Business Intelligence

– SEMINAR WINTER SEMESTER 2012/2013 –

Introduction to the Statistical Software "R"

– SEMINAR PAPER –

Submitted by: **Jonas Tobias Schmitz**
Student-ID:
Advisor: **Prof. Dr. Dirk Neumann**

Table of Contents

1. Introduction ... 3
2. Installation of "R" ... 3
3. Installation of »Contributed Packages« ... 4
4. Simple use of "R" ... 4
4.1 Simple Operations ... 4
4.2 Variables and vectors ... 5
4.3 Tables ... 7
5. Working with data ... 9
5.1 Data Import from Excel ... 9
5.2 Handling of imported data ... 10
6. Coding and recoding of variables ... 11
6.1 Function »ifelse( )« ... 11
6.2 Function »recode( )« ... 12
6.3 Function »cut( )« ... 12
7. Simplifications ... 12
7.1 Predefinition of the search path – function »attach( )« ... 12
7.2 Presentation of the structure of an object – function »str( )« ... 13
7.3 Comprehensive summary of an object – function »summary( )« ... 13
7.4 Workspace ... 14
8. Graphics ... 14
8.1 High-level plotting functions ... 14
8.2 Low-level plotting functions ... 16
8.3 Predefinition of settings – command »par( )« ... 18
9. Conclusion ... 19
10. References ... 20

1. Introduction

The statistical software "R" is an internationally known open source software. "R" provides statistical and graphical functions, classical statistical tests and other data analysis functions. This seminar paper helps one to get started working with "R". Among others one will learn how to do simple operations and how to create tables and data frames. A special focus is placed on working with imported data. For one's work, it is essential to be able to import data as well as to format and to analyse data. Finally, a convenient graphical presentation of analysed data is of importance. In this paper, "R" commands that can be used directly in "R" are indicated in **red**. "R" output is indicated in **blue**.

2. Installation of "R"

The program "R" can be downloaded from the website http://www.cran.r-project.org/. First, one downloads the basic program »base«. In this paper, the installation instruction is limited to Windows operating systems. The link to download the Windows installation file is: http://www.cran.r-project.org/bin/windows/base/ (Figure 1).

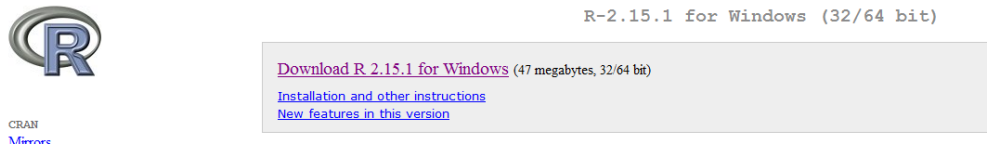


Figure 1. "R" website.

Install the program to a folder where you have full permission to access, e.g. your personal folder or a USB flash drive. During installation one can choose whether the program should run on a 32-bit version or a 64-bit version. One should select the appropriate version for his or her computer (Hatzinger et al. 2011). To find out whether one's system is running on a 32-bit version or a 64-bit version in Windows Vista or Windows 7 the »Windows button« and the »Break button« have to be pressed simultaneously to view the system type. To optimize one's work process, one can install »RStudio«, a development environment for "R". For more information consult the website www.rstudio.com/ide/.

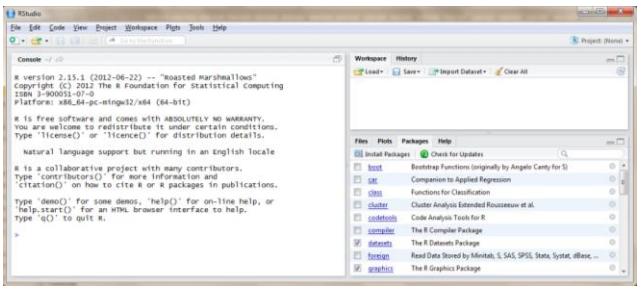


Figure 2. Software »RStudio«.

3. Installation of »Contributed Packages«

The reason for the success of the statistical software "R" is that "R" can be extended by any user with additional functions. Some features are included in the basic version of "R", while others can be downloaded from various archives. For example, one can download packages from the archive CRAN ("The Comprehensive R Archive Network"): http://www.cran.r-project.org/bin/windows/contrib/. For our seminar select the package »car«. One needs it to use the function »recode( )« (cf. Chapter 6.2). The package can be downloaded from http://cran.r-project.org/web/packages/car/index.html. The direct download-URL for Windows is http://cran.r-project.org/bin/windows/contrib/r-release/car_2.0-15.zip. To install the package, one opens "R" and selects in menu bar »*Packages*« ⇨ »*Install package(s) from local zip files…*«.

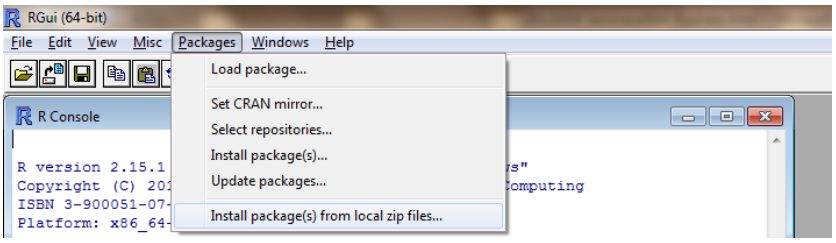


Figure 3. Installing additional packages.

Now one selects the downloaded ZIP-file. After a successful installation the following text appears:

```
> utils: menuInstallLocal ()
package 'car' successfully unpacked and MD5 sums checked
> |
```

To simplify work, it is also possible to download and install packages as follows: Having pressed »*Packages*« ⇨ »*Install package(s)…*«, one selects a mirror (e.g. »*Germany (Goettingen)*«) and the package one wants to install (e.g. »*car*«).

4. Simple use of "R"

4.1 Simple Operations

To run an operation in "R", one writes all commands into main control console. For instance, one can enter the operation »4+1«. "R" calculates the result:

```
> 4+1
[1] 5
```

If one has entered several operations and wants to call up a previous operation again, one can use the arrow keys ↑ and ↓ (Hardin 2012). It will be time-saving to use the arrow keys if one wants to adjust a complex operation. In case of an incomplete operation entry "R" does not output »Error«, it marks it with a »+«.

```
> (4+1/2
+
```

If one wants to cancel an entry of an operation, one has to press »Esc«. The basic arithmetic operations are to be entered as in Table 1.

Operation	**Input in "R"**	**Operating procedure of "R"**
Addition	`3+4`	$3 + 4 = 7$
Subtraction	`3-4`	$3 - 4 = -1$
Multiplication	`3*4`	$3 \cdot 4 = 12$
Division	`3/4`	$3 : 4 = 0.75$
Exponentiation	`3^4`	$3^4 = 81$
Exponential function	`exp(5)`	$e^5 = 148.4132$
Square root	`sqrt(2)`	$\sqrt{2} = 1.414214$
Logarithm to base 10	`log(3)`	$\log(3) = 1.098612$
Sine	`sin(3)`	$\sin(3) = 0.14112$
Cosine	`cos(3)`	$\cos(3) = -0.9899925$
Tangent	`tan(3)`	$\tan(3) = -0.1425465$

Table 1. Basic operations.

An entry of an operation in "R" is similar to an entry of an operation in a calculator. Therefore it is possible to combine operations: `cos(3+4)` = 0.7539023. Keep in mind that "R" is an international program: One uses a full stop instead of a comma, e.g. `0.75` stands for ¾.

4.2 Variables and vectors

It can be useful to store long numbers in variables to simplify work. The assignment of variables is done by the following command: `y <- 0.7539023`. One says »Y gets 0.7539023« (Hatzinger et al. 2011). By entering the variable name and pressing »Enter« one can check whether the data storage was successful:

```
> y <- 0.7539023
> y
[1] 0.7539023
```

Furthermore, it is possible to save evaluated operations in variables: `fish <- 3/cos(4)`. The newly defined variable »fish« attains the result of the operation $\frac{3}{\cos(4)}$ (Groß and Peters 2009). If one wants to title a variable »fish species«, the whitespace has to be replaced by a point: »fish.species«.

One can save nearly everything in variables: It is also possible to store vectors in variables. The function »c( )« generates a row vector (Shedden 2007). An assignment to a variable works in an analogous manner as above: `HEIGHT <- c(192,200,165)`. Subsequently, three values are stored in variable »HEIGHT«. If one wants to extend the vector without entering the previous values again, it can be done as follows: `HEIGHT <- c(HEIGHT, 170)`. In consequence, in variable »HEIGHT« four values are stored:

```
> HEIGHT
[1] 192 200 165 170
```

Due to the fact that some vectors are very long, it can be useful to display just the first elements of a vector to check whether the data storage was successful. The command »head( )« only prints the first few lines (Legendre 2011): `head(HEIGHT)`.

Strings are enclosed in double quotes. Either numbers or strings can be stored in a vector. If one wants to mix numbers and strings in one vector, "R" will define numbers as strings:

```
> AGE <- c("young" , "old",5)
> AGE
[1] "young"   "old"   "5"
```

Other operations with vectors are listed in Table 2.

Operation Explained by taking vector »HEIGHT«. Values of the vector: 192 200 165 170.	**Input in "R"**	**Output**
Number of entries	`length(HEIGHT)`	`4`
Minimum	`min(HEIGHT)`	`165`
Maximum	`max(HEIGHT)`	`200`
Minimum and maximum	`range(HEIGHT)`	`165 200`
Sum	`sum(HEIGHT)`	`727`
Mean value	`mean(HEIGHT)`	`181.75`
Standard deviation	`sd(HEIGHT)`	`16.89921`
Variance	`var(HEIGHT)`	`285.5833`
Output of the fourth element	`HEIGHT[4]`	`170`

Output of the second and of the fourth element	`HEIGHT[c(2,4)]`	`200 170`
Output of the second to the fourth element	`HEIGHT[2:4]`	`200 165 170`
Output of all elements except the third	`HEIGHT[-3]`	`192 200 170`
Output of all elements except the second to the fourth	`HEIGHT[-(2:4)]` (take care of the brackets)	`192`
Update of the third element to 185	`HEIGHT[3] <- 185`	no output (new stored vector: 192 200 185 170)

Table 2. Vector operations.

4.3 Tables

Vectors can be combined to form a table. There are two possibilities to structure it: Tables can be combined by columns and rows (Grüner 2005). For the next example, one creates two vectors: `STUDENT_NUMBER <- c(101,102,103,104)` and `AGE <- c(20,28,22,24)`.

Tables that consist of columns/rows can be generated as follows (whereas »c« stands for columns and »r« for rows). The column/row names are generated from the vector names.

```
> cbind(STUDENT_NUMBER,AGE)
     STUDENT_NUMBER  AGE
[1,]            101   20
[2,]            102   28
[3,]            103   22
[4,]            104   24
```

```
> rbind(STUDENT_NUMBER,AGE)
                [,1][,2][,3][,4]
STUDENT_NUMBER  101 102 103 104
AGE              20  28  22  24
```

One can save tables in variables to facility work: `Y <- rbind(STUDENT_NUMBER,AGE)` and `Z <- cbind(STUDENT_NUMBER,AGE)`. Working with tables one can use the commands on next page (Table 3). The commands will be explained by taking Y as an example.

Operation	Input in "R"	Output
Output of the second row and the third column	`Y[2,3]` alternative: `Y["AGE",3]`	`AGE` `22`
Output of the first three columns	`Y[,1:3]`	`[,1] [,2] [,3]` `STUDENT_NUMBER 101 102 103` `AGE 20 28 22`
Titling the columns	`colnames(Y) <-` `c("Julia","Sebastian",` `"Tina","Florian")`	`Julia Sebastian Tina Florian` `STUDENT_NUMBER 101 102 103 104` `AGE 20 28 22 24`
Output of column names	`colnames(Y)`	`"Julia" "Sebastian" "Tina" "Florian"`
Output of row names	`rownames(Y)`	`"STUDENT_NUMBER" "AGE"`
Output of the number of elements in a matrix	`length(Y)`	`8`
Output of the dimension of the matrix, whereas the first value is the number of rows (2) and the second value is the number of columns (4)	`dim(Y)`	`2 4`

Table 3. Table operations.

Indeed it is not possible to create a table that contains numbers and strings with the commands above. That is the reason why one can create a data frame instead of a table. A data frame is a table composed with several vectors all of the same length but possibly of different value types (Paradis 2005). The positive aspect is that one can use the same commands for data frames as well as for tables.

```
> AGE <- c(20,28,22,24)
> NAME <- c("Julia","Sebastian","Tina","Florian")
> data.frame(NAME,AGE)

        NAME    AGE
1      Julia     20
2  Sebastian     28
3       Tina     22
4    Florian     24
```

Henceforward tables, data frames and vectors are resumed as »objects«.

5. Working with data

5.1 Data Import from Excel

Data do not need to be entered directly in "R". Data can be entered into Excel, e.g. the results of a survey in which people are asked about their age, weight and gender (Table 4). As a decimal separator a point has to be used, e.g. 55.2 stands for 55 ⅕ (cf. Chapter 4.1). In Excel, one has to save the table as a CSV file (e.g. Excel-Data.csv). If one does not prefer to specify an answer option in a survey, »NA« (not available) has to be entered in Excel:

name	age	weight	sex
Julia	27	55	f
Romeo	27	85	m
Robinson	15	45	m
Christine	NA	65	f

Table 4. Sample data.

To import the data-file into "R", one uses the following command:

```
read.csv("C:/Users/path-to-file/Excel-Data.csv",header=TRUE,sep=";")
```

It is important to specify the command: One has to indicate if the CSV file has headings or not (header=TRUE or FALSE). In »Excel-Data.csv« the separator is a semicolon. One has to indicate it with »sep=";"«. If one opens the CSV file with a text editor, one can look up the separator (ETH Zürich 2009). One should remember that nearly everything can be saved in variables (chosen name of variable is »dataX«), e.g. `dataX <- read.csv(_____)`.

Afterwards one can display the data frame by entering the name of the variable and pressing »Enter«. If one wants to display just one column, the name of the data frame, a dollar sign and the name of the column has to be entered: `name.of.variable$name.of.column`.

```
>dataX
       name age weight sex
1     Julia  27     55   f
2     Romeo  27     85   m
3  Robinson  15     45   m
4 Christine  NA     65   f
```

```
> dataX$age
27 27 15 NA
```

5.2 Handling of imported data

To work with imported files one can use the same commands as those of tables (cf. Chapter 4.3). In addition to it, the data frame from Chapter 5.1 is used to become acquainted with the following commands. The data frame »dataX« includes the columns »name«, »age«, »weight« and »sex«.

Operation	**Input in "R"**
Output of age of all male persons	`subset(dataX$age,dataX$sex=="m")`
Mean age of all male persons	`mean(subset(dataX$age,dataX$sex=="m"))`
Output of all men who are at least 21 years old	`subset(dataX, (sex=="m") & (age >= 21))`
Output of the columns »name« and »weight« of all women	`subset(dataX, select=c(name,weight), +(sex=="f"))`
Output of all columns except the column »age«	`subset(dataX, select = -c(age))`
Output of the first three columns	`subset(dataX, select=c(name:weight))`
Add a column »weight_in_pounds«	`dataX <- cbind(dataX,"weight_in_pounds"= dataX$weight*2.20462262)`
Overwrite column »weight« with the values from »weight_in_pounds«	`dataX <- transform(dataX, weight = weight_in_pounds)`
Delete the column »weight_in_pounds«	`dataX$weight_in_pounds <- NULL`
Round the numbers in column »weight« to two decimal places	`dataX <- transform(dataX, weight=round(dataX$weight, digits=2))`

Add a row with the name »Richard« (Janis and Wickham 2007)	`dataX <- rbind(dataX,data.frame(name="Richard", age=90, weight=120,sex="m"))`
Delete the row »Richard«	`dataX <- dataX[-5,1:4]`

Table 5. Generating subsets of data frames.

In case of operation »Output of all men who are at least 21 years old« a greater-than sign is used. Moreover, the following signs can be used: One can use `<` for »less than«, `<=` for »less or equal«, `==` for »equal«, `>=` for »greater or equal«, `!=` for »unequal«, `|` for »or« and `!` for »not« (Hatzinger et al. 2011).

From a data set subgroups can be selected. It is possible to save all data from the data set which belongs to women in a new variable: `femaleDATA <- subset(dataX,dataX$sex=="f")`. With the help of the new list »femaleDATA« one can determine the mean weight of all women:

```
> mean(femaleDATA$weight)
[1] 60
```

In Chapter 5.1 it was discussed how to deal with empty data fields. For further use of incomplete data the options in Table 6 are available.

Operation	**Input in "R"**
Creating a table without the rows with blank fields (line »Christine« will be removed)	`NewTable <- na.omit(dataX)`
Checking for all elements if one element is missing and generating a vector to report	`is.na(dataX)`

Table 6. Dealing with empty data fields.

6. Coding and recoding of variables

It may happen that one wants to adjust variables for output. For instance, there exists a vector P with examination marks: `P <- c(1.3,2,2,3,5)`. The examination marks should be recoded as follows: examination mark 1 to 4 »passed«, examination mark 5 »failed«, examination mark 6 »exmatriculated«.

6.1 Function »ifelse( )«

A simple problem-solving approach is the use of an »ifelse( )« function. First, the function tests a condition. If the condition is fulfilled, »action A« will be realized (e.g. a text output). If the reverse is true and the condition is not fulfilled, »action B« will be realized (Heumann 2009).

```
> ifelse(condition to prove, action A, action B)
```

»ifelse( )« function for the example above:

```
> ifelse(P>=1 & P<=4,"passed", ifelse(P<=5,
"failed","exmatriculated"))
```

6.2 Function »recode( )«

Another way to recode is the use of the command »recode( )« (Field 2011). This feature is included in package »car« (cf. Chapter 3). Each examination mark is assigned to a direct transformation. Undefined examination marks will be ignored in process of recoding – this is a difference to the method of operation of an »ifelse( )« function (cf. Chapter 6.1).

```
> recode(P, '1:4="passed"; 5="failed"; 6="exmatriculated"')
```

6.3 Function »cut( )«

The third method to recode a variable can be realised by using the function »cut( )«. This function divides the data of a vector into parts. It is selectable whether one will set bounds to define parts or one will divide the vector into certain equal parts (e.g. three equal parts). The first limit "R" uses to set the lowest bound automatically is the smallest value of the vector, the last limit "R" uses to set the highest bound automatically is the biggest value of the vector (Hatzinger et al. 2011).

For instance, the function can be implemented as follows. For clarity, the limits are initially stored in a variable: `bounds <- c(1,4,5,6)`.

```
> cut(P, bounds, labels = c("passed","failed","exmatriculated"))
[1] passed passed passed passed failed
Levels: passed failed exmatriculated
```

As output one gets the recoding of the examination marks as well as the three categories (»Levels«) one has set.

7. Simplifications

7.1 Predefinition of the search path – function »attach( )«

It is not always convenient to write long commands, for instance, if one wants to output all men in data set »dataX« (cf. Chapter 5.1):

```
> dataX$weight[dataX$sex=="m"]
[1] 85 45
```

For that reason the »attach( )« function takes a data frame as its argument (Venables et al. 2012). After using the »attach( )« function, "R" will only search for variables in the data set »dataX«. The name of the data set does not need to be typed in any longer. The new search instruction to output all men in data set »dataX« is:

```
> attach(dataX)
> weight[sex=="m"]
[1] 85 45
```

The command `detach(dataX)` will reset the predefined search path (Groß and Peters 2009).

7.2 Presentation of the structure of an object – function »str( )«

The function »str( )« gives information about the structure of objects (Stahel 2005). The best way to become acquainted with this function is to pay attention to an illustration (data frame »dataX«, cf. Chapter 5.1).

```
> str(dataX)
'data.frame':   4 obs. of  4 variables:
 $ name  : Factor w/ 4 levels "Christine","Julia",..: 2 4 3 1
 $ age   : int  27 27 15 NA
 $ weight: int  55 85 45 65
 $ sex   : Factor w/ 2 levels "f","m": 1 2 2 1
```

It is apparent from the first line of output that »dataX« is a data frame. In »dataX« four variables are included in each case with four observations. This is followed by the explicit output of the variables (»$name«, »$age«, …). One can see that the variables »age« and »weight« are defined as integers, whereas »name« and »sex« are defined as factors. Additionally, it is displayed how many characteristics a factor has. For instance, factor »sex« has two characteristics: female and male.

7.3 Comprehensive summary of an object – function »summary( )«

To get an overview of the data one can use the »summary( )« function. The »summary( )« function prints a comprehensive summary of an object (Venables 2012). For analysing, complete data frames (e.g. `summary(dataX)`, data frame »dataX«, cf. Chapter 5.1) can be selected as well as single columns (e.g. `summary(dataX$age)`).

```
> summary(dataX)
        name         age          weight     sex
 Christine:1   Min.   :15   Min.   :45.0   f:2
 Julia    :1   1st Qu.:21   1st Qu.:52.5   m:2
 Robinson :1   Median :27   Median :60.0
 Romeo    :1   Mean   :23   Mean   :62.5
               3rd Qu.:27   3rd Qu.:70.0
               Max.   :27   Max.   :85.0
               NA's   :1
> summary(dataX$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
     15      21      27      23      27      27       1
```

7.4 Workspace

In the previous working process several variables were generated. Any variables were saved in workspace. They are available as long as the program is open. If one closes the program, the workspace will also be cleared in case one does not store the workspace (»*File*« ⇨ »*Save Workspace…*«). With a resumption of work, workspace can be recalled (»*File*« ⇨ »*Load Workspace…*«). The complete workspace can be displayed by entering the command **`ls( )`**. Individual objects in the workspace can be deleted by entering the command **`rm(objectname_one, objectname_two)`** (Faußer 2008).

8. Graphics

For the visual presentation of data and results there are a wide range of functions and contributed packages (cf. Chapter 3). In "R" one distinguishes between high-level plotting functions with which one can create complete graphics and low-level plotting functions. With the help of low-level plotting functions one is able to assemble a graph with individual elements piece by piece. This makes it possible to create more flexible and sophisticated graphics (Groß and Peters 2009).

8.1 High-level plotting functions

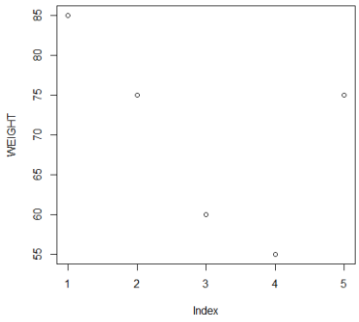
High-level plotting functions create a complete plot of one's data (Venables et al. 2012). To become acquainted with the functions, one wants to visualise properties of several people in graphics. One stores the data in vectors (»NAME« is a factor object, »AGE« and »WEIGHT« are numeric vectors):

```
> NAME <- factor(c("Person A", "Person B", "Person C","Person
D", "Person E"))
> AGE <- c(24,25,27,27,32)
> WEIGHT <- c(85,75,60,55,75)
```

vector »WEIGHT« visualised

in a bar diagram

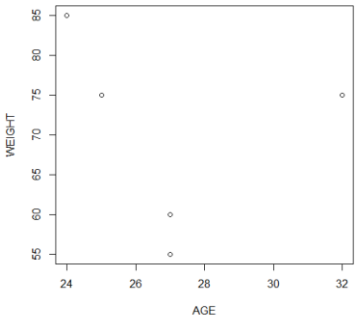
```
plot(WEIGHT)
```



»AGE« visualised as a function of »WEIGHT«

in a scatterplot

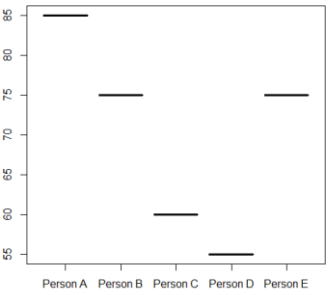
```
plot(AGE,WEIGHT)
```



»WEIGHT« visualised on the y-axis

and »NAME« on the x-axis

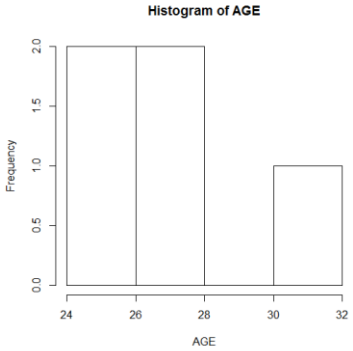
```
plot(NAME,WEIGHT)
```



»AGE« visualised

as a histogram

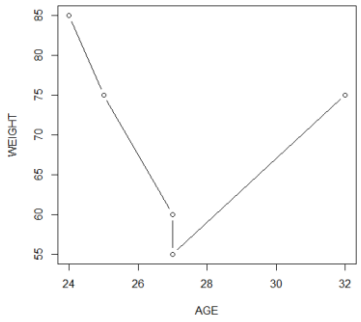
```
hist(AGE)
```



»AGE« visualised as a function of »WEIGHT« in a scatterplot

and the plot points are connected by lines

```
plot(AGE,WEIGHT, type="b")
```



Shift the type to change the style, e.g. »type="h"« plots vertical lines from x-axis to the points (Venables et al. 2012).

Figure 4. Sample plots.

Furthermore, one can plot functions. For instance, one visualises the function $f(x) = x^2$ whereby the x-intercept shall be displayed from -15 to 15. Therefore one creates a sequence from -15 to 15 and saves the sequence in a variable. The step size in the sequence should be defined in $\frac{1}{100}$-steps. Finally, one can plot the function.

```
> x <- seq(from=(-15), to=15, by=0.01)
> plot(x,x^2)
```

Axes, labels and headings are automatically generated unless one requests otherwise (Venables et al. 2012). If one likes to generate axes, labels and titles on one's own, the commands in Table 7 can be useful.

Operation	**Implementation in "R"**	**Example of input**
Deactivate axis	axes = FALSE	`plot(AGE,WEIGHT,axes=FALSE)`
Label the x-axis and the y-axis	xlab=" *label name X* " ylab=" *label name Y* "	`plot(AGE,WEIGHT,xlab ="AGE OF OUR GROUP", ylab="WEIGHT OF OUR GROUP")`
Restrict the area that should be visualised	xlim=c(*x1,x2*) ylim=c(*y1,y2*)	`plot(AGE,WEIGHT, xlim=c(26,30), ylim=c(50,70))`
Add a heading and a subtitle	main ="*heading*" sub="*subtitle*"	`plot(AGE,WEIGHT,main="Diagram")` `plot(AGE,WEIGHT,sub="Diagram")`

Table 7. Customization of high-level plotting functions.

8.2 Low-level plotting functions

Due to the fact that high-level plotting functions do not always produce the desired result, one can use low-level plotting functions. Using low-level plotting functions, one can add elements or one creates a graphic from scratch (Hatzinger et al. 2011).

Based on the diagram generated above (`plot(AGE,WEIGHT)`) one wants to add the appropriate names to the points in the scatterplot. To add labeling one enters the command `text(AGE,WEIGHT,labels=NAME,pos=2)` (Figure 5). »pos=« specifies the positioning of the label. Position 1 adds the label below the points, position 2 adds it to the left, position 3 above and position 4 to the right (Stahel 2009).

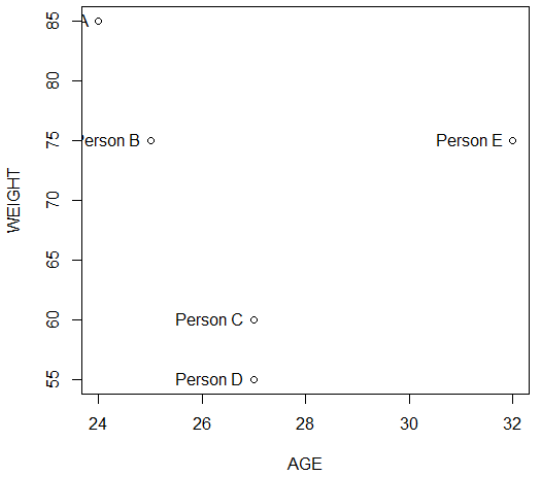


Figure 5. Labeling the points.

Due to the fact that the labels are not formatted satisfactorily, one wants to create the diagram on one's own. Firstly, one creates a blank chart optimized for the values in the vectors: `plot(AGE,WEIGHT,type="n",axes=TRUE,xlim=c(20,35),ylim=c(50,90),xlab="",ylab="")`. Secondly, one labels the axes: `mtext("weight",side=2,line=3)` and `mtext("age",side=1,line=3)`. »line=« indicates how far the label should be away from the axis. »side=« defines the axis: Starting from the bottom (side=1), it goes clockwise to side 4 (Kim 2004). To plot the values one uses the command `points(AGE,WEIGHT,pch=19)`. Whereas »pch=« defines the point character, e.g. number 2 stands for Δ (Chihara 2011). The appropriated labeling occurs as stated above: `text(AGE,WEIGHT,labels=NAME,pos=3)`. To ensure that one can read out the values of »weight« out of the graph, one wants that every value of vector »WEIGHT« is marked with a tick mark on the y-axis: `axis(side=2,at=WEIGHT)` (Fox and Weisberg 2012). Finally, one has recreated the graph with low-level plotting functions even optimized (Figure 6).

Figure 6. Graph created with low-level plotting functions.

For further optimization of graphs, the low-level plotting functions in Table 8 can be useful.

Description	Implementation	Example
Add a straight vertical line Add a straight horizontal line Add the straight line f(x)=mx+c (Chihara 2011)	abline(v=___) abline(h=___) abline(*c*,*m*)	`abline (v=60)` `abline(h=30)` `abline(50,1)`
Add a line connecting points (Nenadić and Zucchini 2004)	lines(*x*,*y*)	`y <- c(50,90)` and `x <- c(20,35)` `lines(x,y)`
Define line type	lty=*number of line type* *1 = blank, 2 = solid, 3 = dashed, 4 = dotdash, 5 = longdash, 6 = twodash (Plank 2010)*	`abline(h=60, lty=5)` alternative: `abline(h=60,lty="longdash")`
Define line thickness	lwd(*thickness*)	`abline(h=85, lwd=3)`
Define colour	col= for a list of all colours enter **>colours( )** into R	`abline(h=70, col="blue")`
Define text size in relation to standard size (e.g. 90 %)	cex=	`mtext("weight",side=2,cex=0.9)`

Table 8. Low-level plotting functions.

8.3 Predefinition of settings – command »par( )«

For the simple reason that the formatting of graphs should be consistent, but the default setting does not always corresponds to wishes, one can predefine independently designs with the command »par( )«. Most commands only change settings temporarily, whereas changes via »par( )« persist until either one issues a further »par( )« command to change settings or closes "R" (Dillon 2011). For instance, one wants to default the line type to »twodash« and the default colour to »blue«, it can be implemented as follows: `par(lty=6,col="blue")` (Plank 2010). Working with several settings, it arises to store settings in variables to enable a faster change, e.g. `setting_one <- par(lty=6,col="blue")` and `setting_two <- par(lty=4,col="red")`. As a consequence the setting can be called as described above, e.g. `par(setting_one)`.

Prior to the conversion of default settings, it is advisable to store the present parameters in a variable. The command `par(no.readonly=TRUE)` can be used to show a full list of

parameters that can be restored (Hatzinger et al. 2011). The basic settings can be saved in a variable as follows: `safety.variable <- par(no.readonly=TRUE)`.

9. Conclusion

All in all one received an overview of basic structures of "R" and commands in "R". This serves as a respectable basis for one's following seminar paper. If one needs more commands than explained, one should keep in mind that "R" is an international used program: Due to this fact a lot more commands can be found in the Internet.

10. References

Chihara, L., 2011, *R Tutorial on plotting in R*, http://www.people.carleton.edu/~lchihara/Splus/RPlot.pdf, 19.09.2012

Dillon, M., 2011, *Hands on R: Advanced Plotting*, http://www.uwyo.edu/mdillon/hor/hor_adv_plotting.pdf, 19.09.2012

ETH Zürich, 2009, *Daten einlesen mit R*, ftp://stat.ethz.ch/WBL/R-Einstieg/Daten_Einlesen_mit_R.pdf, 19.09.2012

Faußer, S., 2008, *Einführung in die statistische Sprache R*, http://www.informatik.uni-ulm.de/ni/Lehre/WS11/DMM/RIntro.pdf, 25.09.2012

Field, A. P., 2011, *The R environment*, http://www.statisticshell.com/docs/dsur_ch3_excerpt.pdf, 24.09.2012

Fox, J., Weisberg, S., 2012, *Package 'car': Companion to Applied Regression*, http://cran.r-project.org/web/packages/car/car.pdf, 25.09.2012

Groß, J., Peters, B., 2009, *R Reader: Arbeiten mit dem Statistikprogramm R*, http://cran.r-project.org/doc/contrib/Grosz+Peters-R-Reader.pdf, 18.09.2012

Grüner, E., 2005, *Statistik mit R*, http://www.staff.uni-marburg.de/~gruener/lehre/r.w05/folien/r_051208.pdf, 24.09.2012

Hardin, J., 2012, *An R Tutorial*, http://pages.pomona.edu/~jsh04747/courses/RTutorial.pdf, 24.09.2012

Hatzinger, R., Hornik, K., Nagel, H., 2011, *R: Einführung durch angewandte Statistik*, 1, München: Pearson Studium

Heumann, C., 2009, *Programmieren in R: Bedingte Anweisungen, Schleifen, Funktionen*, http://www.stat.uni-muenchen.de/~chris/rkurs/Rkurs3.pdf, 24.09.2012

Janis, M., Wickham, H., 2007, *Adding a factor level to an existing data.frame*, http://rwiki.sciviews.org/doku.php?id=tips:data-factors:add_a_factor_level_to_existing_data.frame, 17.09.2012

Kim, D.-Y., 2004, *MAT 356 R Tutorial*, http://math.illinoisstate.edu/dhkim/rstuff/rtutor.html, 19.09.2012

Legendre, P., 2011, *Introduction to the R statistical language*, http://biol09.biol.umontreal.ca/BIO6077/Introduction_to_R.pdf, 24.09.2012

Nenadić, O., Zucchini, W., 2004, *Statistical Analysis with R: a quick start*, http://www.statoek.wiso.uni-goettingen.de/mitarbeiter/ogi/pub/r_workshop.pdf, 23.09.2012

Paradis, E., 2005, *R for Beginners*, http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf, 24.09.2012

Plank, A., 2010, *Grafiken und Statistik in R*, http://www.geo.fu-berlin.de/geol/fachrichtungen/pal/mitarbeiter/plank/Formeln_in_R.pdf, 19.09.012

Shedden, K., 2007, *Overview of R*, http://www.stat.lsa.umich.edu/~kshedden/Courses/Stat600/Notes/R_introduction.pdf, 24.09.2012

Stahel, W., 2005, *Einführung in die Statistik-Umgebung R für angewandte Vorlesungen an der ETHZ*, http://stat.ethz.ch/~stahel/courses/R/R-einf.pdf, 24.09.2012

Stahel, W., 2009, *Einführung in die Statistik-Umgebung R*, http://stat.ethz.ch/~stahel/courses/R/usingr-script-d.pdf, 25.09.2012

Venables, W. N., Smith, D. M., the R Core Team, 2012, *An Introduction to R Notes on R: A Programming Environment for Data Analysis and Graphics*, http://cran.r-project.org/doc/manuals/R-intro.pdf, 18.09.2012