-(1-R2)*(n-1)/(n-p)
```

```
# --
# Fstat  2 degrees of freedom

ST$fstatistic
fst <- sum((Yh-m)**2/(p-1)) /
                 sum((Y-Yh)**2/(n-p))
c(fst,p-1,n-p)

# --
# there is no p-value for Fstat
# neither in M nor in ST
# only in the actual printout
# the p-value of the F stat
# 1-pf(fst,p-1,n-p)
```

This latter p-value is not contained in the result of the `summary()` function, which is an object of class `"summary.lm"`. In our case its value is 0, due to the rounding and the difference from the printed p-value is only due to the specialities of the `print.summary.lm()` function.

4.4. Some classical examples for the regression

To demonstrate the impossibility of automatic model evaluation we present four classical examples.

In point 4.1) we present the example of F.J. Anscombe. The dataset constructed by Anscombe consists of four artificial subsets, for which the 'classical' diagnostic statistics are identical, but the fitted linear model holds essentially in the first case only. In point 4.2) we show Huber's data, which demonstrates that the more sophisticated tests can also be erroneous. Even the tests activated by the `influence.measures()` command detect substantial influence for this artificial data, where there is no such effect. In point 4.3) we discuss the test data of J.W. Longley. This dataset is a classical break test instrument for regression fitting algorithms. From a technical point of view the most delicate step in the regression procedure is the inversion of the X^TX matrix, where the X matrix is the matrix of the given values of the explanatory variables. Here the classical problem of 'multicollinearity' appears, because there is a strong correlation between the explanatory variables. In the point 4.4. we present a more complex model built for the dataset by J.O. Irwin. This is a specialized technique for the case when we must transform the target variable. But this example is very instructive also from the viewpoint of model validity. Here the diagnostics produces strange results, too.

4.4.1. The Anscombe data

F.J. Anscombe published this artificial data set in 1973. This includes four (x, y) observations of length 11, for which the regression of y on x is the same in all cases. At the same time the standard deviation of the y variables, the error of the regressions and the estimated covariance is the same. However...

The dataset can be found within **R**-in the `anscombe` object of the `datasets` library. This is a `data.frame` of size 11×8, the name of the variables are `x1, x2, x3, x4` and `y1, y2, y3, y4`, respectively.

A linear model can be fit to the first pair e.g.by the command

```
(M1<-lm(y1∼x1,anscombe))
```

As a result of this command and the corresponding ones to the further three data pairs:

```
(M2<-lm(y2∼x2,anscombe))
(M3<-lm(y3∼x3,anscombe))
(M4<-lm(y4∼x4,anscombe))
```

we can see that the intercept of the regression is in all four cases, `3`, while its steepness is `1/2`.

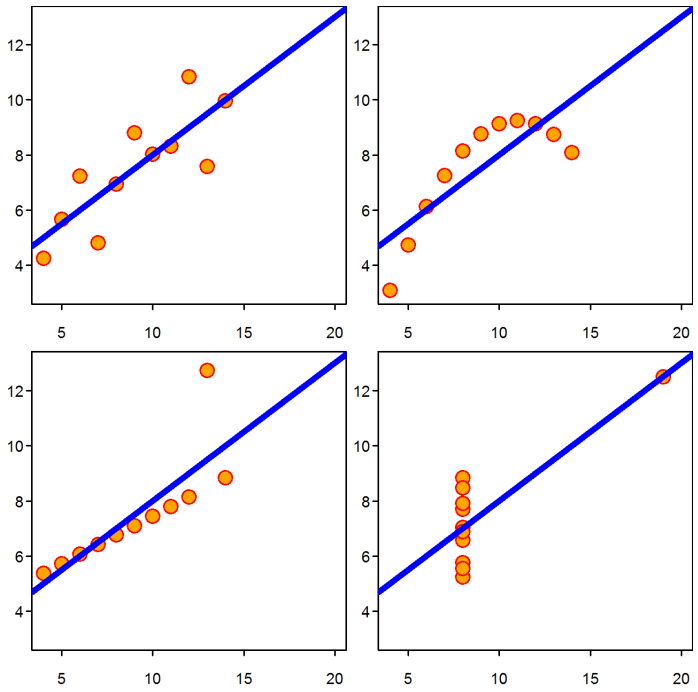
If we apply the `summary()` function to the previously calculated `M` models, like this

```
summary(M1),
```

then the results show that the standard error of

the parameter estimates is always `1.125` and `0.118`, respectively. and the residual error is `1.237` in all 4 cases, and the value of `R2` is `0.6662`. This implies that all of the other statistics are equal, too. Exceptions are the 5 Huber-type numbers of the residuals:, including the error median.

On the following plots one can see the points of the artificial test data sets and the lines fitted by regression methods:



This plot can be constructed by the following sequence of commands:

```
par(mfrow=c(2,2),mar=c(1,1,1,1))
xy<-c(0,4)
plot(anscombe[xy+1]);abline(coef(M1))
plot(anscombe[xy+2]);abline(coef(M2))
plot(anscombe[xy+3]);abline(coef(M3))
plot(anscombe[xy+4]);abline(coef(M4))
```

It can be seen that from the data sets, that the first is the only one, which can be assumed to have been realised, as it is claimed in the statistical conditions of the linear regression model.

In the second case it would be better to approximate the y values with a polynomial of degree 2 of the x values. The third data set includes an outlier, while in the fourth case the x variable looks as being discrete (its two actually observed values are 8 and 19).

4.4.2. The Huber data

This is a dataset of minimal size, where the diagnostic statistics detect outliers, including an effective measurement.

The size of the dataset is 6.

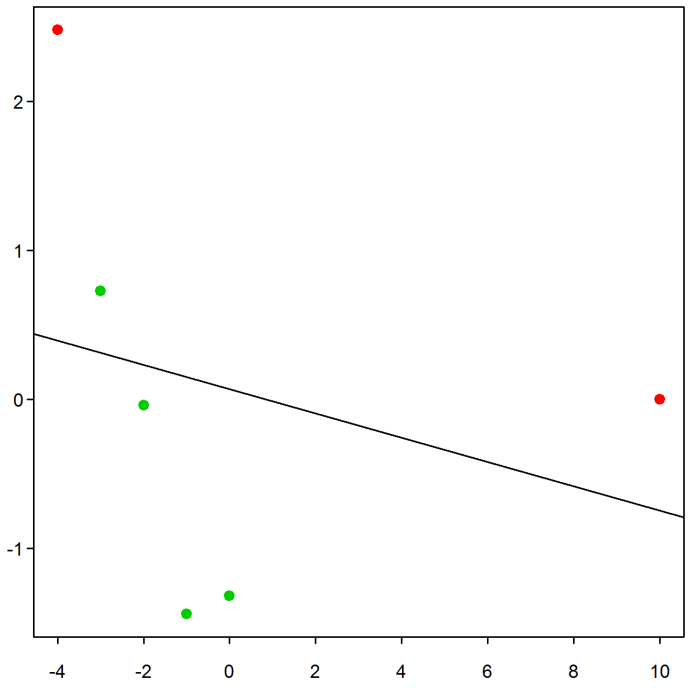
At first we write the data in the (`xh`, `yh`) variable pair.

Then we fit a linear regression by the method `lm()` and evaluate the model, by the `summary()` command.

Finally we check the result of the method `influence.measures()`, and we mark by red colour those cases, which were qualified as 'critical'.

```
xh<-c(-4   ,-3   ,-2   , -1   ,  0   , 10)
yh<-c( 2.48, 0.73,-0.04, -1.44, -1.32,  0)
summary(lmH <- lm(yh ~ xh))
(im <- influence.measures(lmH))
plot(xh,yh,main=
  "Huber's dataset")
abline(lmH);
points(xh[im$is.inf],yh[im$is.inf],
       pch=20,col="red")
```

When we perform the above commands, then the result is the plot below:



On the plot you see the "normal" data points in green, and the two 'outliers' in red. Obviosly the right extreme is the only outlier.

4.4.3. The Longley data

J.W. Longley published in 1967 the data, which is available in the `datasets` library under the name `longley`. Its size is 16×7. The data are the macroeconomical descriptors of the US from the years 1947-62, as follows: (1.) GNP.deflator - implicit price deflation based on 1954; (2.) GNP- gross national product; (3.) Unemployed - number of unemployed, in 100 000; (4.) Armed.Forces - number of people serving at the army, in 100 000; (5.) Population - population of at least 14 year old, in million; (6.) Year - year of the investigation; (7.) Employed - number of employed people, in million.
Longley used this data set for checking the power of the computers, programs and algorithms of the 1960s. It is still a famous test data set, as its determinant is nearly 0, since the data is strongly correlated. This can be seen from the results of the next commands.

```
X<-as.matrix(longley)[,-7]
det(t(X)%*%X)
round(log10(eigen(t(X)%*%X)$va),1)
round(cor(X),3)
```

The first command saves the data 'longley' into the matrix X except the last one, the number of employees, which is the target of our modelling. The second shows that the determinant $det(X^TX)$, which is crucial for the numerical stability of the regression, has a size of 10^{24}. The result of the nest command shows that the logarithms of the eigenvalues of the matrix X^TX are `7.8, 5.3, 5.0, 4.3, 1.4, 0.3` – these are important for the properties of the iterative methods. The last gives that from the 15 correlations between the pairs of the explanatory variables 5 are larger than 0.99!

In spite of these bad signs the above variables have quite good explanatory power for the `Employed` variable. This can be seen from the results of the

```
M<-lm(Employed ~ ., data = longley)
summary(M)
```

commands.
It is worth checking, how the result changes if we omit variables from the model:

```
SM <- step(M);summary(SM)
```

These changes in case of omitting the variables one by one can be checked e.g. by calling the function `drop1(M)`. Otherwise also this `drop1()` function is used by the function `step()` to decide, which variable can be omitted, and which is worth omitting without substantially decreasing the explanatory power of the regression while improving the information criterion.

4.4.4. The Irwin data, generalised regression

W. Smith investigated in 1932, what proportion of mice is saved by an anti-pneumococcus serum from becoming ill in a culture of 17 hours. J.O. Irwin used the part, called S32 of this experiment for interpreting a special regression method in 1937.

In this experiment 40-40 mice were treated with a 1-5-times dose of the tested serum, and all $5 \cdot 40 = 200$ mice were exposed to the same level of infection. It was assumed that the probability of protection in the groups is a $p_x \in [0,1]$ value, which depends on the dose got. They wanted to determine the dependence of the number of protected mice on the logarithm of the dose.

The S32 data:

dose	1	2	3	4	5
investigated	40	40	40	40	40
protected	0	2	14	19	30

Let us suppose that the survivals for the dose x is y_x and that the probability p_x of the reaction in the mice is independent from each other, but depending on the dose x. This implies that under dose x from 40 mice, the survival probability of exactly y animals is:

$$P_x(y) = \binom{40}{y} p_x^y (1-p_x)^{40-y}.$$

It would give an erroneous result if we would apply the regression for the empirical values $(x, \widehat{P}_x = y/40)$. I.e. if we would suppose that the proportion $\widehat{P}_x$ of the protected depends linearly from x (the logarithm of the dose).

It is erroneous, because if we would model the empirical distribution function linearly, then most likely we would get a model, which may give a probability in $[0,1]$ for the investigated doses, but for other reasonable x values the result may fall outside of $[0,1]$, which is obviously an erroneous estimate.

Thus it is practical to transform the probability to be modelled (the observed relative frequency) in a way before its modelling that the transformed value can be an arbitrary number in $\mathbb{R}^+$.

One of the usual transformations is the so-called 'logit' transform.

To this logit transform we first take the odds-ratio $p/(1-p)$ of the investigated probability p and its complement. This is a monotonic transformation of the probability p, so we get a value in $\mathbb{R}^+$.

This odds-ratio is small for small values of p, but it is spectacularly large for larger values of p (that is why it is used in gambling).

But the odds-ratio cannot be negative, so its value is still bounded. That is why it is common to take the logarithm of this ratio. This way we get the logit transform of the probability:

$$\ell = log\left(\frac{p}{1-p}\right).$$

I.e. the logit of to the probability $p \in (0,1)$ is a one-to-one, monotonically increasing transformation of the probability p to $\mathbb{R}$.

There are other methods to be used for the same purpose. The inverse of any univariate distribution is suitable. E.g. Φ^{-1}, the inverse of the distribution function of the normal distribution. This is also a monotonic transformation from $[0,1] \to \mathbb{R}$, and the value $\Phi^{-1}(p)$ is usually called the probit of the probability.

We shall use the logit transform, but any other function $[0,1] \to \mathbb{R}$ would do.

Transformations used for this purpose are usually called 'link' functions. The inverse of the link function is called 'transmission' function, which is in case of the log–transformation

$$p = \frac{e^\ell}{1+e^\ell}.$$

Thus we choose a linear regression, for which the logit of the probability of survival is a linear function of the logarithm of the dose:

$$log\left(\frac{p_x}{1-p_x}\right) \approx a + bx = \ell(x)$$

in a way that by the probability

$$\widehat{p}_x = \frac{e^{\ell(x)}}{1+e^{\ell(x)}}$$

the result of the experiment, i.e.

$$P_{a,b}(x,y) = \prod \widehat{P}_{x,y} = \binom{40}{y} \widehat{p}_x{}^y (1-\widehat{p}_x)^{40-y}$$

should be the largest possible value.

It means that now we do not apply the least squares method, but we maximize the $P_{a,b}(x,y)$ probability of the sample, which depends on the a and b parameters. I.e. we accept those parameter value (a,b) as estimator, for which the sample is the most probable.

This model fitting method is called *maximum likelihood*.

Let us compute in **R** the fitting of the sketched model!

In our case, as we have a very simple data set, which is given by 7 numbers, it is the easiest to type the data immediately to our program. In the next program-part, the available number of observations per dose is `mi`, the number of survivals is `yi` and the respective dose is in the variable `xi`.

```
xi <- c( 1, 2, 3, 4, 5)
yi <- c( 0, 2,14,19,30)
mi <- c(40,40,40,40,40)
summary(lmI<-glm(cbind(yi,mi-yi)~xi,
                 family = binomial))
```

This data set and model has the curiosity that the applied diagnostic tools qualify 3 variables out of the 5 as crucial, as it is shown by the result of the next command:

```
(imI <- influence.measures(lmI))
```

```
  dfb.1 dfb.xi dffit cov.r cook.d  hat
1 -0.81   0.76 -0.81  0.80   0.25 0.26
2 -0.48   0.42 -0.50  2.43   0.26 0.37
3  1.30  -0.92  1.74  0.25   1.29 0.35 *
4  0.00  -0.07 -0.25  3.36   0.08 0.39 *
5  0.29  -0.40 -0.51  5.21   0.36 0.63 *
```

The process marks those lines with '`*`', which are considered as crucial from the point of the regression.

V. Analysis of variance (ANOVA)

5.1. The solution of an elementary ANOVA problem in **R** with a 5-lines program

As an introduction, we present the ANOVA model fitting and interpretation, in a very simple case, omitting the exact explanations. It is namely easier to understand the general method, if we show first the solution of a special elementary problem in **R**.

Let us choose the `MASS::npk` dataset. This `data.frame` has 24 observations and 5 variables. The `$yields` column gives the pea-yield pro unit area in 24 cases. The other columns describe the circumstances of the cultivation. `$block` shows the soil type from the 6 possibilities. The `$N`, `$P`, `$K` variables show if there was additional nitrogen, phosphorus or potassium treatment.

We first need some preparatory commands. The first loads the `MASS` package. The name of this package comes from the book, titled "Venables & Ripley: Modern Applied Statistics with S" to which it was a supplement. The second command shows the structure of the `npk` dataset. The third command replaces the names of the original `yields` and `block` variables to the shorter `Y` and `B`. The fourth defines a function, which will be used several times. It gives back the squared length of a vector. Finally, the fifth prints the first 6 lines of the working data set.

```
library(MASS)
str(npk)
names(npk)[c(1,5)]<-c("B","Y")
SS<-function(u) as.numeric(crossprod(u))
head(npk)
```

The main task of this part is the interpretation of the results of the next 5 commands. The first command fits, prints and saves an ANOVA model to the `M` object. The next lines provide 'only' informations to the understanding of the results. The second line prints the effect and degree of freedom of the members of the model. The third shows the parameters of the fitted model. The fourth prints statistics about the parameters of the estimated model. The last, fifth gives the estimated value of the yield for the 6 different soil types.

```
(M<-aov(Y~B,npk))
summary(M)
coef(M)
summary.lm(M)
predict(M,data.frame(B=factor(1:6)))
```

First we show the results of the five command lines. Then we summarize their evaluation. Finally we interpret the results of the commands separately.

The results of the five commands

The result of the `(M<-aov(Y~.,npk))` command:

```
  Call:
     aov(formula = Y ~ B, data = npk)

  Terms:
                         B Residuals
  Sum of Squares  343.295   533.070
  Deg. of Freedom       5        18

  Residual standard error: 5.441967
  Estimated effects may be unbalanced
```

The result of the `summary(M)` command:

```
            Df Sum Sq Mean Sq F value  Pr(>F)
B            5 343.29   68.66  2.3184 0.08607 .
Residuals   18 533.07   29.61
```

The result of the `coef(M)` command:

```
  (Intercept)       B2       B3       B4       B5       B6
       54.025    3.425    6.750   -3.900   -3.500    2.325
```

The result of the `summary.lm(M)` command:

```
Call:
aov(formula = Y ~ B, data = npk)

Residuals:
    Min      1Q  Median      3Q     Max
-7.2250 -3.4937 -0.5375  2.1062 11.8750

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   54.025      2.721  19.855 1.09e-13 ***
B2             3.425      3.848   0.890   0.3852
B3             6.750      3.848   1.754   0.0964 .
B4            -3.900      3.848  -1.013   0.3243
B5            -3.500      3.848  -0.910   0.3751
B6             2.325      3.848   0.604   0.5532

Residual standard error: 5.442 on 18 degrees of freedom
Multiple R-squared: 0.3917,     Adjusted R-squared: 0.2228
F-statistic: 2.318 on 5 and 18 DF,  p-value: 0.08607
```

The result of the `predict()` command:

```
      1      2      3      4      5      6
 54.025 57.450 60.775 50.125 50.525 56.350
```

The general evaluation of the model

Now we have used only the yield `Y` and the soil type `B` variables from the `aov` dataset and we looked an answer for the following question:

> "*What is the effect of the soil type to the yield?* "

In order to solve this problem, using the $\texttt{Y} \sim \texttt{B}$ parametrisation, we chose the following model:

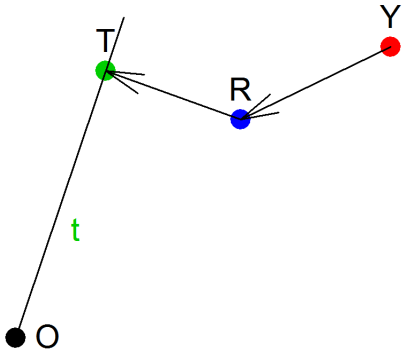
$$y_k = \mu + \beta_{B_k} + e_k \qquad k = 1, \ldots, 24.$$

Here y_k, $k = 1, \ldots, 24$ denotes the value of the target variable, i.e. the pea yield in the different cases. μ denotes a baseline value, valid for all growing circumstances. The B_k indices of the β parameters, $k = 1, \ldots, 24$ gives the soil type of the kth observed cultivation as given in variable `B`. As the possible values of `B` are $1, \ldots, 6$, the possible values of β_{B_k} are $\beta_1, \ldots, \beta_6$ according to the soil type of the given experiment containing in B_k. Finally, the e_k, $k = 1, \ldots, 24$ values are the errors of the model, based on the first two parameters. Namely the errors of the model, which defines the yield as the sum of the μ baseline value and the modifying value β_j, $j = 1, \ldots, 6$, depending on the soil type. We assume that these e_k errors are observed values of the ε_k, $k = 1, \ldots, 24$ variables with 0 mean and σ standard deviation. It is supposed that the variables ε_k, $k = 1, \ldots, 24$ are independent from each other and follow the $\mathcal{N}(0, \sigma)$ distribution (i.e. they have normal distribution with expectation 0 and standard deviation σ).

By this model e.g. for the first measurement we assume that the $y_1 = 49.5$ yield we got is equal to $\mu + \beta_1$ plus an additional e_1 error, as $\texttt{npk\$B}[1] == 1$. For the fifth observation, the $y_5 = 59.8$ yield is equal to $\mu + \beta_2 + e_5$, as $\texttt{npk\$B}[5] == 2$, etc.

In the model μ and $\beta_1, ..., \beta_6$ are the unknown parameters. The `aov()` method gave that parameter set as an estimator for these values, for which the $\sum_{k=1}^{24} e_k^2$ error sum of squares is minimal.

To a visual interpretation let us choose three 24 dimensional points!



Let `Y` $\in \mathbb{R}^{24}$ be the point, which has coordinates $y_1, ..., y_{24}$. Let `R` the point also in $\mathbb{R}^{24}$, which has coordinates being equal to the model-based estimated values of the target variable, i.e. $\widehat{y}_1, ..., \widehat{y}_{24}$. This is just the same vector what can be got by the `predict(M)` command. Let `T` the point in $\mathbb{R}^{24}$, which has $\overline{Y}$ in all of the coordinates, i.e. which has the average of the target variable in each of its coordinates.

Let us call the `Y` point as the measurement point, the `R` point as model or regression point, the `T` point as minimal model or complete average point.
These three points can be calculated as:

```
Y<-npk$Y
R<-predict(M)
T<-rep(mean(npk$Y),24)
```

The $YRT\Delta$ triangle we have got by the method above is a right angled triangle, so by the Pythagorean theorem we have

$$\overline{YR}^2 + \overline{RT}^2 = \overline{YT}^2.$$

The right angle property can also be proven by the following command:

```
c(SS(T-R),SS(R-Y),SS(T-Y))
```

as the result is a sequence of three such numbers, for which the third is the sum of the first two:

```
343.29 533.07 876.36
```

If we investigate the result of the first `aov()` and the second `summary()` of the above five commands, then we find the first two numbers in both of them.

The results of `aov`:

```
                        B Residuals
Sum of Squares    343.295   533.070
Deg. of Freedom         5        18

Residual standard error: 5.441967
```

and the results of `summary`:

```
            Df Sum Sq Mean Sq F value  Pr(>F)
B            5 343.29   68.66  2.3184 0.08607 .
Residuals   18 533.07   29.61
```

I.e. the sum of squares $\overline{\mathtt{TR}}^2$ is printed under the label `B`, and the sum of squares $\overline{\mathtt{RY}}^2$ is printed under the label `Residuals`.

The interpretation of this fact is the following:

The three points are all in $\mathbb{R}^{24}$, but `R` is in the subspace, which is spanned by the $b_1, ..., b_6$ below and `T` in the one, which is spanned by t.

$$t = \begin{pmatrix} 1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1 \end{pmatrix},\ b_1 = \begin{pmatrix} 1\\1\\1\\1\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix},\ b_2 = \begin{pmatrix} \\ \\ \\ \\ 1\\1\\1\\1\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix},\ b_3 = \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ 1\\1\\1\\1\\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix},\ b_4 = \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ 1\\1\\1\\1\\ \\ \\ \\ \\ \\ \\ \end{pmatrix},\ b_5 = \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ 1\\1\\1\\1\\ \\ \\ \end{pmatrix},\ b_6 = \begin{pmatrix} \\ 1\\1\\1\\1 \end{pmatrix}.$$

I.e. the coordinates of R are the following:

```
     1      2      3      4      5      6
54.025 54.025 54.025 54.025 57.450 57.450
     7      8      9     10     11     12
57.450 57.450 60.775 60.775 60.775 60.775
    13     14     15     16     17     18
50.125 50.125 50.125 50.125 50.525 50.525
    19     20     21     22     23     24
50.525 50.525 56.350 56.350 56.350 56.350
```

while T is the constant 54.875 all the time.

Let us observe that the above 7 vectors span only a 6 dimensional subspace, as t equals with the sum of $\mathtt{b}_1, \ldots, \mathtt{b}_6$. I.e. t is in the subspace spanned by $\mathtt{b}_1, \ldots, \mathtt{b}_6$.

As we have searched for the parameter set, which minimizes the $\sum e_k^2$ and the parametrization looked for the solution among the linear combinations of the b vectors, the R point, corresponding to $\widehat{Y}$ is the projection of Y to the subspace, which is spanned by $\mathtt{b}_1, \ldots, \mathtt{b}_6$.

The projection of the Y point to the line t is the T point as it is an elementary fact that for any real numbers $y_1, \ldots, y_n$ the $a = \overline{y}$ is the constant, which minimizes $\sum_k (y_k - a)^2$.

Taking these two orthogonality into account and the fact that t is part of the subspace spanned by $\mathtt{b}_1, \ldots, \mathtt{b}_6$ it is also true that the projection of R for t is also T.

Thus the Residuals in the printout are the distance of the measurement Y and its projection R to a 6 dimensional subspace. Similarly B is the distance of R and its projection to the diagonal T.

Thus the results can be interpreted as follows. We have two models for the measured data: the

$$y_k = \mu + d_k \qquad k = 1, \ldots, 24$$

minimal model, where the $d_1, \ldots, d_{24}$ errors are observed values of such $\delta_1, \ldots, \delta_{24}$ random variables, which are independent and have identical $\mathcal{N}(0, \sigma_d)$ distribution. And the third, the ANOVA model, which assumes

$$y_k = \mu + \beta_{B_k} + e_k \qquad k = 1, \ldots, 24.$$

By the previous arguments we have T as the approximation of the Y measurements by the minimal model, similarly if the R is also considered as measurements, then T is the approximation of the R by the minimal model.

Thus the Residuals x Sum of Squares=533.070 measures the distance between the R approximation and the Y measurements, while the B x Sum of Squares=343.295 gives the distance between the R ANOVA model and the T minimal model.

The second line, having the label Deg. of Freedom of the aov table and the first column of the summary table, labelled Df gives the degree of freedom of the previous row, resp. the next column. Here Residuals x Df=18 and B x Df=5. These degrees of freedom show the dimensionality of the line segments, which length we are speaking about.

As while Y, R, T each are a 24 dimensional point, the line segments $\overline{\mathtt{YR}}$ and $\overline{\mathtt{RT}}$ respectively, lie within a subspace, which can be exactly determined. Namely, $\overline{\mathtt{YR}}$ lies in the 18 dimensional orthogonal subspace of the subspace spanned by the vectors $\mathtt{b}_1, \ldots, \mathtt{b}_6$ and $\overline{\mathtt{RT}}$ lies in the 5 dimensional subspace of the subspace spanned by the vectors $\mathtt{b}_1, \ldots, \mathtt{b}_6$, which is orthogonal to the spanned subspace by t.

Next we interpret the printout of the 5 commands applying the geometric interpretation and the two models (the minimal and the ANOVA).

First command line: aov()

The printout of the aov command contains one more element besides the already seen sum of squares and degree of freedom, the estimation of the ε errors of the ANOVA model. This gives:

```
Residual standard error:  5.44.
```

This result can also be got by the command

```
sqrt(SS(Y-R)/18)
```

ensuring that this value can indeed be considered as an estimator of the ANOVA error ε, denoted by σ.

This observation is based on the assumption of the ANOVA model that the $y_1, \ldots, y_{24}$ observations have independent coordinates, expected value $\mu + \beta_{B_k}$ and the same σ standard deviation.

As R is the estimator of $\mu + \beta_{B_k}$, the squared length of the line segment $\overline{\texttt{RY}}$ can be considered as an estimator of the error sum of squares. This sum of squares have to be divided by 18 instead of 24, as while $\overline{\texttt{RY}}$ is in $\mathbb{R}^{24}$, it is orthogonal to the vectors $b_1, \ldots, b_6$. So we have demonstrated, why is the `Residual standard error` an estimator of the ANOVA variance.

Second command: `summary()`

In the printout of the `summary` command there are four further numbers beyond the ones already shown:

```
            Df Sum Sq Mean Sq F value  Pr(>F)
B            5 343.29   68.66  2.3184 0.08607 .
Residuals   18 533.07   29.61
```

We can observe that the values in the `Mean Sq` column are the ratio of the numbers in the previous two columns. The number in the `F` column is the ratio of the two numbers in the `Mean Sq` column.

The last, `Pr(>F)` column contains a single p-`value`. This can directly be got by the commands

```
F<-(SS(R-T)/5)/(SS(Y-R)/18)
1-pf(F,5,18)
```

Let us observe that the `F value` is the ratio of the length of the line segments $\overline{\texttt{RT}}$ and $\overline{\texttt{YR}}$, apart from normalisation. Thus the `F value` is large, if the `R` ANOVA model is far away from the `T` minimal model, comparing the distance between the `Y` measurement and the `R` ANOVA model. And if `F` is large, the p-`value` is small, showing that the ANOVA model is an important improvement, compared to the minimal model. On the other side, a large p-`value` means that the ANOVA model is not much better than the minimal one.

The normalisation in the formula can be understood if we consider the following: both distances are divided by the dimension of the corresponding line segments, thus the F statistics gives the ratio of the averages per dimension. Basically, as a general setup we may understand this reasoning as a decision about a walk from the `T` point towards the measurement `Y`. Now there is just one ANOVA model we investigate, so we have one intermediate point, the `R`. It is worth undertaking this step `TR`, if it is long enough, compared to `RY`, taking into account the increase in the number of parameters as well.

It is worth taking these normalised squared distances, as if $\beta_1 = \cdots = \beta_6$, then the $\overline{\texttt{TR}}$ and $\overline{\texttt{RY}}$ (random) line segments are independent normally distributed vectors, thus the computed variable has F distribution. Thus the p-`value` in the last column has uniform$[0,1]$ distribution in this case, which means that the $[0, \alpha]$ interval can be applied as an optimal critical set at level α, as we have already seen in Section 4.

The p-`value` is linked to the hypothesis test, where H_0 is that the he observed values of the target variable are from the minimal model, and the H_1 is that the observed values of the target variable come from the ANOVA model. In other words, under H_0 the β values are equal, while under H_1 there is at least one that differs from the rest.

Thus in our case, when the p-`value` is `0.086`, then at the level of 10% we must reject the equality of the β values, while at the level of 5% we must accept it.

Third command: `coef()`

The third command, `coef` prints out the parameters of the fitted model. As we see, in our case this list contains the `(Intercept) B2 B3 B4 B5 B6` values. So `B1` is missing!

The reason for it is that the previously defined ANOVA model is overparametrised. Thus a model with parameters $\mu, \beta_1, \ldots, \beta_6$ fits the measured data just as well as for an arbitrary d the $\mu - d, \beta_1 + d, \ldots, \beta_6 + d$. This problem has been solved in our case so that the value `B1` was chosen as 0, which is not contained in the printout, where we get the values of the `(Intercept)` and the `B2 B3 B4 B5 B6` parameters.

Fourth command: `summary.lm()`

The result of the `summary.lm` command is the same as the one we get in case of a linear regression, for continuous explanatory variables.

As it can be seen the printouts of this command consists of four parts. These are the following: 1.) The essential part of the command, which resulted in the

creation of M; 2.) Some statistics for the residuals; 3.) The estimation of the coefficients and their statistical evaluation; 4.) Some statistics about the aov model in general.

The essential part in our case is the third part about the coefficients:

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   54.025      2.721  19.855 1.09e-13 ***
B2             3.425      3.848   0.890   0.3852
B3             6.750      3.848   1.754   0.0964 .
B4            -3.900      3.848  -1.013   0.3243
B5            -3.500      3.848  -0.910   0.3751
B6             2.325      3.848   0.604   0.5532
```

The first column of the table gives the estimators for the model parameters, which we have already seen. The second is the standard error of these estimators. The third, the `t value` is the ratio of the first two columns. The fourth is the p-`value` of the `t value`. I.e. the probability that a number with the t_{18} distribution is larger in absolute value than the `t value` we have got.

This latter statement can easily be checked by the commands

```
w<-summary.lm(M)$coef
round(2*pt(-abs(w[,"t value"]),18),5)
round(w[,"Pr(>|t|)"],5)
```

The statistics of `summary.lm(M)` are suitable to decide about the separate null hypotheses whether the parameters are equal to zero.

In our case the answer is clear about the `(Intercept)`, which is the estimator of the μ parameter. If μ were equal to zero, then the probability would be very small that we get this or an even larger `t value`. Thus we reject the hypothesis of $\mu = 0$. The situation about `B3`, i.e. the estimator of β_3 is not as clear. If we calculate with a 10% first type error probability, then it is justified to reject the $\beta_3 = 0$ hypothesis. But at the level of 5% we must accept that `B3=0`, similar to the other parameters, which have such a small difference from 0 (compared to their standard deviation) that a larger deviation would have a probability of `30-50%` even for a parameter having 0 value, in case of the given sample size and number of parameters.
The printout of the fourth part of the fourth command:

```
Residual standard error: 5.442 on 18 degrees of freedom
Multiple R-squared: 0.3917,     Adjusted R-squared: 0.2228
F-statistic: 2.318 on 5 and 18 DF,  p-value: 0.08607
```

Here we see essentially the same statistics, which we have got earlier by the `aov` and the `summary` commands.

The `R2` and `adjR2` of the second line are new. These results can be got by the commands

```
(r2<-cor(Y,predict(M))^2)
SS(R-T)/SS(Y-T)
1-23*(1-r2)/18
```

as well, and these command lines give at the same time the interpretation of the given statistics.

The first line shows that `R2` is the correlation between the target value and its approximations. The second line gives an interpretation why is it common to call the value of `R2` as 'interpreted proportion of deviation'. Its value is now `0.3917`, and here we have calculated the ratio of the squared differences of `R` from the model average and the differences of `Y` from the model average. The third line shows, how can the adjusted `R2` be calculated from `R2`. For a more detailed explanation of these coefficients see the chapter on regression.

Fifth command line: `predict()`

The last row of the five command lines show, how can the model got by the `aov` command be applied.

The first argument of the `predict()` command is the `M` model, the second is a `data.frame` with all the possible values of the variable `B` (in our case (1,2,...,6)), which are assumed by the `M` model as explanatory variables. The result of this command is the estimated value of `Y` for all possible values of `B`.

We may get the same result by the command lines

```
b<-coef(M)
c(b[1],b[1]+b[-1])
```

as well.

It may be a question, why must the parameters be interpreted this way? The practical result is that the program calculated the estimations by assuming β_1 being equal to 0. We come back with a more complete answer later.

The problem is the fact that – as we have seen – `t` is an element of the spanned subspace by the $b_1, \ldots, b_6$ vectors. The definition of the points `R` and `T` is una-

nimous. The only question is, how should this be expressed by the `t` and `b` vectors. One possible way is setting β_1 equal to 0, as it is done by the program. But it would be possible to define any other β as 0. Or we could have define them as their sum being equal to 0. If another β would be a structural zero, then it could be interpreted as the baseline being the result of this β. If the sum of the β values would be 0, then it would mean that the average effect of the B factor is zero.

But how can we know that the system uses the $\beta_1 == 0$ assumption?
This is shown by the result of the

```
M$contrasts
```

command, as seen by the

```
$B
[1] "contr.treatment"
```

lines.
Namely, the calculation method of the so-called contrasts (the β values) was `"treatment"`. If it were `"sum"`, then `B6` would be missing, and its value would be not zero, but `-(B1+..+B5)`.

5.2. `ANOVA` as a model and a method

The `ANOVA` *model* is a linear model, where the target variable is continuous and where *all the explanatory variables are discrete*, i.e. such that the number of its possible values are finite. The `ANOVA` model is fitted using the methodologies of the regression. And we check if the fitted more complicated model is better than the simpler ones, by using the methods of the `ANOVA`. Thus the notion of `ANOVA` has two meanings: first it is a *model*, second it is a *method*, similarly to the regression.

As methods, both regression and `ANOVA` should be applied in all linear models. Both are a technique. Regression methods determine the optimal parameters of a model. `ANOVA` helps to decide if the fitted model is good enough.

In spite of the possible confusion caused by the twofold meaning of the notions, it is natural to use the name of the two methods for the models, as in case of continuous explanatory variables the fitting, while for discrete explanatory variables the model checking is the most important question.

The linear models may have many more parameters in case of discrete explanatory variables. As one is tempted to link a separate parameter to each of the possible values, or even possibly to some combinations of values. Thus in case of the discrete variables the possible decrease of the number of parameters is in the focus.

`ANOVA` has got its name from the method, that allows us to decide if a model with fewer parameters is essentially as good as a one having more parameters. We investigate, whether the errors of the two models have the same standard deviations?

5.2.1. The history of the name

The name '`ANOVA`' has its origins back in the ages, when there were no computers available, so the models were fitted only in cases when one had the same number of observations for every combination of the explanatory variables. In this case the statistics, together with the estimators of the parameters can easily be calculated by computing averages, differences and squares.

5.3. The basic cases of ANOVA

Example 1: ANOVA for one factor

Let us repeat the model of the introduction, but now focusing more on the data analytic part.

The `npk` is a frequently cited dataset of the `MASS` package, which is also to be found in the `datasets` package, available immediately after starting the system.

The contents of the data `npk` is the following:

No	B	N	P	K	Y	No	B	N	P	K	Y
1	1	0	1	1	49.5	13	4	1	0	0	62.0
2	1	1	1	0	62.8	14	4	1	1	1	48.8
3	1	0	0	0	46.8	15	4	0	0	1	45.5
4	1	1	0	1	57.0	16	4	0	1	0	44.2
5	2	1	0	0	59.8	17	5	1	1	0	52.0
6	2	1	1	1	58.5	18	5	0	0	0	51.5
7	2	0	0	1	55.5	19	5	1	0	1	49.8
8	2	0	1	0	56.0	20	5	0	1	1	48.8
9	3	0	1	0	62.8	21	6	1	0	1	57.2
10	3	1	1	1	55.8	22	6	1	1	0	59.0
11	3	1	0	0	69.5	23	6	0	1	1	53.2
12	3	0	0	1	55.0	24	6	0	0	0	56.0

The 24 lines show the pea yield (`Y`≡ yield) in 6 blocks (`B`≡ block), under two different (0//1) level of nitrogen (`N`), phosphorus (`P`) or potassium (`K`) treatment.

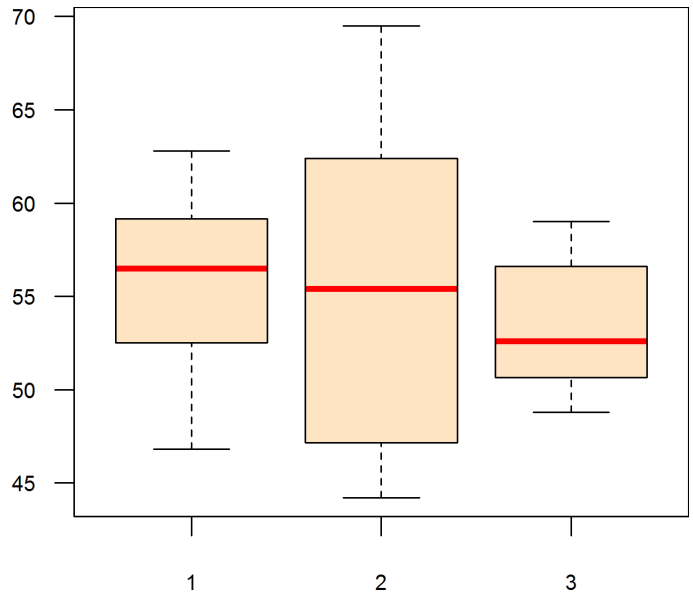
In order to get a good overview of the sample results, we join the groups no. 1 and 2, 3 and 4, 5 and 6 as follows:

```
levels(npk[,1])<-c(1,1,2,2,3,3)
names(npk)[c(1,5)]<-c("B","Y")
```

As a result of these commands we have also renamed the `block` and the `yield` variables. Thus there are 3 groups with respect of the `B` variable with 8 observations in every group, and the yield amount is contained by the variable `Y`. This can also be seen from the results of the following commands.

```
ls()
names(npk)
table(npk[,1])
```

The first show that there is indeed a new `npk` variable in '`.GlobalEnv`'. The result of the second is the list of the column names of this variable: `B`, `N`, `P`, `K` and `Y`. As the result of the last one, the observations can be divided into 3 groups with respect of the `B` grouping variable and we have 8 observations in each group.



Let us consider the modified dataset `npk` and investigate the pea yield in the 3 different levels of the variable `B` with the help of the boxplot command:

```
boxplot(Y~B,npk)
```

The three boxes with whiskers graphically show some of the basic statistics of the yields in the three groups. The red line corresponds to the median of the group, the orange boxes are bounded by the quartiles and the whiskers last till the minimum-maximum values in the groups.

We could have used the `range` parameters of the `boxplot()` command. This defines the maximum length of the whiskers in comparison of the box width (the interquartile range). Its default value is 1.5 times. If there are values outside of this range,

then these are drawn separately by circles.

We can calculate these statistics for each group by using a suitable parametrization of the `aggregate` command. The number of observations can be got as follows:

```
aggregate(npk$Y,npk["B"],length)
```

The others by replacing `length` by `median`, `IQR`, `mean` and `sd` respectively. The last column is the estimated standard deviation of the mean, i.e. $\texttt{SD}/\sqrt{\texttt{n}}$.

```
   N  Med   IQR  Mean   SD   SE
g1 8 56.5  4.85 55.74 5.27 1.86
g2 8 55.4 14.22 55.45 8.98 3.17
g3 8 52.6  5.22 53.44 3.63 1.28
```

Considering the previous figure and the above table, we cannot be sure that the expected value of the groups is different. The differences in the `Mean` are small, especially if we consider the `SE` as well. How can we formulate our question about the possible equality of the three groups more precisely and how can we give an exact answer to it?
Formulated for the `npk` dataset, the typical hypothesis of the `ANOVA` model is the following.

'Is it true, that the yield is equal in the blocks?'

Formulating this in the language of the hypothesis testing, we have two models for the data. The $\texttt{M}_0$ minimal model and the $\texttt{M}_1$ ANOVA model, as follows:

$\texttt{M}_0$: $y_{i,j} = \mu_0 + d_{i,j}$

$\texttt{M}_1$: $y_{i,j} = \mu_1 + \beta_i + e_{i,j}$

Here $i = 1, \ldots, g$ shows the group of the observation and $j = 1, \ldots, n_i$ the number of $y_{i,j}$ within the group. We have assumed that the observations can be divided into g groups, in the ith group there are n_i observations, the total number of observations is $\sum_{i=1}^{g} n_i = n$. We assume that the $d_{i,j}$ and the $e_{i,j}$ are realisations of such random variables, which are independent, with 0 mean, identical standard deviation and normal distribution.

Both models include a baseline value μ_0 resp. μ_1 for the observations, which does not depend on the group. The $\beta_1, \ldots, \beta_g$ values give the differences of the expectations in the group from this baseline value.

Without any further conditions the $\texttt{M}_1$ model is overparametrized. Let us ensure the uniqueness – in accordance to the default in **R** – so that we take a parameter system, where $\beta_1 = 0$.

The decision for the hypothesis can be given by the `ANOVA` table, which can be got by the command `(M<-aov(Y~B,npk))`

```
                        B Residuals
Sum of Squares      25.13    851.24
Deg. of Freedom         2        21

Residual standard error: 6.36672
Estimated effects may be unbalanced
```

The model we got can be summarised by the `summary(M)` command:

```
           Df  SumSq MeanSq Fvalue Pr(>F)
B           2  25.13  12.56   0.31   0.74
Residuals 21 851.24  40.54
```

This last result show that the p-`value` of the sample is `0.7368`. Thus it has a large probability, that under the same conditions taking the same number of observations, we get a larger F-value than the current one. Thus we do not have to reject the null hypothesis,

'Based on the one-factor ANOVA it can be accepted that the expected value of the yield is the same in the three blocks.'

Let us study the results in some detail!

The last table is called the decomposition of the Sum of Squares (SS). Its structure is almost like a standard within the area of the linear models. Quite often a last row is added, which contains the sums of the first two columns:

```
           Df  SumSq MeanSq Fvalue Pr(>F)
B           2  25.13  12.56   0.31   0.74
Residuals 21 851.24  40.54
Total     23 876.36
```

The `Residual` row corresponds to the 'Within Groups' and the `B` row to the 'Between Groups' sum of squares.

The `Df` column is the degree of freedom of the expressions in the given row . In the `B` (Between Groups) row we have 2, as the number of groups is $g = 3$, and the degree of freedom of this component is $g-1$.

In the `Residual` (Within Groups) row we have 21, as the number of observations is $n = 24$, and the

degree of freedom of this component is always $n-g$.

In the `MeanSq` column there are the mean squares. This is the ratio of the previous two columns. `25.13/2=12.56` and `851.24/21=40.54`.

The`Fvalue` is the ratio of the two `MeanSq` values: `Fvalue` $= 12.56/40.54 \approx 0.31$.

The last number in the table is a p-`value`. The probability that a random number from the $F_{2,21}$ distribution is larger than the `Fvalue` we have got is, $1 - \texttt{pf}(.31, 2, 21)$ which is indeed ≈ 0.737.

Namely the `Fvalue` has $F_{2,21}$ distribution under the given conditions, if the expected value of the three groups is equal. The result we have got can thus be interpreted as follows: the probability that the same statistics for a similar experiment under the equality of the group expectations is with a probability of ≈ 0.737 larger than the `Fvalue` we have got. This is a large probability, so we cannot reject the hypothesis of the equality. That we intend to reject this hypothesis if the `Fvalue` is large, can be deduced from the fact that the `Fvalue` is large if `SS.B` is large compared to `SS.R`.
We can get the estimated pea yield from the `aov` model for the groups by the `predict(M,data.frame(B=factor(1:3)))` command:

```
      1       2       3
55.7375 55.4500 53.4375
```

These estimators coincide with the group-averages, which we have already seen at the beginning of our analysis.

The `predict(M)` command calculates with the following coefficients:

```
(Intercept)      B2          B3
55.7375       -0.2875     -2.3000
```

This can be got by the
`coefficients(M)`
command, which can be abbreviated as `coef(M)`. As it can be seen, the baseline value (Intercept) is the average of the first group. The further coefficients are the differences between the group means of code '2' and '3' and the baseline.

The `oneway.test()` command

If we do not need the above mentioned more complicated statistics, just a hypothesis test about the equality of the expectations, then the command

```
oneway.test(Y ~ B, npk, var.equal=TRUE)
```

is also suitable. The result:

```
One-way analysis of means
Data:  Y and B
F=0.31, num.df=2, denom.df=21, p-value=.74
```

corresponds to the previous ones.

The `var.equal=TRUE` option means that the standard deviations are equal in the three groups with respect of `B`. If we use the default `FALSE` option for the equality of the standard deviations, then we get an answer by a method that does not use this equality.

Example 2: ANOVA in case of several factors

Until now, in the investigated cases, there was just one factor. If we have more grouping factors, then it can be checked if the expected value of the target variable is the same in the groups which may be got by taking into consideration more factors at the same time.

If we have two grouping variables `A` and `B`, which have r resp. c values, then the question can be about the equality of the expectations in the $r \times c$ groups.

But this question can be answered by the one-factor model. We just have to replace (`A`, `B`) by `X`, which is an artificial grouping variable with rc possible values, corresponding to the all possible level combinations of the two variables. And the investigation can be carried out as a one-factor ANOVA for `X`.

But this simplification causes two major potential problems:

This model has many more parameters than the one-way model. `X` has $1+(rc-1)$ parameters, while the two one-way models $1+(r-1)$ resp. $1+(c-1)$.

If we get that the groups do not have the same expectations, then we cannot decide, which factor (`R` or `C`) is the reason for this difference.

Thus we introduce a new model. This model is simpler than the product model we have just seen. Here we consider the effect of the factors *additively*. This results in an intermediate model, which has $\texttt{1}+(\texttt{r}-\texttt{1})+(\texttt{c}-\texttt{1})$ parameters.

Formally:

Let the observed values be $\xi_{i,j,k}$, $i = 1,...,r$, $j = 1,...,c$ $k = 1,...,m_{i,j}$, where $\xi_{i,j,k}$ has type i with respect of the first factor, type j according to the second, and it is the kth among these observations. So $m_{i,j}$ is the number of observation under the circumstance (i,j).

Now we suppose that $m_{i,j} = m$, i.e. there is an equal number of observations in each cell. Under this condition the design matrix of the observations is orthogonal. This ensures that the effect of a factor can be considered as being independent from the effects of the other factors.

The model, interpreting the target variable by `A`:

$$\xi_{i,j,k} = \mu + \alpha_i + \varepsilon_{i,j,k}$$

The model, interpreting the target variable by `B`:

$$\xi_{i,j,k} = \mu + \beta_j + \varepsilon_{i,j,k}$$

The model, interpreting the target variable by $\texttt{A}+\texttt{B}$:

$$\xi_{i,j,k} = \mu + \alpha_i + \beta_j + \varepsilon_{i,j,k}$$

We assume that the $\varepsilon_{i,j,k}$ errors are independent, have 0 expected value and the same standard deviation.

Thus for two possible factors, including the

$$\xi_{i,j,k} = \mu + \varepsilon_{i,j,k}$$

model, we have altogether five models, where the number of parameters is r, c, $r+c-1$, rc resp. 1. All of these models is an `ANOVA` model, and the two-way `ANOVA` method helps us to find the one, which is the simplest of these, with the property that neither of the more complicated ones are significantly better.

If we have more factors, then we choose the subsets of these explanatory variables on a way that if a subset is chosen, than all of its subsets are chosen as well.

If the effect belongs to a subset with more than one elements, then it is called *cross effect*. If the subset, corresponding to the effect has just one element, then it is called *main effect*. The number of elements is called the order of the effect. E.g. the previous γ is a second-order effect, while the main effects are first-order effects.

In the investigated model we approximate the observed value by the sum of the effects, which were considered. Thus in the model to each observed value there are exactly as many parameters as the number of the chosen factors.

We assume as usual, that the errors are independent, normally distributed with 0 expected value and σ standard deviation.

We show the main features of the method in a new data set.

The `ToothGrowth` data set has a size of 60×3. It shows the effect of the vitamin C (ascorbic acid) to the tooth growth of guinea pigs (`len`). (C.I. Bliss: The Effect of Vitamin C on Tooth Growth in Guinea Pigs, The Statistics of Bioassay, Academic Press, 1952).

The dose of the vitamin C (`dose`) had three levels (0.5, 1.0 and 2.0 mg). The source (`supp`) could be orange juice (`OJ`) or a pill (`VC`).

We have exactly 10 observations for each of the 3×2 possibilities, so this is an orthogonal design.

So if we assume that both factors have an effect, then it can be formulated as

`len~dose+supp`.

Let us suppose as above, that the observations are $\xi_{i,j,k}$ where $i = 1,2,3$, $j = 1,2$ and $k = 1,2,3,4$. The assumed model:

$$\xi_{i,j,k} = \mu + \alpha_i + \beta_j + \varepsilon_{i,j,k}$$

where μ is the baseline value, α is the effect of the dose with $\sum_{i=1}^{3} \alpha_i = 0$, and β is the effect of the

method (supp), with $\sum_{j=1}^{2}\beta_j = 0$, and $\varepsilon_{i,j,k}$ is the error.

Another model is

$$\xi_{i,j,k} = \mu + \alpha_i + \beta_j + \gamma_{i,j} + \varepsilon_{i,j,k}$$

where also the *interaction* is included. In order to ensure the uniqueness, we assume for the interaction that $\sum_{i=1}^{3}\gamma_{i,j} = 0$ and $\sum_{j=1}^{2}\gamma_{i,j} = 0$. We shall express this model in `aov` as

`len~dose*supp`.

The `ToothGrowth` data is also included in the `datasets` library. This time we create again a modified version, especially as in the original the `dose` variable is not a *factor*. And we also modify the name of the doses from '`0.5`', '`1`', '`2`' to '`L`', '`M`', '`H`'.

```
ToothGrowth$dose<-factor(ToothGrowth$dose)
levels(ToothGrowth$dose)<-c("L","M","H")
ToothGrowth$dose<-C(ToothGrowth$dose,sum)
ToothGrowth$supp<-C(ToothGrowth$supp,sum)
```

The parameters and the statistics can be calculated as follows:

```
M<-aov(len~dose+supp,ToothGrowth)
summary(M)
summary.lm(M)
```

The usual table with the decomposition of the sum of squares:

```
      Df  Sum.Sq Mean.Sq F.value Pr(>F)
dose   2 2426.43 1213.22  82.811 <2e-16
supp   1  205.35  205.35  14.017 0.0004
Res   56  820.43   14.65
```

The coefficients of the fitted model:

```
Coefficients:
         Estimate  S.E  t.value Pr(>|t|)
(Incept)  18.8133  0.4941  38.073 <2e-16
dose1     -8.2083  0.6988 -11.746 <2e-16
dose2      0.9217  0.6988   1.319 0.1926
supp1      1.8500  0.4941   3.744 0.0004
```

We have the following results:

- The table shows that both effects are significant.
- However, just one of the groups differs significantly from 0. Thus the mean value in group '2' can be considered as being equal to the baseline value.

To fit a model with interactions we need the following command:

```
M<-aov(len~dose*supp,ToothGrowth)
summary(M)
summary.lm(M)
```

The most important parts of the result:

The table for the decomposition of the sum of squares :

```
           Df  Sum Sq Mean Sq F.val Pr(>F)
dose        2 2426.43 1213.22  92.0 <2e-16
supp        1  205.35  205.35  15.6 0.0002
dose:supp   2  108.32   54.16   4.1 0.0219
Residuals  54  712.11   13.19
```

The parameters of the fitted model:

```
Coefficients:
          Estimate  S.E  t.value Pr(>|t|)
(Incept)   18.8133 0.4688  40.130 <2e-16
dose1      -8.2083 0.6630 -12.381 <2e-16
dose2       0.9217 0.6630   1.390 0.1702
supp1       1.8500 0.4688   3.946 0.0002
dose1:supp1 0.7750 0.6630   1.169 0.2476
dose2:supp1 1.1150 0.6630   1.682 0.0984
```

It must be mentioned, that

- as the plan is orthogonal, the estimators of those parameters do not change, that were already included in the narrower model.
- As the model has size 3×2, to the interaction there are $(3-1)(2-1) = 2$ parameters.
- It is interesting that while the table shows a significant interaction term, neither of its parameters differ significantly from 0.

The '`F.val`' columns are the ratio of the actual `Sum.Sq` and the `Residual Sum.Sq` – as before –, normed with the corresponding degrees of freedom.

Other functions, related to the `ANOVA` models:

`fitted()` or `fitted.values()`: the $\hat{y}$, i.e. the approximations to y by the model;
`proj()`: the same as the one before, but it gives the values of the factors as well;
`residuals()`: the differences $y - \hat{y}$ i.e. the error of the model in the observation points;
`model.tables()`: the model parameters (effects) in a tabulated form;

`model.tables(., "means")`: the default of this parameter is `effects` if we overwrite it as shown, then we get the averages for the groups of observations;

`model.tables(., se=TRUE)` (an option to the previous command): we get the standard deviations of the parameters as well.

References

[1] W.N. Venables, B.D. Ripley
Modern Applied Statistics with S, Fourth edition. (Springer, 2002)

[2] W.N. Venables, D.M. Smith and the R Core Team
Notes on R: A Programming Environment for Data Analysis and Graphics
An Introduction to R, `www.r-project.org`, (Version 3.2.2 (2015-08-14))

[3] Ryan T.A. at al
Minitab Student Handbook, (Duxbury Press, 1976)

[4] R.A. Johnson, G.K. Bhattacharyya
Statistics: Principles and Methods, (Wiley, 2014)

[5] M. Hollander, D. A. Wolfe
Nonparametric Statistical Methods, (Wiley, 1973)

[6] D.A. Belsley, E. Kuh and R.E. Welsch
Regression diagnostics, (Wiley, 1980)

[7] G. James, D. Witten, T. Hastie, R. Tibshirani
An Introduction to Statistical Learning with Applications in R, (Springer, 2014)