Preparation

Matthias Sand, Jan-Philipp Kolb and Stefan Zins

11 Januar 2016

Why use R?

- 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.

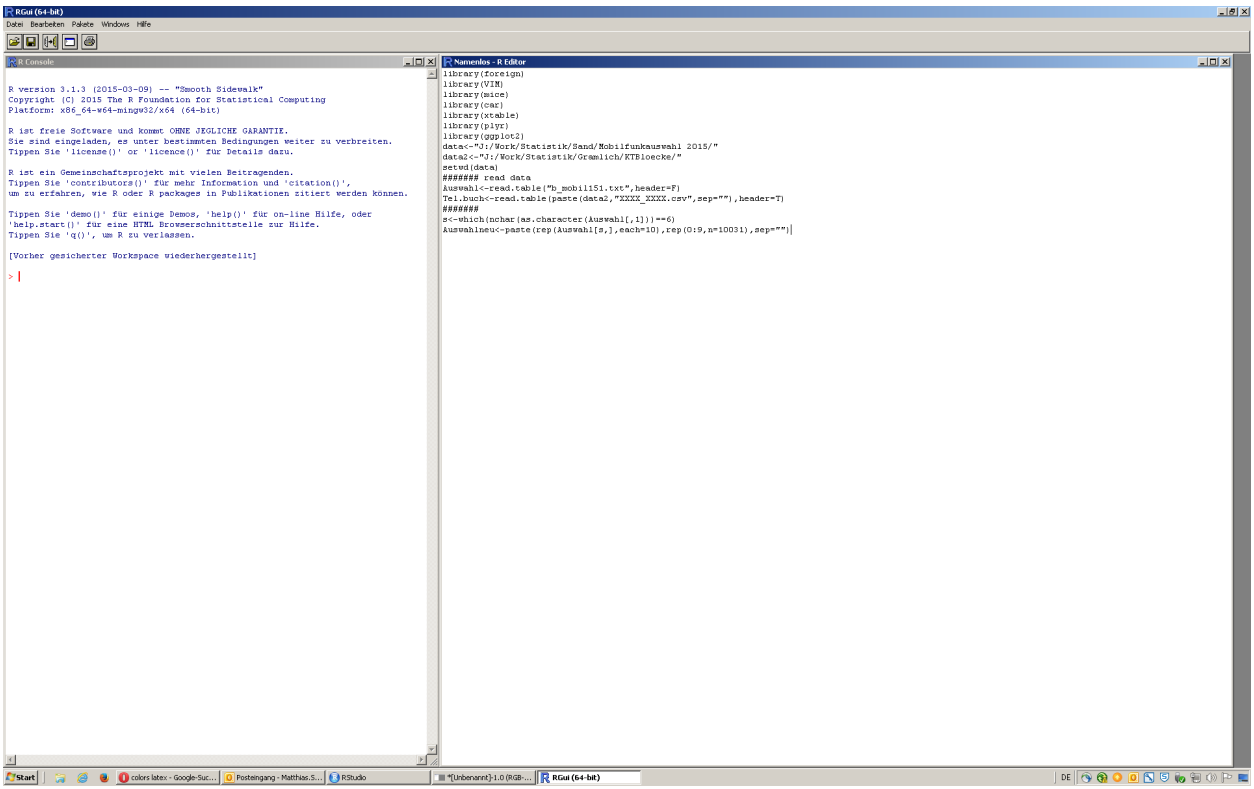
Get R

R can be downloaded here:

www.r-project.org

It can be installed on Windows and Linux plattforms as well as on Macs.

R Basic



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

Rstudio

In this course we will use the rstudio gui which can be downloaded here:

www.rstudio.com

First operations

```
# Comments
```

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

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

- creates a vector with ten standard-normal-distributed values
- more information can be found here

```
mean(x)
```

```
## [1] -0.03308492
```

calculates the mean of variable x

More basic commands:

```
length(x)
```

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

```
max(x)
```

```
## [1] 1.470161
```

```
min(x)
```

```
## [1] -1.438529
```

```
sd(x)
```

```
## [1] 0.7771644
```

```
var(x)
```

```
## [1] 0.6039845
```

```
median(x)
```

```
## [1] 0.09008214
```

Getting help

- Introduction to R
- stackoverflow
- Thomas Girke - Programming in R

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

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:

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

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

```
##  [1] -0.69191136  0.19595419  1.34991630 -1.22889738  0.42167029
##  [6] -0.46015239 -0.11364512  0.71961694 -1.11921175  0.75742722
## [11]  0.27610333  0.75288802 -0.38420546  0.84810895  0.14894390
## [16] -0.13681510  0.92701314  0.01302583 -0.92732977 -0.79345348
```

Installing and Loading Packages

Many functions are already implemented in basic R. For more specific tasks libraries 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("lattice")
install.packages("survey")
```

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

Indexing

```
# 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
```

```r
A2[2,]
```

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

```r
A2[,1]
```

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

```r
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
```

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

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

Sequences

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

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

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

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

```
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
```

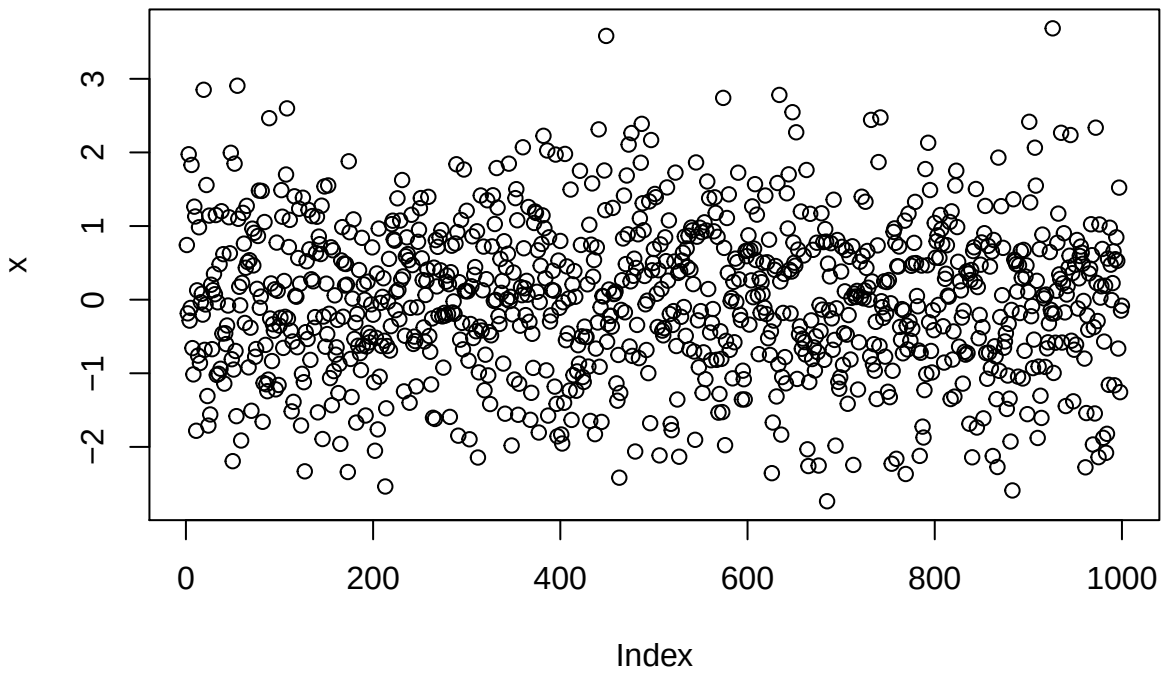
```
rep("A",10)
```

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

Basic Graphics

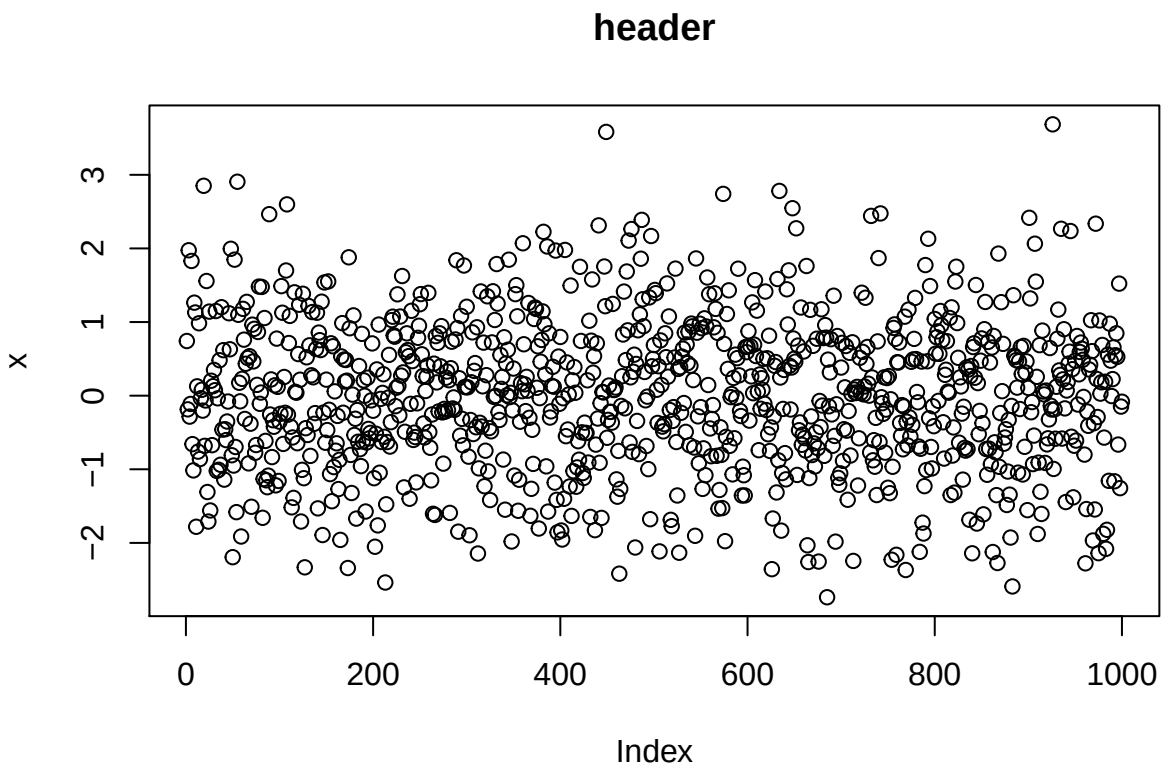
The plot function is the easiest option to get a graphic:

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



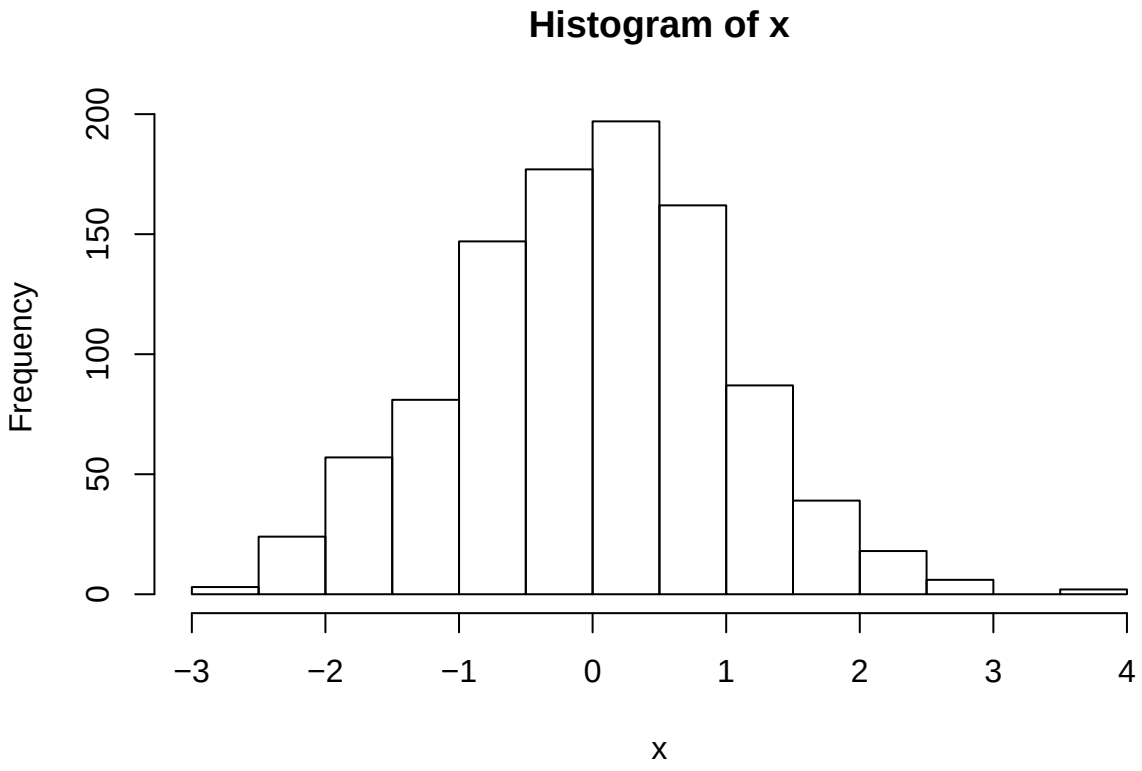
Adding a header:

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



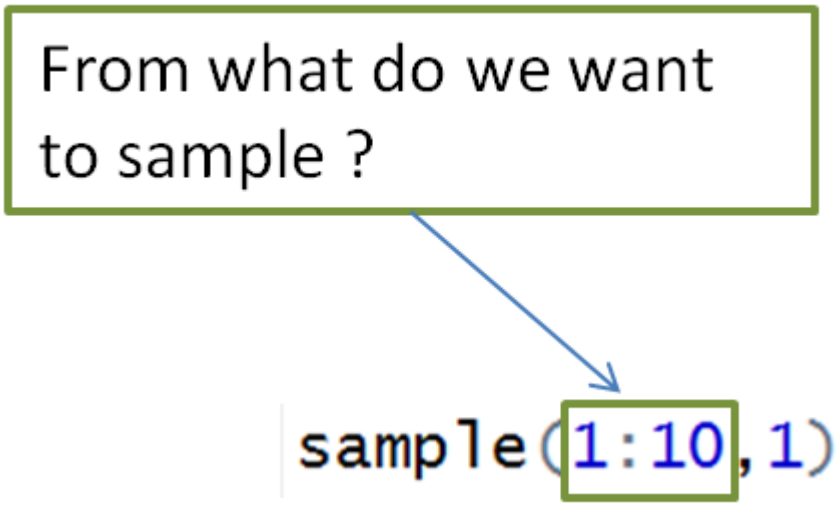
Histogram

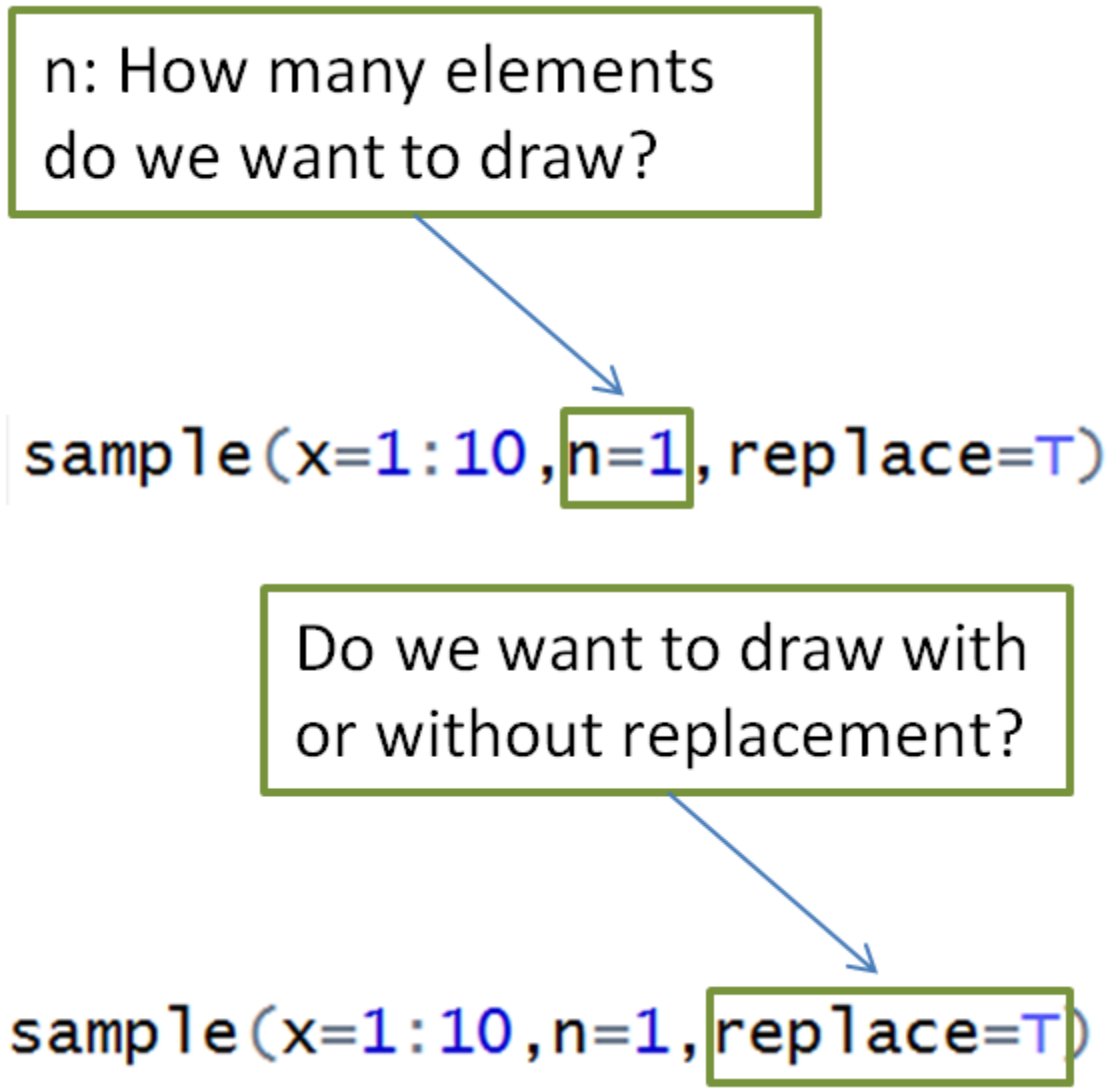
```
hist(x)
```



The sample function

Usage of the command sample






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

```
## [1] 7
```

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

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

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 Women04  Tot04  Men03 Women03
## 1 Aartselaar 11001        1        11   6971    7169  14140   7010    7243
## 2     Anvers 11002        1        11 223677  233642 457319 221767  232405
## 3   Boechout 11004        1        11   6027    5927  11954   6005    5942
## 4       Boom 11005        1        11   7640    8066  15706   7535    7952
## 5   Borsbeek 11007        1        11   4948    5328  10276   4951    5322
## 6 Brasschaat 11008        1        11  18142   18916  37058  18217   18903
##    Tot03 Diffmen Diffwom DiffTOT TaxableIncome Totaltaxation averageincome
## 1  14253     -39     -74    -113     242104077      74976114         33809
## 2 454172    1910    1237    3147    5416418842    1423715652         22072
## 3  11947      22     -15       7     167616996      50739035         29453
## 4  15487     105     114     219     186075961      46636930         21907
## 5  10273      -3       6       3     143225590      40564374         26632
## 6  37120     -75      13     -62     533368826     153629397         30574
##   medianincome
## 1        23901
## 2        17226
## 3        21613
## 4        17537
## 5        20739
## 6        21523
```

Also foreign datasets can be imported:

```
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
```

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

Merging

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.cs

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:

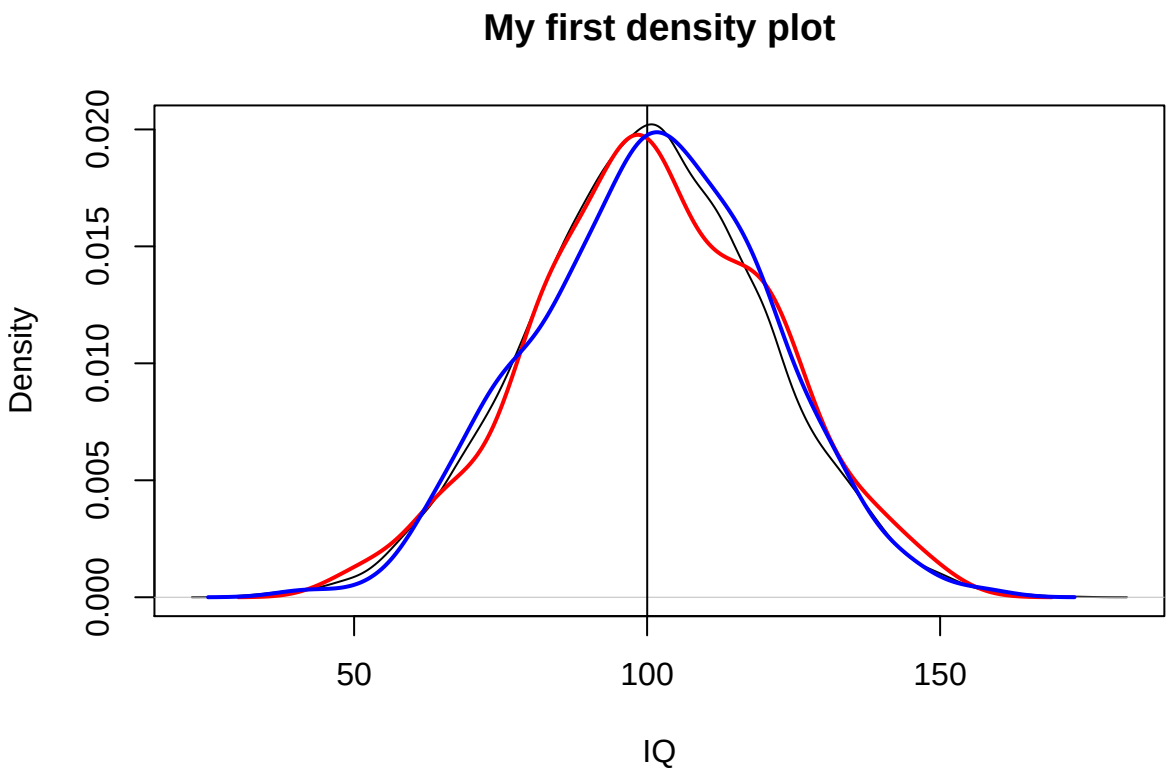
```
s.SRS <- sample(1:nrow(my.pop),500,replace=T)
s.SRSWOR <- sample(1:nrow(my.pop),500,replace=F)
```

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

```
##        X              id          gender      education         iq
##  Min.   :  18   Min.   :  18   female:248   average:130   Min.   : 46.12
##  1st Qu.:2411   1st Qu.:2411   male  :252   high   :136   1st Qu.: 87.25
##  Median :5034   Median :5034                low    :200   Median :100.34
##  Mean   :4960   Mean   :4960                none   : 34   Mean   :100.94
##  3rd Qu.:7542   3rd Qu.:7542                               3rd Qu.:115.62
##  Max.   :9926   Max.   :9926                               Max.   :153.23
```

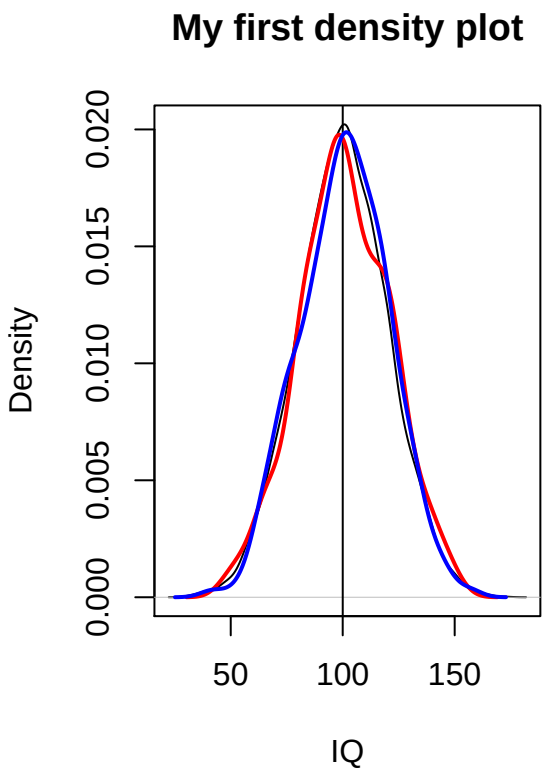
Making graphics to compare the samples:

```
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)
```

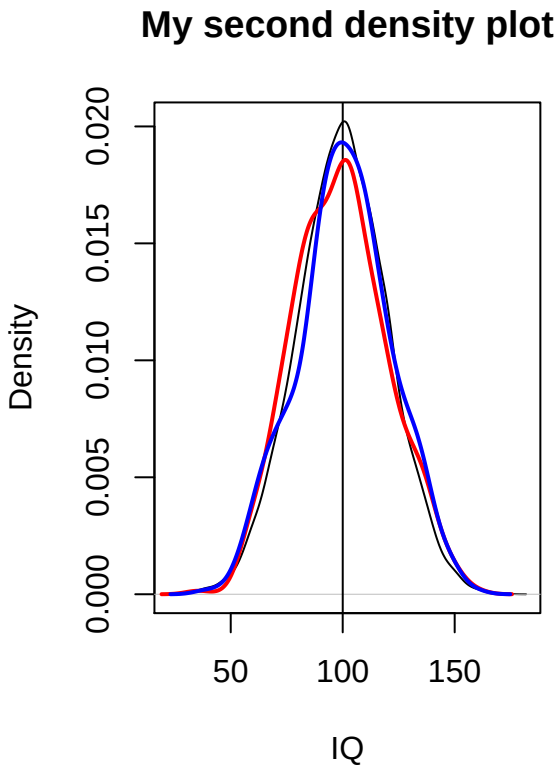



```
library("sampling")
set.seed(42)
s.SRS1 <- srswr(500,nrow(my.pop))
s.SRSWOR1 <- srswor(500,nrow(my.pop))
my.samp.SRS1 <- rbind(my.pop[s.SRS1!=0,]
                      ,my.pop[s.SRS1>1,])
my.samp.SRSWOR1 <- my.pop[s.SRSWOR1==1,]
```

```
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"