Introduction to Statistical Computing with R (Ch1)

22 January 2019

Syllabus, learning outcomes and assessment

- Go to the syllabus that includes learning outcomes of the course **Link to Syllabus**
- Examination
    1. **Project work** at the end of the course. Solutions should be presented in terms of a report. Individually and independent. The front page should make clear who the student is, with contact information. All clearly documented R-code should be provided in supplementary materials. Graded by Swedish scale U-VG.
    2. **Home exercises** based on the tutorials. During the tutorials you will work with exercises that you are supposed to summarise and hand-in due to deadlines. After the deadlines we will publish the answers to the exercises on learn. If you hand-in all of the exercises in time, except for one, then you have the possibility to be counted up to pass the home exercises.

About R

- R is a statistical package which includes a number of built-in statistical and computational functions
- R is a free open source software
  - Download it from `http://cran.r-project.org`
  - It runs on Linux, OS X and Windows
- There is a vast library of contributed open source packages including functions (13711 now 2019, 6195 last year)
  - These packages are written by statisticians, researcher and scientist around the world
  - You install packages by command *install.packages(<name of package>)* or by follow click-menus in the R interface
  - You find lists and information about all available packages on `http://cran.r-project.org`
- R is also a programming language (a bit similar syntax as C & C++)

Pros with R

- R is the most popular statistical package among statistical researchers
  - Because of this many of the most recent methods and estimators are available in R through contributed packages
  - Methods in R often appears years before they are implemented in commercial packages like SAS and STATA
- R is also used extensively by companies like Pfizer, Google and Oracle and it is an extremely popular language in the areas of analytics, data mining and data science (see recent poll by Kdnuggets Link to Poll)
- In Sweden companies and organisations have been lagging behind a bit but R is used by e.g. Statisticon (consultant company), by Pensionsmyndigheten and at several departments within the academic community
- R is a programming language which makes it simple to make your own functions, loops etc. Much easier than in e.g. SAS

Cons with R

- It takes a while to learn compare to menu based statistical packages like SPSS, Minitab and gretl (if you have no programming background)
- The base functions in R are actually written in FORTRAN, C/C++ and it takes some time to compile
  - Looping is slow compared to C and FORTAN
- Poor memory management: loads data and everything in RAM
  - However later into the course we will work with a Big Data example and you will see that this problem can be resolved
  - With help of some contributed packages we can save data on the hard drive instead

Exercise: Install and open R

- Download R from `http://cran.r-project.org`
- If you have not brought a laptop join someone who has
- Open R
  - R comes with a default interface including an outcome window and a text editor
  - The text editor is used to write commands in (R-code)
  - Other text editors and interfaces can be used, e.g. Tinn-R, R-studio and Emacs

Always use a R-script!

- *file* $- >$ *New script* (Windows), or *file* $- >$ *New document* (Mac)
- Write all your commands in the R-script
- Select a part of the scrip (or select the whole script) and run it by
  - **ctrl-r** or **cmd-enter** to run the selected part of the script
- Save your R-scrip under menu *file* or by **ctrl-s** or **cmd-s**

Exercise: Install a contributed package

- Install the package *car*
- Load the package in R with the command *library(car)*
- Get basic information about the package by *?car*
- Get more information by *help(package="car")*

R Objects

- Every symbol in R is an object and every object belongs to a class
- Some classes are: numeric, character, logical, data.frame, matrix, and list
- You can name an object by assigning, e.g. the numerical value 5 to the symbol 'a':

```
a<-5
```

  - The traditional assignment symbol is `'<-'`
  - It also works in most cases with `'='`

Arithmetic operators

- $+$ additon
- $-$ subtraction
- * muktiplication
- $/$ division
- $\wedge$ or ** exponentiation

Examples Arithmetic operators

```
> 1+1
[1] 2
> 2-1
[1] 1
> 10*10
[1] 100
> 10/10
[1] 1
> 10^2
[1] 100
#Assign 10^2 to symbol 'a'
> a<-10^2
> a
[1] 100
```

Some R functions

- abs(x): absolut value
- sqrt(x): square root
- log(x): natural logarithm
- exp(x): exponential e^x
- mean(x): sample average
- var(x): sample variance
- rnorm(n,mean=0,sd=1): generates sample of n from N(0,1)
- round(x, digits=n): round x of to n digits
- summary(x): generic function that works on several R objects
- length(x): number of elements in a vector x (measures size on other objects too)

More functions here
`http://www.statmethods.net/management/functions.html`

Data Items

- Number data: - integer (no decimal)
  - numeric (with decimal)
- Text data: - character
  - factor (example: high , low)

Factor are character data, but coded as numeric mode. Each number is associated with a given string, the so-called levels.

#Convert data from factor(F) to character(C)

C = as.charater(F)

#Convert data from character(C) to factor(F)

F = as.factor(C)

Structure of Data Items

- Data are constructed on various of ways
- Use str() command to examine structure of objects
- Use class() command to check the type of object

- Vectors: - One dimensional object
  - Smallest unit

- Matrix: - Two dimensional object
  - Columns must be of the same type
- Array: - More than two dimensional object
  - Columns must be same length
  - Matrix in more dimensions

Structure of Data Items

. Data frames: - Two dimensional object (tabell form)
- Mixed items, not of the same type,
- Have same length of column
- Rectangular shape pads out with NA

. Lists - Series of items bundled together
- seperately named vectors
- Not same type and not same length

Vectors and Variables

```
 n<-1000
# This generates a vector of length 1000 x<-
rnorm(n,mean=1,sd=sqrt(2))
x
# We can also enter data into a vector by keyboard,
# using the 'combine' function
z<-c(3,5,6,1)
z
```

# We can also enter data into a vector by the using seq(start, end,intervall) keyboard, # using the 'combine' function
s<-seq(1,10,by=2)
s

Arithmetic element-wise operations on Vectors

```
# We can also enter data into a vector by keyboard,
# using the 'combine' function
z<-c(3,5,6,1)
z/2
z+1
z*2
z^2
# Operation with vector of same length
y<-c(1,5,2,1)
z/y
z*y
```

Example drawing sample from $N(1, 2)$ and computing summary statistics with R functions

```
> n<-1000
> x<-rnorm(n,mean=1,sd=sqrt(2))
> mean(x)
[1] 1.05555
> var(x)
[1] 2.039799
> round(mean(x),digits=2)
[1] 1.06
> summary(x)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.79200  0.09312  1.04600  1.05600  2.01900  6.41900
```

Non-numerical Vectors

```
# Character vector
Names <- c("John", "Danie", 'Mary',"Oskar")
#Logical vector
male<-c(TRUE,TRUE,FALSE,TRUE)
```

Logical operators

- '$==$'': equal to; '$! =$': not equal to
- '$<$': less than; '$>$': greater than
- '$<=$': less or equal; '$>=$': greater or equal
- '$-$': not
- '&': and; '$|$': or
- '!' negation of a logical object

Indexing vector elements

```
# Selecting element 1 in vector y
y[1]
# selecting the two first elements
y[c(1,2)]
#or
y[1:2]; y[-c(3,4)]
#select all elements equal to 1
y[y==1]
#less or equal to 2
y[y<=2]
#Select the names of men
Names[male]
#names of women
Names[!male]
#With name Danie or Mary
Names[Names=="Danie" | Names=="Mary"]
```

Exercise

Select the values in x that are smaller than 0

```
# set.seed just makes sure we are all drawing the same x
set.seed(7)
x<-rnorm(10)
```

User defined functions

You assign a function like this:

```
Sd<-function(z){
sqrt(var(z))
}
```

this function is called "Sd", has a single argument z (should be a vector) and computes the sample standard deviation.

Example

```
Sd<-function(z){
sqrt(var(z))
}
#function call
Sd(z=x)
#compare to corresponding R function
sd(x)
```

Example: Function that computes average for selected group

```
# z is numeric
# group is a logical vector that indicates
# the selected group
Avg.group<-function(z,group){
mean(z[group])
}
# some data
z<-c(3,5,6,1)
#for
male<-c(TRUE,TRUE,FALSE,TRUE)
#function call
Avg.group(z,group=male)
```

Example: Function that computes average for selected group

```
# z is numeric
# group is a logical vector that indicates
# the selected group
Avg.group<-function(z,group){
mean(z[group])
}
# some data
z<-c(3,5,6,1)
#for
male<-c(TRUE,TRUE,FALSE,TRUE)
#function call
Avg.group(z,group=male)
```

Conditionals (Ch 8.3.1)

if/else conditioning

```
Avg_var.group<-function(z,group,Mean=TRUE){
if (Mean) {
mean(z[group])
}
else {
var(z[group])
}
}
Avg_var.group(z,group=male,Mean=FALSE)

#or
Avg_var.group<-function(z,group,Mean=TRUE){
ifelse(Mean,mean(z[group]),var(z[group]))
}
Avg_var.group(z,group=male,Mean=FALSE)
```

For looping (Iteration; Ch 8.3.2)

Simple hello world example

```
for (i in c(1,2,3)){
print(i)
print("Hello world!")
}

#or
n<-3
for (i in 1:n){
print(i)
print("Hello world!")
}
```

Example: Draw from N(0,1) 10 times and compute and store averages

```
n<-1000
B<-10
store<-NULL
for (i in 1:B){
x<-rnorm(n)
store<-c(store,mean(x))
}
store

#or
store<-vector(mode = "numeric", length = 10)
for (i in 1:B){
x<-rnorm(n)
store[i]<-mean(x)
}
store
```

While looping (Iteration; Ch 8.3.2)

Simple hello world example

```
n<-3
i<-1
while (i<=n){
print(i)
print("Hello world!")
i<-i+1
}
```

Exercise

Write a for or while loop of the *Hello world* example but the last iteration should print *See you world!* instead of *Hello world!*

Next lecture

This is enough as an introduction

Next lecture: Data manipulation (mainly Ch2)