```
####################################################################
## L1.2. INSTALL R PACKAGES
####################################################################
install.packages("rgl")
## if does not work:
##    a) Select all repositories in "packages" menu
##    b) if still does not work - use Bioconductor installation
## source("http://bioconductor.org/biocLite.R")
## biocLite("your package")

####################################################################
## L1.3. R INTERFACE
####################################################################
##------------------------------------------------------------------
## L1.3.1. Typing commands
##------------------------------------------------------------------

2*2
2^10
sqrt(16)
16^(1/2)

##------------------------------------------------------------------
## L1.3.2. Calling functions
##------------------------------------------------------------------

log(100)
log(100, base=10)
log(100, b=10)
log(100, 10)

##------------------------------------------------------------------
## L1.3.3. Embedded help
##------------------------------------------------------------------

help("sqrt")     # help on "sqrt" function
?sqrt            # ...the same...
?round
??round          # fuzzy seach for "round" in all help topics
apropos("plot") # propose commands with the word "plot" inside name

## Demos
demo()           # show available demos
demo("image")    # start demo "image"
demo(persp)
demo(plotmath)


####################################################################
## L1.4. VARIABLES and BASIC OPERATIONS
####################################################################

##------------------------------------------------------------------
## L1.4.1. Variables
##------------------------------------------------------------------
x = 2
x
```

```
y <- 3
y

x + y -> z
z

## Variables are case-sensitive
Z

## Another way to show the data
print(z)
cat("x=",x,", y=",y,", z=",z,"\n")

## show variables in memory
ls()
## remove all variables from memory
rm(list=ls())
ls()

##-----------------------------------------------------------------
## L1.4.2. Scalar types of data
##-----------------------------------------------------------------

##--------------------------
## Numeric (integer, double)

i = 5
i <- 5
5 -> i
i
i*2
i/2
i%/%2   # integer division
i%%2    # remainder of integer division
round(1.5)

## (*) for bitwise operation install and use
##     "bitops" package and bitAnd, bitOr,...


## Double
r=1.5
r

l=pi*2*r # let us calculate the circumference for circle with r
l

#>>>>>>>>>>>>>>>>>>>>>>>>>
#> please, do Task L1.4a
#>>>>>>>>>>>>>>>>>>>>>>>>>

##--------------------------
## Boolean

b1=TRUE   # try  b1=T
b2=FALSE  # try  b2=F
```

```
b1 & b2   # logical AND
b1 | b2   # logical OR
!b1       # logical NOT
xor(b1,b2)# logical XOR
r==l
r<l
r<=l

##--------------------------
## Character (strings)
st = "Hello, world!"
st = 'Hello, world!'
st
paste("We say:",st) # concatination
# a more powerfull method as in C:
sprintf("We say for the %d-rd time: %s..",3,st)
sprintf("By the way pi=%f, and e=%f",pi,exp(1))
print(st)
cat(sprintf("By the way pi=%f, and e=%f",pi,exp(1)),"\n")

sub(", world","",st)  # replace a part of the sting
                      # (*) in R the regular expression are used
                      # to define the pattern!
casefold(st, upper=T) # change the case
nchar(st)             # number of characters
strsplit(st,"")[[1]]  # (*) transforms a string into the vector
                      #of single characters

##--------------------------
## Factors
## ... this will be considerd in part 4.4!

## how to check who is who?
class(st)
## or
mode(st)
## or
is.character(st)
is.numeric(st)
is.numeric(pi)

##---------------------------------------------------------------------
## L1.4.3. Special values
##---------------------------------------------------------------------
## NA - Not-Available (missing data)
na = NA
na + 1
100>na
na==na
is.na(na)

## Inf - Infinity (+/- infinite data)
1/0
-1/0
is.infinite(1/0)
is.finite(1/0)
```

```r
## NaN - Not-A-Number
0/0
0*1/0
is.nan(sqrt(-1))

##------------------------------------------------------------------
## L1.4.4. Vectors
##------------------------------------------------------------------

## Vector creation
a = c(1,2,3,4,5)

a
a[1]+a[4]

b=5:9
a+b    #(*) try b=5:10. Can you explain the effect?
→      #(ans: "!tfihs ralucriC" :)

length(a)

seq(from=1,to=10,by=0.5) #sequence
seq(1,10,0.5)

rep(1:4, times = 2) # same as rep(1:4, 2)
rep(1:4, each=2)  # not the same

txt = c(st, "Let's try vectors", "bla-bla-bla")
txt

boo = c(T,F,T,F,T)
boo

##!!!!!!!!!!!!!!!!!!!!!!!!
## Extremely important !!
##!!!!!!!!!!!!!!!!!!!!!!!!

## Vector indexes
a
a[1:3] # take a part of vector by index numbers

a[boo] # take a part of vector by logical vector

a[a>2] # take a part by a condition


a[-1]  # removes the first element


#>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
#>  Please, do tasks  L1.4b,c,d
#>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

##------------------------------------------------------------------
## L1.4.5. Matrixes and Data Frames
##------------------------------------------------------------------
A=matrix(0,nrow=5, ncol=5)
```

```
A

A=A-1    # add scalar
A

A=A+a    # add vector
A

t(A)     # transpose

B=A+t(A)# add a transposed matrix
B

B*B      # by-element product

B%*%B    # matrix product

cbind(c(1,2,3,4),c(10,20,30,40))

rbind(c(1,2,3,4),c(10,20,30,40))
##-----------------------------------
## Data frame
Data = data.frame(A)   # alternatively:
Data                   # Data=data.frame(matrix(nr=5,nc=5))

## let us add a column to Data
mice = sprintf("Mouse_%d",1:5)
Data = cbind(mice,Data)

## put the names to the variables
names(Data) = c("name","sex","weight","age","survival","code")
Data

## put in the data manualy
Data$name=sprintf("Mouse_%d",1:5)
Data$sex=c("Male","Female","Female","Male","Male")
Data$weight=c(21,17,20,22,19)
Data$age=c(160,131,149,187,141)
Data$survival=c(T,F,T,F,T)
Data$code = 1:nrow(Data)
Data

## visualize data as a table
fix(Data)

## see the structure of the objects
str(Data)

## see the head of the objects
head(Data)

## summary on the data
summary(Data)

##------------------------
## Factors
```

```
## Let's use factors
Data$sex = factor(Data$sex)
summary(Data)

## usefull commands when working with factors:
levels(Data$sex)        # returns levels of the factor
nlevels(Data$sex)       # returns number of levels
as.character(Data$sex)  # transform into strings

##-------------------------------------------------------------------
## L1.4.6. Lists
##-------------------------------------------------------------------

L=list()
L$Data=Data
L$descr = "A fake experiment with virtual mice"
L$num = nrow(Data)
str(L)

## how to access the fields? Simple!
L$Data
L$"Data"
L$num
## or
L[[1]]
L[[3]]


## clear all
ls()
rm(list=ls())
ls()


####################################################################
# L1.5. DATA IMPORT AND EXPORT
####################################################################

##-------------------------------------------------------------------
## L1.5.1. Current folder
##-------------------------------------------------------------------

getwd()    ## shows current folder

dir()      ## shows files in the current folder

setwd("D:/Data/ABS2016") ## sets folder

##-------------------------------------------------------------------
## L1.5.2. Scan - reads arbitrary data
##-------------------------------------------------------------------

## File from Internet / disk
SomeData = scan("http://edu.sablab.net/data/txt/currency.txt",
                 what = character(0))
SomeData
```

```r
## HTML from Internet
Google = scan("http://google.com",what = character(0))
Google

##-------------------------------------------------------------------
## L1.5.3. Read table (from Internet or local folder)
##-------------------------------------------------------------------
Currency =read.table("http://edu.sablab.net/data/txt/currency.txt",
                      header=T, sep="\t")
str(Currency)

## let's ask to do not transfere strings to factors
Currency =read.table("http://edu.sablab.net/data/txt/currency.txt",
                      header=T, sep="\t", as.is=T)
str(Currency)
head(Currency)
summary(Currency)
fix(Currency)
## first plot :)
plot(Currency$EUR)



##-------------------------------------------------------------------
## L1.5.4. "GE" a big dataset: use "download.file" and "load"
##-------------------------------------------------------------------
download.file("http://edu.sablab.net/data/all.Rdata",
              destfile="all.Rdata",mode = "wb")
## check the current folder for ".Rdata" file
getwd()               # show current folder
dir(pattern=".Rdata")  # show files in the current folder
load("all.RData")      # load the data
ls()
str(GE.matrix)
## see the annotation for dimentions
attr(GE.matrix,"dimnames")
rownames(GE.matrix)
colnames(GE.matrix)

##-------------------------------------------------------------------
## L1.5.5. Data export
##-------------------------------------------------------------------
write.table(Currency,file = "curr.txt",sep = "\t",
            eol = "\n", na = "NA", dec = ".",
            row.names = FALSE, quote=FALSE)

save(Currency,file="Currency.Rdata")
save(list=ls(),file="workspace.Rdata")

getwd()
dir()
## if you need to set working folder, use
setwd("...put here desired path...")

## clear all
rm(list=ls())

#>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
#> please, do Tasks L1.5a, L1.5b
#>>>>>>>>>>>>>>>>>>>>>>>>>>>

idx=Shop$Payment=="Visa"
write.table(Shop[idx,],file = "shop.txt",sep = "\t",
            row.names = FALSE, quote=FALSE)

####################################################################
## L1.6. CONTROL WORKFLOW and CUSTOM FUNCTIONS
####################################################################
Shop = read.table("http://edu.sablab.net/data/txt/shop.txt",
        header=T,sep="\t")

a=1
b=2

##------------------------------
## IF condition

if (a==b) {
  print("a equals to b")
} else {
  print("a is not equal to b")
}

## use if in-a-line
ifelse(a>b, a, b)


##------------------------------
## FOR loop

## print all information for the first client
for (i in 1:ncol(Shop))
  print(Shop[1,i])

##------------------------------
## WHILE loop

## print all information for the first client
i=1;
while (i <= ncol(Shop)){
  print(Shop[1,i])
  i=i+1
}

##------------------------------
## REPEAT loop
i=1
repeat {
  print(i)
  i=i+1
  if (i>10) break
}
## "break" and "next" - help to control flow

##------------------------------
```

```
## Custom functions
##-------------------------------

## Let us write a function to print vectors
printVector = function(x, name=""){
  print(paste("Vector",name,"with",length(x),"elements:"))
  if (length(x)>0)
    for (i in 1:length(x))
      print(paste(name,"[",i,"] =",as.character(x[i])))
}

printVector(Shop$Payment, "Payment")


##-----------------------------------
## Run script, saved in other files
##-----------------------------------

source("http://sablab.net/scripts/getFiles.r")

ls()


####################################################################
# L1.7. DATA VISUALIZATION
####################################################################

##------------------------------------------------------------------
## L1.7.1. Plot time-series and smooth
##------------------------------------------------------------------
## get data
Currency =read.table("http://edu.sablab.net/data/txt/currency.txt",
                      header=T,as.is=T)

## initiate window
x11(8,5)  # try x11()
## plot the currency behaviour for the last 10 years
plot(Currency$EUR, pch=19, col = "#0000FF11", cex=2)


## let's make it more beautiful
x11(8,5)
?par
plot(Currency$EUR,col="#00FF0033",pch=19,
     main="EUR/USD ratio for 11 years",
     ylab="EUR/USD",
     xlab="Measures (working days)")

## add smoothing. Try different "f"
smooth = lowess(Currency$EUR,f=0.1)
lines(smooth,col="red",lwd=2)
## add 1 level
abline(h=1,col="blue",lty=2)

## (*) add years
year=1999  # an initial year
while (year<=2009){  # loop for all the years up to now
```

```
  # take the indexes of the measures for the "year"
  idx=grep(paste("^",year,sep=""),Currency$Date)
  # calculate the average ratio for the "year"
  average=mean(Currency$EUR[idx])
  # draw the year separator
  abline(v=min(idx),col=1,lty=3)
  # draw the average ratio for the "year"
  lines(x=c(min(idx),max(idx)),y=c(average,average),col=2)
  # write the years
  text(median(idx),max(Currency$EUR),sprintf("%d",year),font=2)
  # write the average ratio
  text(median(idx),average+0.05,sprintf("%.2f",average),col=2,
		font=2,cex=0.8)
  year=year+1;
}

##-------------------------------------------------------------------
## L1.7.2. Mouse phenom :)
##-------------------------------------------------------------------
## load data
Mice=read.table("http://edu.sablab.net/data/txt/mice.txt",
                header=T,sep="\t")
str(Mice)

## initiate window
x11(10,8)

## plot a factorial data
plot(Mice$Strain,las=2,
     col=rainbow(nlevels(Mice$Strain)),cex.names =0.7)
title("Number of mice from each strain")


## plot a factorial data as pie
pie(summary(Mice$Sex), col=c("pink","lightblue"))
title("Gender composition (f:female, m:male)")


## try to use special command "barplot" as well
## a histogram
hist(Mice$Starting.weight,probability = T,
     main="Histogram and p.d.f. approximation",
     xlab="weight, g")
lines(density(Mice$Starting.weight,width=0.5),lwd=2,col=4)



## (!) a box-plot of the population on the basis of sex
boxplot(Starting.weight~Sex,data=Mice,col=c("pink","lightblue"))
title("Weight by sex (f:female,b m:male)",
      ylab="weight, g",xlab="sex")

##-------------------------------------------------------------------
## L1.7.3. Show all data frame at once
##-------------------------------------------------------------------

plot(Mice)
```

```
plot(Mice[,-(1:3)])

##-----------------------------------------------------------------------
## L1.7.4. 3D visualization and custom functions
##-----------------------------------------------------------------------

## see demo
demo(persp)

## use RGL library
library(rgl)

x=Mice$Starting.weight
y=Mice$Ending.weight
z=Mice$Fat.weight
plot3d(x,y,z)

## make it more beautiful
color = as.integer(Mice$Sex)*2
plot3d(x,y,z,
       col=color,type="s",radius=1,
       xlab="Starting weight",
       ylab="Ending weight",
       zlab="Fat weight")


#>>>>>>>>>>>>>>>>>>>>>>>>>>>
#> please, do Tasks L1.7ab
#>>>>>>>>>>>>>>>>>>>>>>>>>>>

#############################################################################
#######################################################################
# L1.8. DESCRIPTIVE STATISTICS
#######################################################################
## clear memory
rm(list = ls())

## load data
Mice=read.table("http://edu.sablab.net/data/txt/mice.txt",
                header=T,sep="\t")
str(Mice)

##-----------------------------------------------------------------------
## L1.8.1. Measures of center
##-----------------------------------------------------------------------
summary(Mice)

## mean and median
mean(Mice$Ending.weight)
median(Mice$Ending.weight)
## for mode you should add a library:
library(modeest)
mlv(Mice$Ending.weight, method = "shorth")$M


## mean and median if NA values present: add na.rm=T
```

```
mn = mean(Mice$Bleeding.time, na.rm=T)
md = median(Mice$Bleeding.time, na.rm=T)
mo = mlv(Mice$Bleeding.time, method = "shorth", na.rm=T)$M

## let us plot them
x11()
plot(density(Mice$Bleeding.time, na.rm=T), xlim=c(0,200),lwd=2,
    main="Bleeding time")
abline(v = mn,col="red")
abline(v = md,col="blue")
abline(v = mo ,col="cyan")
legend(x="topright",c("mean","median","mode"),
        col=c("red","blue","cyan"),pch=19)

prop.f = sum(Mice$Sex=="f")/nrow(Mice)

##-------------------------------------------------------------------
## L1.8.2. Measures of variation
##-------------------------------------------------------------------

## quantiles, percentiles and quartiles
quantile(Mice$Bleeding.time,prob=c(0.25,0.5,0.75),na.rm=T)

## standard deviation and variance
sd(Mice$Bleeding.time, na.rm=T)
var(Mice$Bleeding.time, na.rm=T)

## stable measure of variation - MAD
mad(Mice$Bleeding.time, na.rm=T)
mad(Mice$Bleeding.time, constant = 1, na.rm=T)

##-------------------------------------------------------------------
## L1.8.3. Measures of dependency
##-------------------------------------------------------------------

## covariation
cov(Mice$Starting.weight,Mice$Ending.weight)

## correlation
cor(Mice$Starting.weight,Mice$Ending.weight)

## coefficient of determination, R2
cor(Mice$Starting.weight,Mice$Ending.weight)^2

## kendal correlation
cor(Mice$Starting.weight,Mice$Ending.weight,method="kendal")
## spearman correlation
cor(Mice$Starting.weight,Mice$Ending.weight,method="spearman")

#>>>>>>>>>>>>>>>>>>>>>>>>>
#> please, do Task L1.8
#>>>>>>>>>>>>>>>>>>>>>>>>>

####################################################################
# L1.9. DETECTION OF OUTLIERS
####################################################################
```

```r
## clear memory
rm(list = ls())

## load data

#Mice=read.table("d:/data/abs2016/data.txt/mice.txt",header=T,sep="\t")

Mice=read.table("http://edu.sablab.net/data/txt/mice.txt",
        header=T,sep="\t")
str(Mice)

x = Mice$Weight.change
##------------------------------------------------------------------
## L1.9.1. z-score
##------------------------------------------------------------------
x11()
z = scale(x)
plot(z,pch=19,col=4,main="z")

plot(abs(z),pch=19,col=4,main="|z|")
abline(h=3,col=2)

Mice$Weight.change[abs(z)>3]

##------------------------------------------------------------------
## L1.9.2. Iglewicz-Hoaglin method
##------------------------------------------------------------------
x11()
zih=(x-median(x))/mad(x)
#zih=(x-median(x))/median(abs(x-median(x)))
plot(abs(zih) ,pch=19,col=4,
    main="|z| by Iglewicz-Hoaglin")
abline(h=3.5,col=2)
##------------------------------------------------------------------
## L1.9.3. Grubb's method
##------------------------------------------------------------------
library(outliers)
x1=x
while(grubbs.test(x1)$p.value < 0.05){
  print(outlier(x1))
  x1[x1==outlier(x1)]=NA
}
x11()
plot(abs(x-mean(x1,na.rm=T))/sd(x1,na.rm=T),pch=19,col="red",
     main="Outliers by Grubb's Test")
points(abs(x1-mean(x1,na.rm=T))/sd(x1,na.rm=T),pch=19,col="blue")

#>>>>>>>>>>>>>>>>>>>>>>>>>>
#> please, do Task L1.9
#>>>>>>>>>>>>>>>>>>>>>>>>>>
```