Data analysis with R

Lecture 1

Introduction to R

- R is a functional language that uses many of the symbols as the widely used general purpose languages C, C++ and Java.
- R has a language core that uses standard forms of algebra and allows the calculations, functions are then built on top of the core, allowing a limitness extension of the language.
- Most of R-code can be run also in the commercial software S-PLUS, without modifications.

Introduction to R (2)

- All R entities, including functions, data structures etc. are R-objects and are operated on as data.
  - You can type *ls()* or *objects()* to see all the R-objects currently in the workspace.
- When quitting R, you automatically lose every object you have created to the workspace.
  - However you can save your workspace as an image (*.Rdata), which you can load next time you use R.

Working environment

- Working directory is the directory where R will read and write files by default.
- You can access it by *getwd()* and change it by *setwd()*
- Object is a data structure or function that R recognizes
  - Functions, as well as data, exist as \objects"
  - Note also, eg, formula objects, expression objects, etc.

Working environment (2)

- Workspace is the current user's "database"
- Everything that user has created during the current R-session, is stored in the workspace.
- Image files are used to store R objects, that is workspace contents as *.Rdata. You can do this by *save.image()*

Online help

- R has excellent help pages online. You can access them in several ways.
- Type in *help(help)* to get information about the help features.
- To get help on a specific function just type for example *help(plot)*
- If you're in a search of a function which does a certain thing you can search it by *help.search()* Eg. *help.search("sort")* gives a list of functions which have "sort" in their alias or title

Overview of R

- The > at the start of the line is the command prompt. User commands are typed following this.

```
> 5+6
[1] 11
> 12/3
[1] 4
> 10 < 1
[1] FALSE
```

- The [1] says “first element will follow”
- So as you can see, results are printed together with the source code.

R-syntax

- Command separator: End of line or ;
  - Eg. 6*5; print(10-4)
- Upper case is different than lower case
  - A ? a
- Assignment symbol is <-
  - Eg. X <- 9
- Comments are introduced with # sign

Utility functions

- Functions that act on the contents of the workspace

```
ls() # List contents of workspace
rm(x, y, z) # Remove x, y, and z from workspace
rm(list=c("x", "y", "z")) # Alternative to rm(x, y, z)
rm(list=ls()) # Remove contents of workspace
str(airquality)
```

- See examples.

NAs

- Missing values are stored in R with NA. This is different from many other statistical software, so important to remember
- Eg. *X <- c(2,4,5,NA,4,6,NA)*
  - [1] 2 4 5 NA 4 6 NA
- Missing values play an important role in many statistical functions
  - That's why many of them have options on how to handle the NAs. (*na.action*)

Comparison operators:

- <
- >
- <=
- >=
- == (equality)
- != (inequality)

Vectors in R

- Vectors in R can have mode ”logical”, ”character”, ”numeric” or ”list”. Examples of these:
  - *c(6,4,2,4,0)* [numeric]
  - *c(T,F,T,F,T,F) [logical]*

    [1] TRUE FALSE TRUE FALSE TRUE FALSE
  - *c(”Helsinki”,”Turku”,”Salo”) [character]*
  - *list(2,”Helsinki”) [list]*

    [[1]]

    [1] 2

    [[2]]

    [1] "Helsinki"

Factors

- A factor is stored with values 1,2,3...k
- The levels are character strings. Example:
  - Sex <- c(rep(”Male”,50),rep(”Female”,70))
    - Then change to factor typing
  - Sex <- factor(Sex); levels(Sex)
    - [1] "Female" "Male"
- Many functions require the use of factors when applicable. So it’s important to know how to do it.
- Notice that, by default, the levels are taken in alphanumeric order.

Changing classes

- When doing data analysis with R, you have to have the variables in right class.
- You can check, whether a variable is from certain class by typing for example *is.factor(sex)*
- To change the class you can type *as.factor(sex)*
- Same kind of functions apply for other classes as well. (eg. As.numeric, is.data.frame, etc.)

Data frames

- Data frames are a very import part of R modeling and R-graphics.
- Data frames offer a tidy way to supply data to modeling functions.
- Data frames are essential practically always when we do data analysis with R.
- Data frames are generalization of matrix-objects.
- Every column can have different modes.

Data frames; properties

- Data frames have row names
  - Assessed by row.names(Cars93.summary)
- And column names
  - Assessed by colnames(Cars93.summary)
- To assess a certain column of a data frame we can use $-sign
  - Eg. Cars93.summary$Max.passengers
- There are other ways to do the same thing as well.
  - Cars93.summary[,4]
  - Cars93.summary[,"abbrev"]
  - Cars93.summary[[4]]

Data frames; properties (2)

- Other useful functions with data frames
  - names() #names of columns
  - dim() #Dimensions
  - summary() #Numeric summary details
  - str() #More technical summary
  - class() #Class of data frame column
  - attach() #Avoid repeated references
- See examples

Length & subsets

- All R objects have a length, which can also be 0.
- Eg. With vectors the length is the number of values in the vector, with data-frames the length is rows x columns. Length is very useful function with user created function and loops.
- Subsetting can be done in several ways.
  - Easiest way is to use *subset(), eg. subset(x, x<10)*
  - *x[c(1,3:4)]*
  - *x[-c(2,5)]*

subsets()

- We can also subset datasets based on some condition(s)
- For example take only observations, which are above or below some pre-determined limit
  - Xsub <- x[x>10]
  - High <- airquality[airquality$Temp > 60,]
- NOTE! With data-frames we first refer to rows and then after the ”,” to columns, as shown above.

read.table - function

- Most common way to read in data to R is to use *read.table* –function.
- Usually used with txt-files
- Basic syntax:
  - *Read.table(file,header=FALSE, sep="\", ...)*
    - Has many options that the users can modify, but the only obligatory option is file.
  - Special functions have been created based on read.table to read in different kinds of data
    - Eg. *read.csv*, *read.xls*, *read.delim*, *read.xport*

scan - function

- scan – function is also for reading in data to R
- It is a lot more flexible tool than eg. *read.table*
  - The data doesn't have to be in a table form, for example
- Usually, though, *read.table* is enough and we should prefer it in table-like datasets for it's convinience.

  scan(file, ...other options)

AND & OR -operators

- The basic AND & OR –operators are typed with symbols as **&**(=AND) and | (=OR).
- However, sometimes these are not enough and we need also Logical AND, and Logical OR. These are typed as symbols as **&&** and ||.
  - See examples.

R-editors

- There are several text-editors that we can use to make our programming easier
- For example
  - Tinn-R, Emacs, WinEdt, Gnu
- R comes with it's own editor: Rgui, which is the same as Notepad. Use of other editors is always beneficial for the programmer
- Later during the course, we are going to see the use of Tinn-R.