TMA4267 - Linear Statistical Models
Introduction to R and Rstudio

Mette Langaas

19 January 2017

R is a free software environment for statistical computing and graphics. It runs on a wide variety of UNIX platforms, Windows and MacOS. R can be downloaded from `http://www.r-project.org/`.

We recommend that you run R within the R-studio system. R-studio can be downloaded from `http://rstudio.org/`

Notice: you need to download *both R and Rstudio.*

Part A: Using Rstudio - what are the different windows? Start Rstudio. Then you (probably) have the following four windows.

- Source (the script window),
- Console (where the R commands are executed),
- Workspace/History (the objects that you have in your workspace, and the commands you have executed),
- Files/Plots/Packages/Help.

We go through the first 3 windows (and Help in the 4th window) and talk about the relationship between Rstudio and R, why we should write in the source (script) window, and what is available in the Workspace/History window.

We start making an Rintro.R file in the source window (File- Save as), and go to the working directory there you want to put your TMA4267 R stuff (Session- set working directory - choose working directory).

Just to be clear on this: when you are finished you may choose Rstudio-Quit Rstudio, and answer if you want your workspace and your script to be saved. Alternatively, you may write `q()` in the command window to quit R, if you answer yes to "Save workspace image" all the objects you have created are found in a `.RData` file. Remember to save the script.

Part B: Trying out R-commands - an oversized calculator?

Write and execute the following commands from the source window.

```
2+3
2*6
3*10^4-3*5^2
sqrt(9)
```

```
log(3,base=10)
?log
log
exp(34)
gamma(3)
factorial(2)
choose(10,4)
1:4
c(1,2,3,4)
sum(1:5)
heights = c(192,185,174,195,173)
shoes = c(46,43,40,45,40)
ratio = heights/shoes
ratio
```

Observe: we can both use "=" and "< −" for assigning content to an object. The function `c` combines values into a vector (concatenate).

Part C: Importing and exporting

For our exercises source code will be available for all problems involving R. You may either download the code to your computer and open the code in the source window (what I recommend) or run the code directly from the course www-page by writing in the console window

```
source("https://www.math.ntnu.no/emner/TMA4267/2017v/RintroPartB.R",echo=TRUE)
```

If you want to export the commands you have written *together* with the output `sink` might be the best solution. For exercise 1, problem 1 (getting to know matrices in R), if you want to save commands and output to a text-file called Ex1P1res.txt (in your working directory)

```
sink("Ex1P1res.txt")
source("https://www.math.ntnu.no/emner/TMA4267/2017v/RecEx11.r",echo=TRUE)
sink()
```

NB: the file Ex1P1res.txt will be a text file, for you to read. (Remark: If the code involves plotting, you may add a command so R pauses whenever a new plot is made, `par(ask=TRUE)`.)

Reading and writing data into R may be a bit tricky if the format of the data is not defined exactly, and we postpone that to when we need to read data for the first time.

Exporting plots:

```
ds = rnorm(100)
hist(ds)
```

should produce a boxplot, that we see in Plots window. You may save it by choosing Export in the Plots window (I suggest pdf or svg). Alternatively, you may write

```
dev.copy2pdf(file="box.pdf")
```

The file will be saved to your working directory.

A third solution `pdf("box.pdf"); boxplot(ds); dev.off()` to make a file named `box.pdf` with the boxplot, then it is possible to save many plots together in one pdf-file.

Part D: Functions and libraries/packages

We use the function `rnorm` to sample independent data from the univariate normal distribution.

```
rnorm #lists the function code
?rnorm
rnorm() #gives error
rnorm(n=100,mean=0,sd=1)
lm # more to see, will be what we use to perform linear regression
```

See Part E where a simple function is programmed.

In Part 1 of the course, we use the library named ellipse. This can be installed by use of the Packages tab in the Files/Plots/... window.

Alternatively, by writing the following in the source or command window:

```
install.packages("ellipse")
# choose Norway for downloading
library(ellipse) # to make library available in the current session
```

If you are not in command of you laptop, create a folder names Rlibs shomewhere to put R libraries in and use `install.packages("ellipse",lib="M://Rlibs/")`.

Part E: Plotting distributions and writing a simple function

The t-distribution can be plotted and critical values marked.

```
qt(0.005,8) # need at least this far - ok go to -4,4
x <- seq(-4,4,length=500)
y <- dt(x,8)
plot(x,y,type="l", main="T-distribution with 8 df")
abline(v=qt(0.025,8))
abline(v=qt(0.975,8))
```

Alternatively, use `curve`:

```
curve(dt(x,df=8),-4,4,n=500)
```

Writing a simple Z-test as a function To not make it too complicated let us assume that we are testing $\mu = 0$ in one sample and that we use a two-sided test with significance level 0.05. Then we need a function with the data and given standard deviation as input.

```
myz.test <- function(x,sd)
{
n <- length(x)
xbar <- mean(x)
zobs <- xbar/(sd/sqrt(n))
print(paste("Zobs",zobs))
```

```
pval<- 2*pnorm(abs(zobs),lower.tail=FALSE)
print(paste("P-value",pval))
return(list("statistic"=zobs,"p.value"=pval))
}

testds <- rnorm(100,mean=0.5,sd=6)
myz.test(testds,sd=6)
```

If you want to run this function for many simulated data sets and put the results from `myz.test` into a matrix you may use the following simple for-loop:

```
mu=0
sigma=4
n=10 # size of each data set
B=1000 # number of data sets to simulate
results <- matrix(NA,ncol=2,nrow=B)
for (i in 1:B)
{
  testX <- rnorm(n,mean=mu,sd=sigma)
  thisresult <- myz.test(testX,sigma)
  results[i,]<- c(thisresult$statistic,thisresult$p.value)
}
results
hist(results[,2])
```

What is done here? What does the histogram display?

This is just as an illustration of how to make a for-loop and collect results - and see a list, and in R (sadly) for-loops are inefficient solutions and should be avoided.

Part F: More on vectors and matrices

Useful for vectors:

```
x = 1:5
x = seq(1,5,length=5)
x = c(1,2,3,4,5)
x[2] = 10
x[3:4] = 0
x[-2] = 1
x[c(1,4)] = 4
x[x>4] = 10
y = log(x)
z = exp(y)
z = z + y
y = x * y
z = y / x
a = t(x)%*%y        # t(): transpose
min(x)
max(x)
sum(x)
```

```
mean(x)
var(x)
length(x)
sort(x)
order(x)
sort(x) == x[order(x)]
```

For matrices:

```
A = matrix(1:6, nrow = 3, ncol = 2)
A
#     [,1] [,2]
#[1,]    1    4
#[2,]    2    5
#[3,]    3    6
B = matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE )
B
#     [,1] [,2] [,3]
#[1,]    1    2    3
#[2,]    4    5    6
A%*%B                       # matrix multiplication
#    [,1] [,2] [,3]
#[1,]   17   22   27
#[2,]   22   29   36
#[3,]   27   36   45
A*t(B)
#     [,1] [,2]
#[1,]    1   16
#[2,]    4   25
#[3,]    9   36
```

The functions `cbind` and `rbind` can also be used to create matrices:

```
x1 = 1:3
x2 = c(7,6,6)
x3 = c(12,19,21)
A = cbind(x1,x2,x3)    # Bind vectors x1, x2, and x3 into a matrix.
                        # Treats each as a column.
A = rbind(x1,x2,x3)    # Bind vectors x1, x2, and x3 into a matrix.
                        # Treats each as a row.
```

Other matrix commands are

```
dim(A)             # get the dimensions of a matrix
nrow(A)            # number of rows
ncol(A)            # number of columns
apply(A,1,sum)     # apply the sum function to the rows of A
apply(A,2,sum)     # apply the sum function to the columns of A
sum(diag(A))       # trace of A
A = diag(1:3)
solve(A)           # inverse of A
det(A)             # determinant of A
```