Preparation

Matthias Sand, Jan-Philipp Kolb and Stefan Zins

11 Januar 2016

Introduction

This document can be used for the preparation for the GRADE- workshop Sampling and Estimation at the university of Frankfurt. Hints for further reading are included at the end of each section.

Why use R?

There are several arguments for the use of R as a tool for sampling and estimation:

- Rapid implementation of new (scientific) developments
- Quick development of new tools that fit the user's demand
- Over 5,000 packages contributed by users available on CRAN
- Open Source - You can create your own objects, functions and packages
- Reproducibility

More arguments for the usage of R can be found here or here.

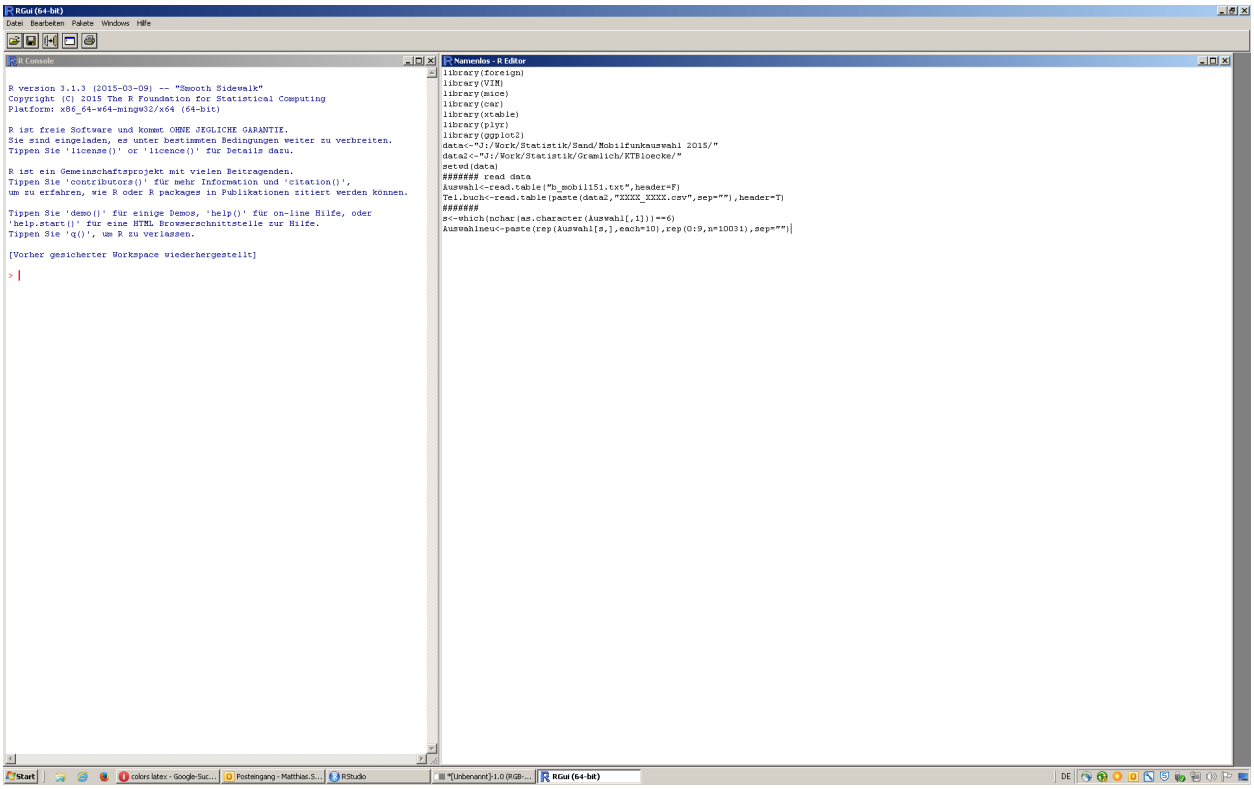
Get R

R can be installed on Windows and Linux plattforms as well as on Macs. If you have not done already please download R from here.

The installation process should be straightforward. If you have problems you can read an introduction or watch an intro on youtube.

Rstudio

The basic R looks like this:



Most R-users prefer the graphical user interface (GUI)

Several GUI's are available. In this course we will use the rstudio GUI which can be downloaded here.

- How to install Rstudio (youtube | dummies)

First operations

If you work with R you should work with scripts and it is important to have everything clear. To reuse scripts it is necessary to comment the code with hashes:

```
# Comments
```

Create new variables with the assignment operator `<-`:

```
x <- 1 # numeric
y <- "a" # string
z <- T # logical
```

The following line creates a vector with ten standard-normal-distributed values.

```
x <- rnorm(10,0,1)
```

`rnorm` is a function which takes several arguments. More information on assignments can be found here.

Functions

```
mean(x)
```

```
## [1] -0.04509283
```

calculates the mean of variable x

More basic commands:

```
length(x)
```

```
## [1] 10
```

```
max(x)
```

```
## [1] 1.695672
```

```
min(x)
```

```
## [1] -1.68667
```

```
sd(x)
```

```
## [1] 1.070286
```

```
var(x)
```

```
## [1] 1.145512
```

```
median(x)
```

```
## [1] -0.07816721
```

Getting help

A uncountable number of introductions is available. The manuals on CRAN are comprehensive.

- Introduction to R
- Thomas Girke - Programming in R
- A collection of tutorial videos can be found here

For more specific questions and solutions for error messages it is useful to use a search engine. Alternatively forums like stackoverflow can be used.

If you have problems to find the commands use a reference card

A basic help is always embedded in R. Get the help page for a command:

```
help.start()

help(mean)

  # if you know already the function name:
?mean
```

Often you can get examples like the following one for linear regression.

```
example(lm)
```

Draw random numbers:

In the following three different functions are used to draw random numbers:

```
  # Uniform Distribution
x1 <- runif(1000)
  # Normal distribution
x2 <- rnorm(1000)
  # Exponential distribution
x3 <- rexp(1000)

rnorm(20,mean=0,sd=1)
```

```
##  [1]  1.91967315 -0.17675239  0.13454451 -0.10132354 -0.02288272
##  [6] -0.25615719  0.22916582  0.63070266 -0.02754458 -1.40669222
## [11] -0.53506626 -0.89264682 -1.02542526 -0.74236442 -1.48102779
## [16] -0.94673200  0.84061541  0.81760944  0.16831448 -1.76951298
```

Installing and Loading Packages

Many functions are already implemented in basic R. For more specific tasks libraries/packages have to be installed. This can be done using the command `install.packages`. After the installation the package must be loaded with the command `library`.

```
install.packages("sampling")
library("sampling")
```

Here is a list of packages which are relevant for the workshop:

- foreign - Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...
- sampling - Survey Sampling
- survey - analysis of complex survey samples

```
install.packages("foreign")
install.packages("lattice")
install.packages("survey")
install.packages("plyr")
```

A list on the most popular R-packages can be found here.

Indexing

Indexing is an important concept, e.g. to select subgroups. In the following the indexing for the different data types are presented.

```
# vector
A1 <- c(1,2,3,4)
A1
```

```
## [1] 1 2 3 4
```

```
A1[1]
```

```
## [1] 1
```

```
A1[4]
```

```
## [1] 4
```

```
A1[1:3]
```

```
## [1] 1 2 3
```

```
A1[-4]
```

```
## [1] 1 2 3
```

```
# dataframe

AA <- 4:1
A2 <- cbind(A1,AA)
A2[1,1]
```

```
## A1
##  1
```

```
A2[2,]
```

```
## A1 AA
##  2  3
```

```
A2[,1]
```

```
## [1] 1 2 3 4
```

```
A2[,1:2]
```

```
##      A1 AA
## [1,]  1  4
## [2,]  2  3
## [3,]  3  2
## [4,]  4  1
```

```r
# array

A3 <- array(1:8,c(2,2,2))
A3
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```r
A3[,,2]
```

```
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```r
# list

A4 <- list(A1,1)
A4
```

```
## [[1]]
## [1] 1 2 3 4
##
## [[2]]
## [1] 1
```

```r
A4[[2]]
```

```
## [1] 1
```

Sequences

```r
# sequence from 1 to 10
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
seq(-2,8,by=1.5)
```

```
## [1] -2.0 -0.5  1.0  2.5  4.0  5.5  7.0
```

```r
a<-seq(3,12,length=12)

b<- seq(to=5,length=12,by=0.2)

d <-1:10
d<- seq(1,10,1)
d <- seq(length=10,from=1,by=1)

# replicate 1 10 times
rep(1,10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

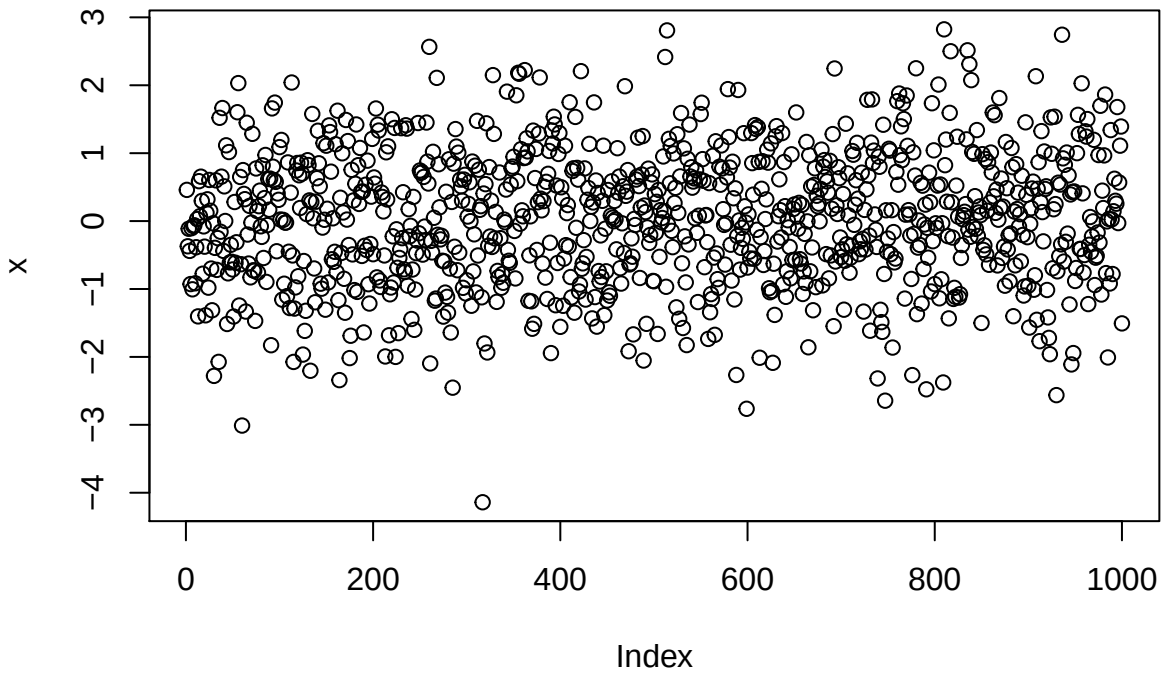
```r
rep("A",10)
```

```
##  [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

Basic Graphics

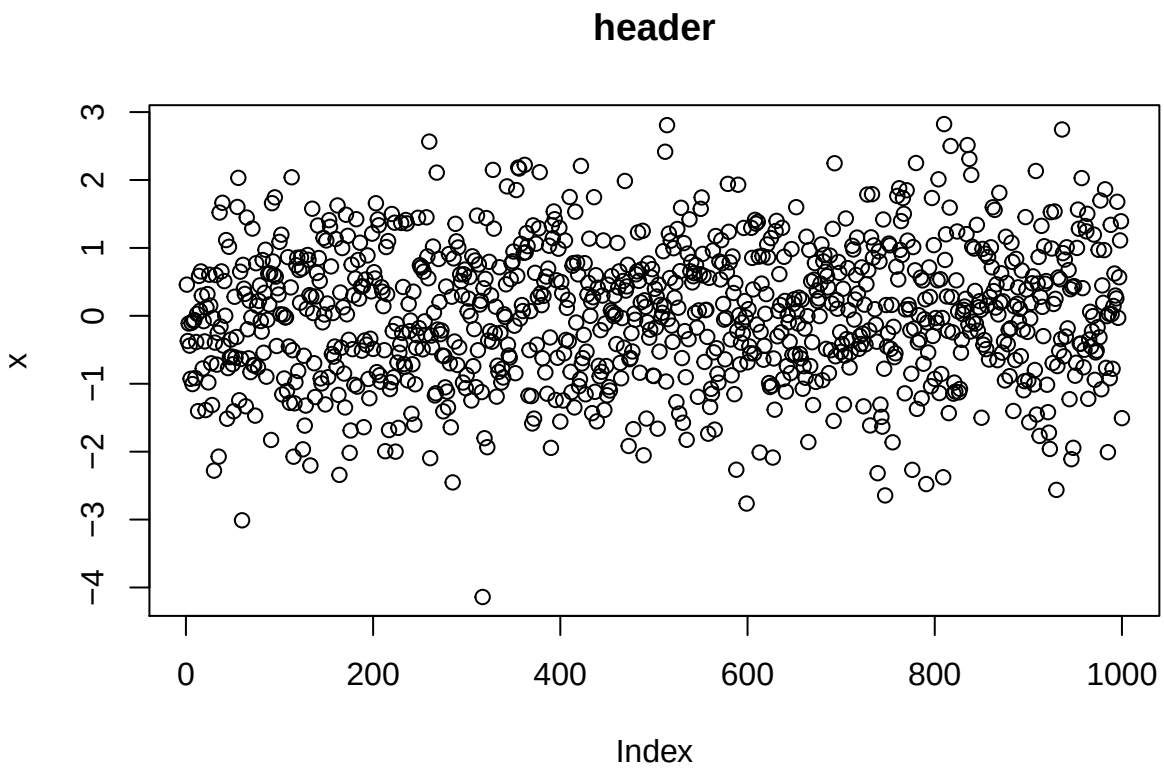
The plot function is the easiest option to get a graphic:

```r
x <- rnorm(1000,0,1)
plot(x)
```



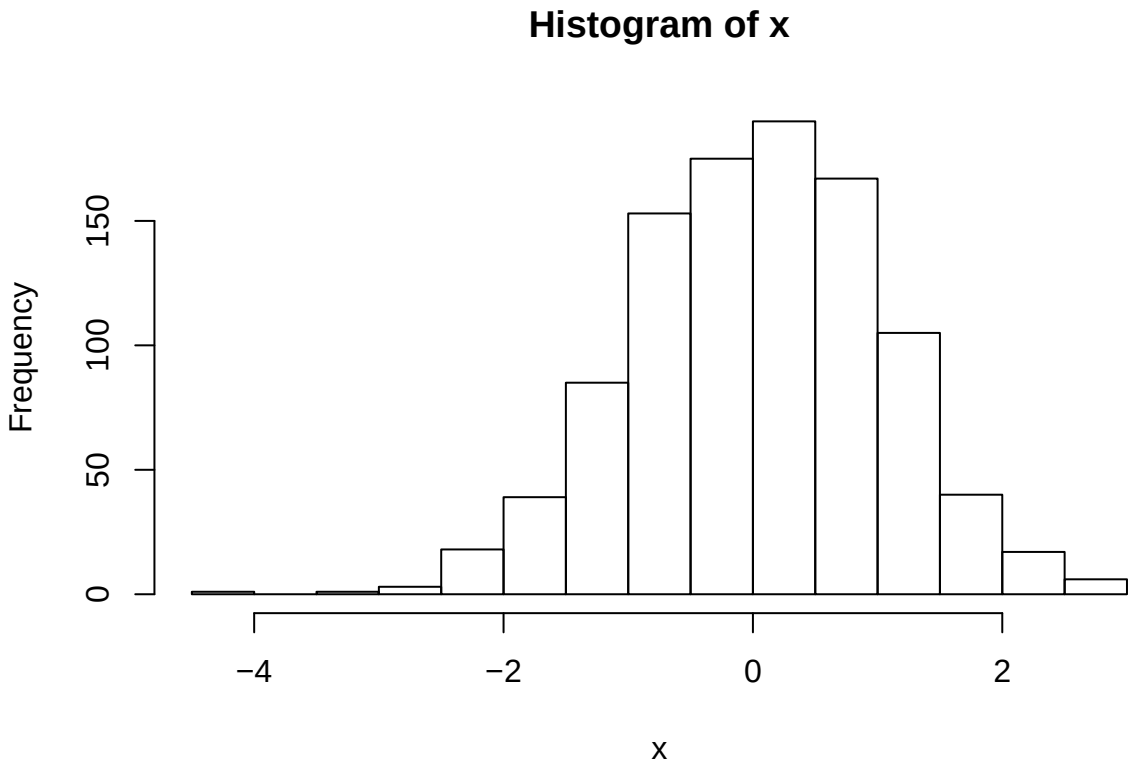
Adding a header:

```
plot(x,main="header")
```



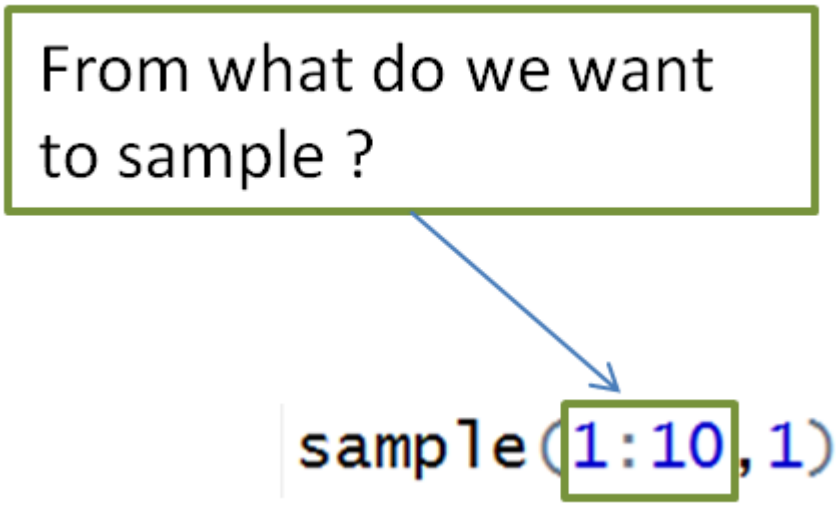
If we want an histogram, we can use the following command:

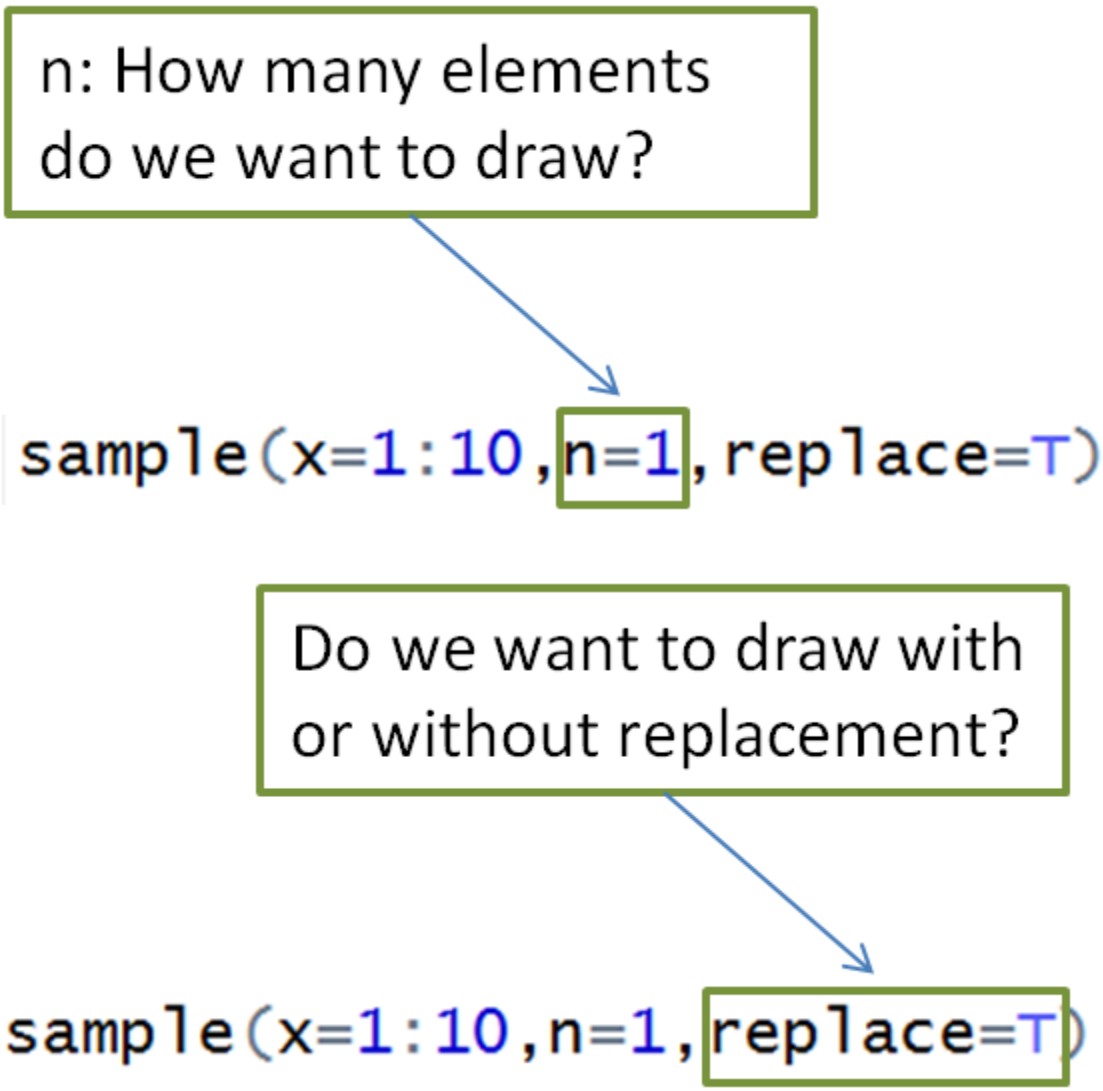
```
hist(x)
```



The sample function

Usage of the command sample






```
sample(x=1:10,1)
```

```
## [1] 1
```

```
sample(x=1:10,1,replace=T)
```

```
## [1] 4
```

Working Directory and Workspace

Declaring a working directory

```
path<-"C:/"

setwd(path)
```

```
getwd()

dir()
```

- It is always useful to define and set your working directory at the beginning of each script
- `getwd()` displays you your current working directory
- `dir()` shows you all objects in a specific directory
- `ls()` lists all objects in your workspace
- `rm()` removes a object from your workspace

```
rm(list = ls())
```

Data Import and Export in R

Some datasets are implemented in R-packages:

```
library("sampling")
data(belgianmunicipalities)
```

```
head(belgianmunicipalities)
```

Commune	INS	Province	Arrondiss	Men04
Aartselaar	11001	1	11	6971
Anvers	11002	1	11	223677
Boechout	11004	1	11	6027
Boom	11005	1	11	7640
Borsbeek	11007	1	11	4948
Brasschaat	11008	1	11	18142
Brecht	11009	1	11	12975
Edegem	11013	1	11	10614

Also foreign datasets can be imported:

```
link <- "https://raw.githubusercontent.com/BernStZi/
SamplingAndEstimation/master/excercise/data/my.pop.csv"

my.pop <- read.csv(link)
head(my.pop)
```

```
##   X id gender education        iq
## 1 1  1   male      high 123.26218
## 2 2  2   male      none  96.19531
## 3 3  3   male       low  94.21088
## 4 4  4 female      high  92.02308
## 5 5  5   male   average 114.18485
## 6 6  6   male   average  67.54705
```

In the following the European Social Survey (ESS) data will be used. The data can be downloaded here. We can import spss data using the command `read.spss` from R-package `foreign`.

```
library(foreign)
ESS7 <- read.spss("ESS7e01.sav",to.data.frame=T)
```

As default the data is imported to a list but it is more convenient to work with `data.frames`. Therefore we have to specify in a further argument, that we want to work with a `data.frame`.

With the package `foreign`it is also possible to import stata-data:

```
library(foreign)
ESS7s <- read.dta("ESS7e01.dta")
```

In the first example a country file and sample data for Sweden is needed.

```
library(foreign)
ESS5_SE <- read.spss("ESS5_SE_SDDF.por",to.data.frame=T)
```

Some Links on import and export of data in R:

- Quick R on importing data
- Quick R on exporting data

Subsetting

Select the first 100 rows of a dataset and assign the information to a new object `bgm`:

```
bgm <- belgianmunicipalities[1:100,]
```

Select only the entries for the first province:

```
bgm1 <- belgianmunicipalities[
  belgianmunicipalities$Province==1,]
```

Select only Communes with a total population bigger than 20000:

```
bgm20 <- belgianmunicipalities[
  belgianmunicipalities$Tot04>20000,]
```

Merging

If you are not shure on the usage of a command, it is always useful to have a look at the help page of the command. E.g. we need to use the command `merge` to combine datasets. There is a section `Example` at the end of each helpfile. There you will find code which can be copy-pasted to the console:

```
authors <- data.frame(
    surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
    nationality = c("US", "Australia", "US", "UK", "Australia"),
    deceased = c("yes", rep("no", 4)))
books <- data.frame(
    name = I(c("Tukey", "Venables", "Tierney",
             "Ripley", "Ripley", "McNeil", "R Core")),
    title = c("Exploratory Data Analysis",
              "Modern Applied Statistics ...",
              "LISP-STAT",
              "Spatial Statistics", "Stochastic Simulation",
              "Interactive Data Analysis",
              "An Introduction to R"),
    other.author = c(NA, "Ripley", NA, NA, NA, NA,
                     "Venables & Smith"))

(m1 <- merge(authors, books, by.x = "surname", by.y = "name"))
```

```
##    surname nationality deceased                        title other.author
## 1   McNeil   Australia       no     Interactive Data Analysis         <NA>
## 2   Ripley          UK       no            Spatial Statistics         <NA>
## 3   Ripley          UK       no         Stochastic Simulation         <NA>
## 4  Tierney          US       no                     LISP-STAT         <NA>
## 5    Tukey          US      yes     Exploratory Data Analysis         <NA>
## 6 Venables   Australia       no Modern Applied Statistics ...       Ripley
```

A first example dataset

The first example dataset is a synthetic example. For more information on the generation of this dataset see the r-code here.

```
link <- "https://raw.githubusercontent.com/BernStZi/
SamplingAndEsimation/master/excercise/data/my.pop.csv"

my.pop <- read.csv(link)
head(my.pop)
```

```
##   X id gender education        iq
## 1 1  1   male      high 123.26218
## 2 2  2   male      none  96.19531
## 3 3  3   male       low  94.21088
## 4 4  4 female      high  92.02308
## 5 5  5   male   average 114.18485
## 6 6  6   male   average  67.54705
```

The dollar sign can also be used to access the columns

```
head(my.pop$gender)
```

```
## [1] male   male   male   female male   male
## Levels: female male
```

With the command `table` we get a frequency table:

```
table(my.pop$gender)
```

```
##
## female   male
##   5125   4875
```

With `prop.table` we get the relative frequencies:

```
tabA <- table(my.pop$gender)
prop.table(tabA)
```

```
##
## female   male
## 0.5125 0.4875
```

Simple Example on Sampling

Summary of the dataset:

```
summary(my.pop)
```

```
##        X               id           gender       education
##  Min.   :    1   Min.   :    1   female:5125   average:2851
##  1st Qu.: 2501   1st Qu.: 2501   male  :4875   high   :2820
##  Median : 5000   Median : 5000                 low    :3588
##  Mean   : 5000   Mean   : 5000                 none   : 741
##  3rd Qu.: 7500   3rd Qu.: 7500
##  Max.   :10000   Max.   :10000
##        iq
##  Min.   : 30.93
##  1st Qu.: 86.50
##  Median :100.08
##  Mean   :100.02
##  3rd Qu.:113.60
##  Max.   :173.26
```

```
prop.table(table(my.pop$gender,my.pop$education))
```

```
##
##          average   high    low   none
##   female  0.1449 0.1465 0.1844 0.0367
##   male    0.1402 0.1355 0.1744 0.0374
```

```
var(my.pop$iq)*(nrow(my.pop)-1)/nrow(my.pop)
```

```
## [1] 406.1684
```

In the following example two simply random samples are drawn, one with replacement and one without replacement:

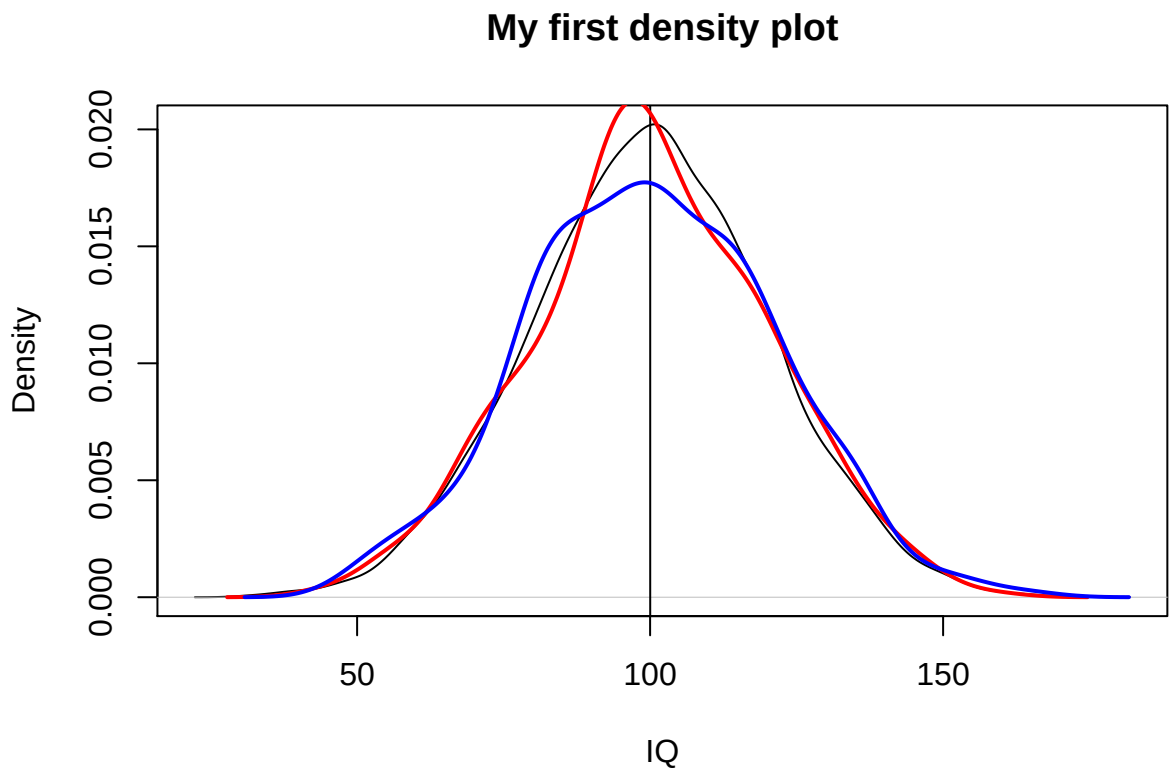
```
s.SRS <- sample(1:nrow(my.pop),500,replace=T)
s.SRSWOR <- sample(1:nrow(my.pop),500,replace=F)
```

```
my.samp.SRS <- my.pop[s.SRS,]
my.samp.SRSWOR <- my.pop[s.SRSWOR,]
summary(my.samp.SRS)
```

```
##        X              id          gender      education         iq
##  Min.   :   1   Min.   :   1   female:267   average:141   Min.   : 43.36
##  1st Qu.:2335   1st Qu.:2335   male  :233   high   :141   1st Qu.: 87.64
##  Median :4966   Median :4966                low    :193   Median : 99.72
##  Mean   :4960   Mean   :4960                none   : 25   Mean   :100.13
##  3rd Qu.:7635   3rd Qu.:7635                               3rd Qu.:114.35
##  Max.   :9963   Max.   :9963                               Max.   :159.11
```

Making graphics to compare the samples:

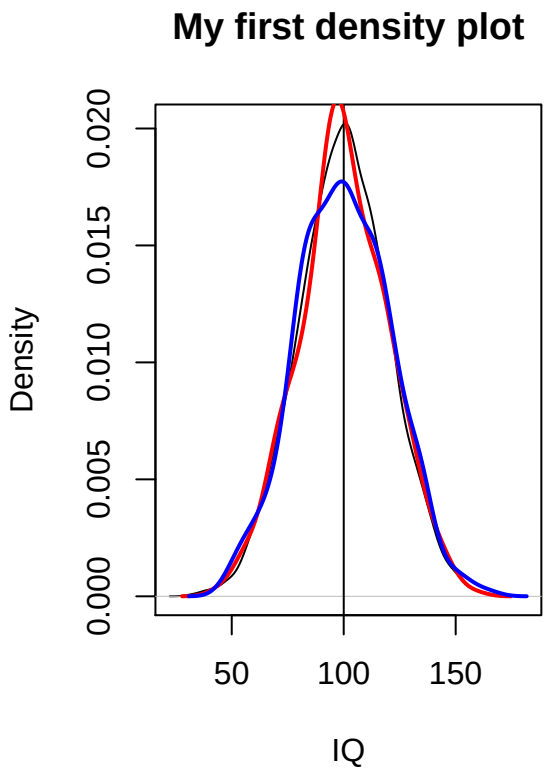
```
plot(density(my.pop$iq),main = "My first density plot"
     , xlab = "IQ")
abline(v=mean(my.pop$iq), col = "black")
lines(density(my.samp.SRS$iq),col = "red",lwd=2)
lines(density(my.samp.SRSWOR$iq),col = "blue",lwd=2)
```



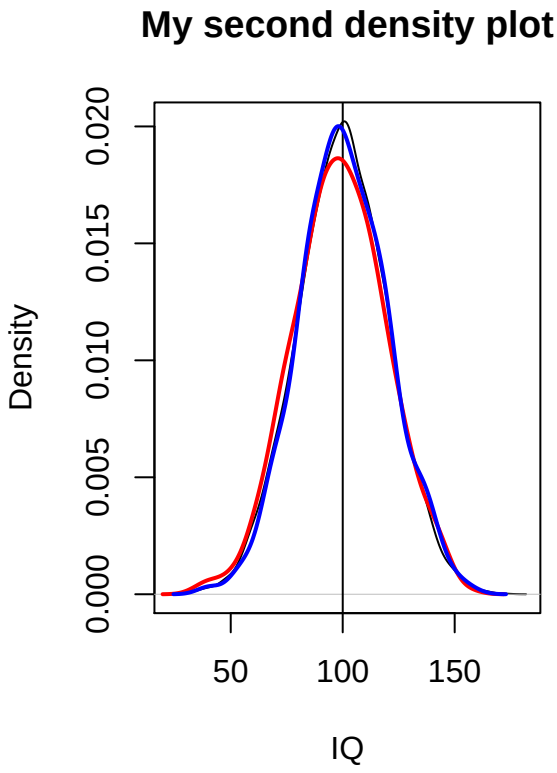
The package **sampling** is very useful to draw samples. An introduction to the package can be found here.

```
library("sampling")
s.SRS1 <- srswr(500,nrow(my.pop))
s.SRSWOR1 <- srswor(500,nrow(my.pop))
my.samp.SRS1 <- rbind(my.pop[s.SRS1!=0,]
                      ,my.pop[s.SRS1>1,])
my.samp.SRSWOR1 <- my.pop[s.SRSWOR1==1,]
```

```
par(mfrow=c(1,2))
plot(density(my.pop$iq),main = "My first density plot"
     , xlab = "IQ")
abline(v=mean(my.pop$iq), col = "black")
lines(density(my.samp.SRS$iq),col = "red",lwd=2)
lines(density(my.samp.SRSWOR$iq),col = "blue",lwd=2)
```




```
par(mfrow=c(1,2))
plot(density(my.pop$iq),main = "My second density plot"
     , xlab = "IQ")
abline(v=mean(my.pop$iq), col = "black")
lines(density(my.samp.SRS1$iq),col = "red",lwd=2)
lines(density(my.samp.SRSWOR1$iq),col = "blue",lwd=2)
```



- should yield same results
- routine may differ because of "starting point"
- Kerns - Introduction to Probability and Statistics Using R