Preparation

Jan-Philipp Kolb, Matthias Sand and Stefan Zins

21 Januar 2016

Introduction

This document can be used for the preparation to the GRADE- workshop “Sampling and Estimation” at the University of Frankfurt. Hints for further reading are embedded 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.

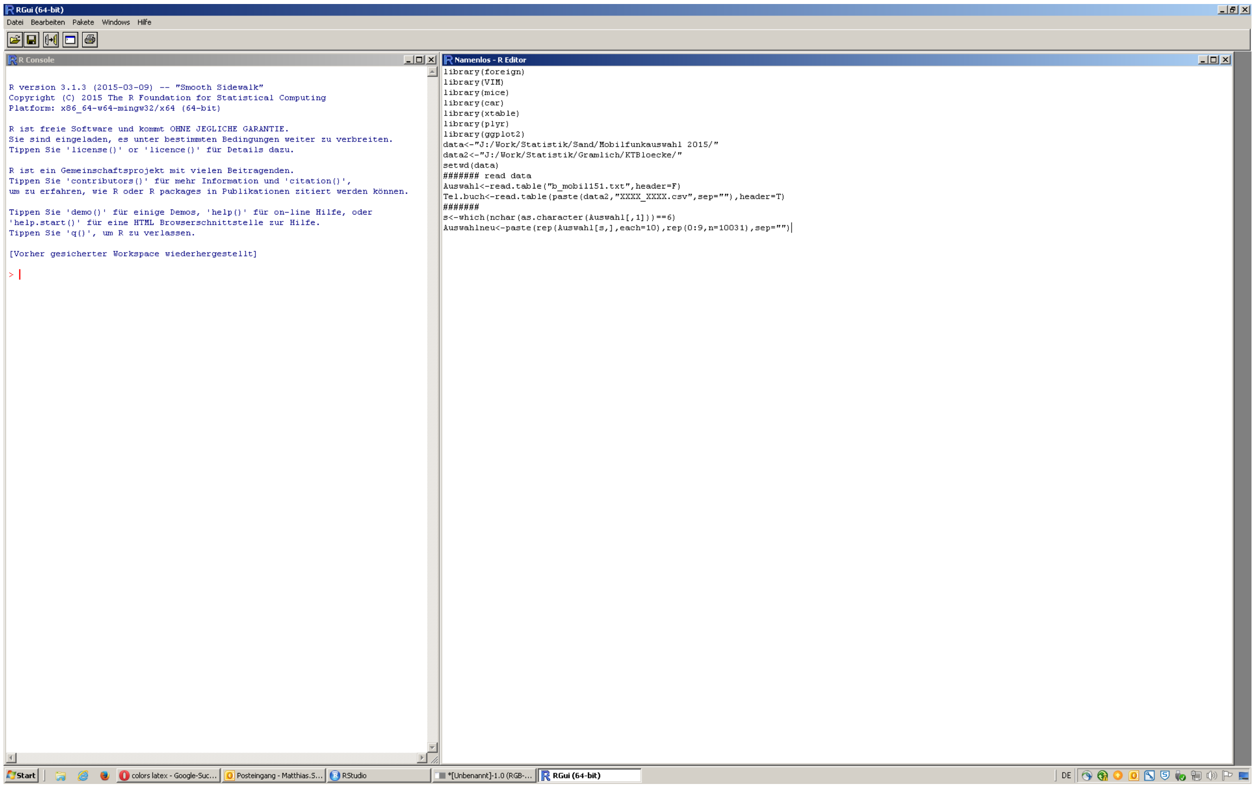
How to get R?

R can be installed on Windows and Linux plattforms as well as on Macs. If you have not done it 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:

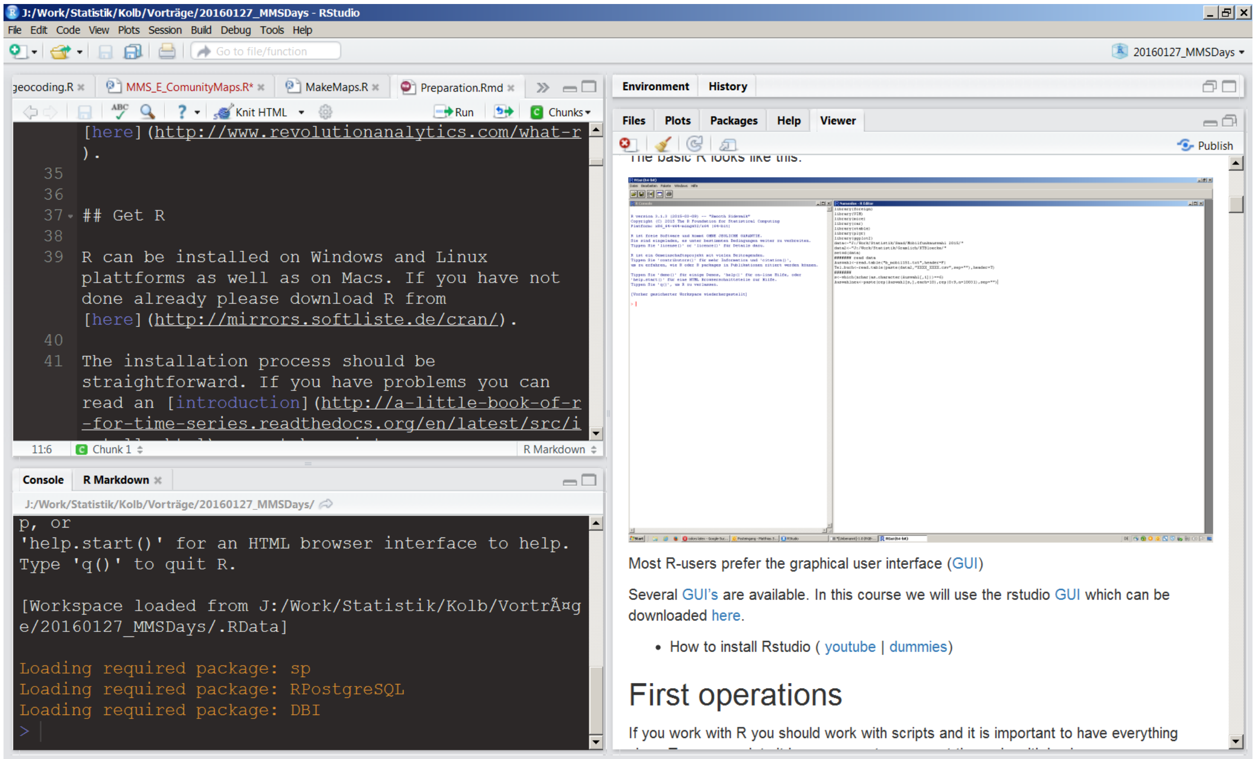


```
R version 3.1.3 (2015-03-09) -- "Smooth Sidewalk"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R ist freie Software und kommt OHNE JEGLICHE GARANTIE.
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.
Tippen Sie 'license()' or 'licence()' für Details dazu.

R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.
Tippen Sie 'contributors()' für mehr Information und 'citation()',
um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.

Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.

[Vorher gesicherter Workspace wiederhergestellt]

> 
```

```
library(foreign)
library(VIM)
library(mice)
library(car)
library(xtable)
library(plyr)
library(ggplot2)
data<-"J:/Work/Statistik/Sand/Mobilfunkauswahl 2015/"
data2<-"J:/Work/Statistik/Gramlich/KTBloecke/"
setwd(data)
####### read data
Auswahl<-read.table("b_mobil151.txt",header=F)
Tel.buch<-read.table(paste(data2,"XXXX_XXXX.csv",sep=""),header=T)
#######
s<-which(nchar(as.character(Auswahl[,1]))==6)
Auswahlneu<-paste(rep(Auswahl[s,],each=10),rep(0:9,n=10031),sep="")
```

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.



```
[here](http://www.revolutionanalytics.com/what-r
).


## 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](http://mirrors.softliste.de/cran/).

The installation process should be
straightforward. If you have problems you can
read an [introduction](http://a-little-book-of-r
-for-time-series.readthedocs.org/en/latest/src/i
```

```
p, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from J:/Work/Statistik/Kolb/VortrÃ¤g
e/20160127_MMSDays/.RData]

Loading required package: sp
Loading required package: RPostgreSQL
Loading required package: DBI
> 
```

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

- How to install Rstudio (youtube | dummies)

First operations

If you work with R you should work with scripts that should be well structured and lucid. To re-use 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] 0.4709898
```

calculates the mean of variable x

More basic commands:

```
length(x)
```

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

```
max(x)
```

```
## [1] 2.480975
```

```
min(x)
```

```
## [1] -1.352089
```

```
sd(x)
```

```
## [1] 1.244018
```

```
var(x)
```

```
## [1] 1.54758
```

```
median(x)
```

```
## [1] 0.569035
```

Errors and Warnings

If an error occurs - you have to fix it:

```
1/"a"
```

```
## Error in 1/"a": nicht-numerisches Argument für binären Operator
```

```
a <- 5
1/a
```

```
## [1] 0.2
```

You should always read the warnings, but sometimes you can ignore them:

```
a <- c(1,2,3)
b <- c(4,5)
cbind(a,b)
```

```
## Warning in cbind(a, b): number of rows of result is not a multiple of
## vector length (arg 2)
```

```
##      a b
## [1,] 1 4
## [2,] 2 5
## [3,] 3 4
```

Getting help

Countless introductions to R are 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 e.g. in respect of 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]  1.77820049  1.28257024 -1.57853446  0.10590360 -0.49937097
##  [6]  0.31166292 -0.55803206  1.46068910  1.00098182  0.65627229
## [11] -0.68706585  0.02171968 -0.29612342 -0.49144347  0.48665233
## [16] -1.81519504  0.47540448 -1.52105960 -0.24762693  1.70467267
```

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
- plyr - Tools for Splitting, Applying and Combining Data
- Matrix - Sparse and Dense Matrix Classes and Methods

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

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.

First indexing for vectors:

```
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
```

Indexing for dataframes:

```
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
```

Indexing for arrays:

```
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
```

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

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

Indexing for list's:

```
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
```

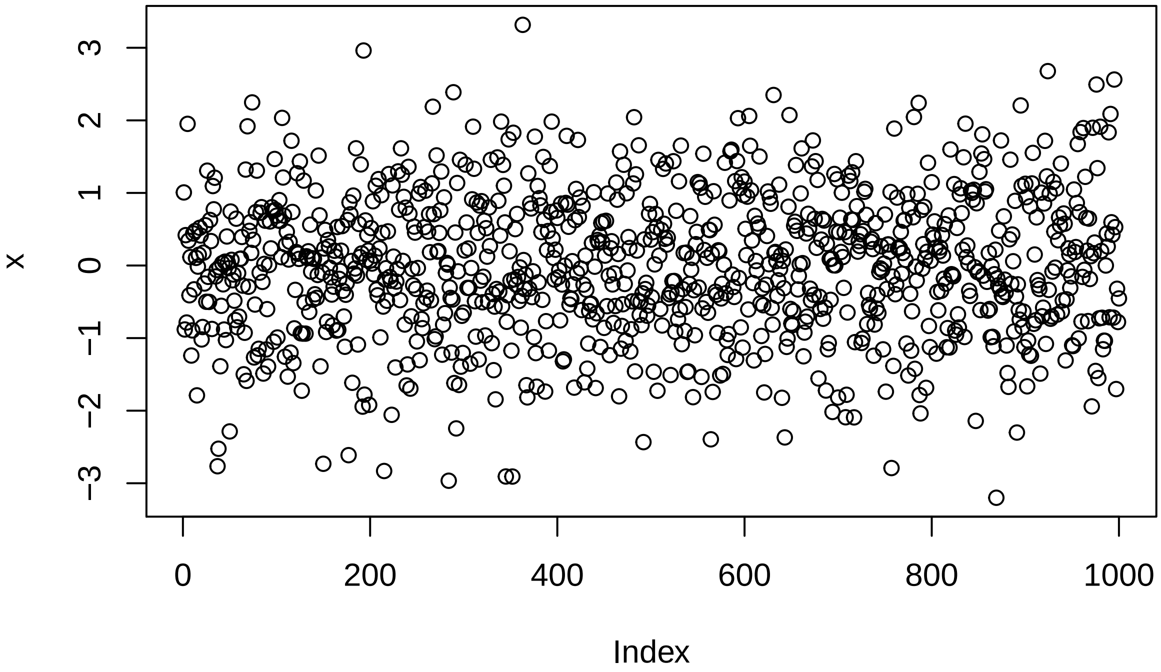
```
rep("A",10)
```

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

Basic Visualisations

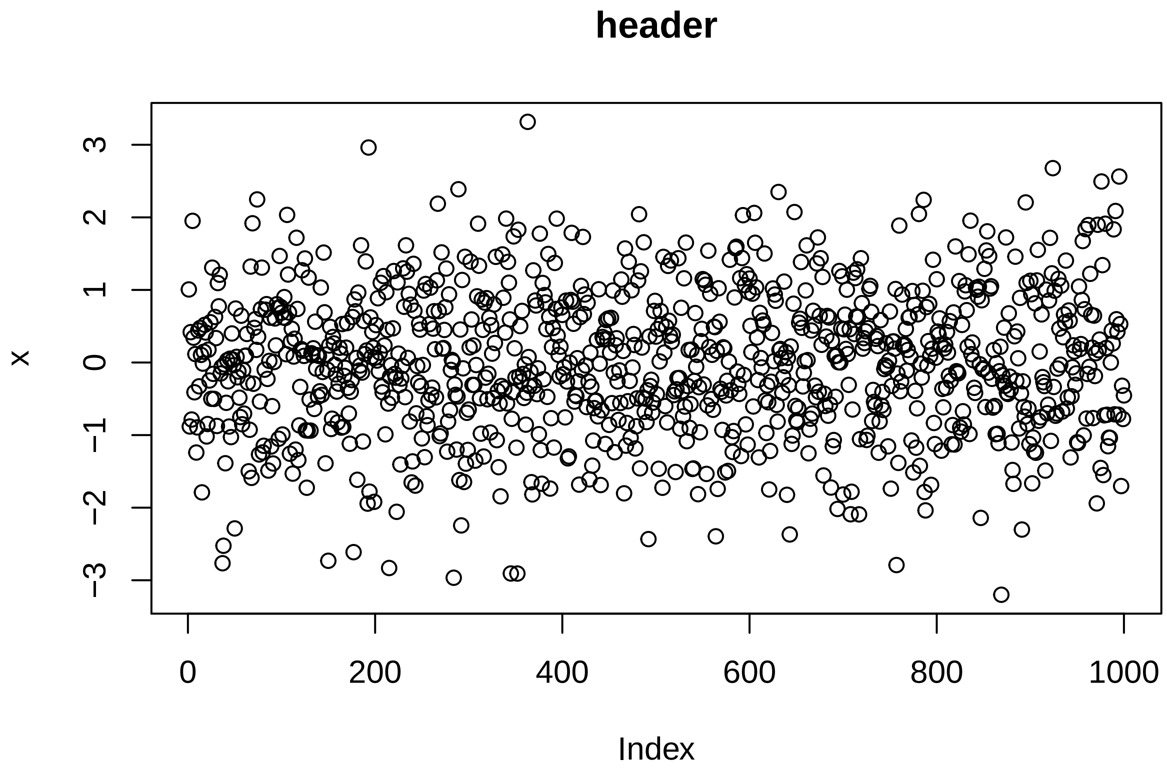
The plot function is the easiest option to get a graphic:

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



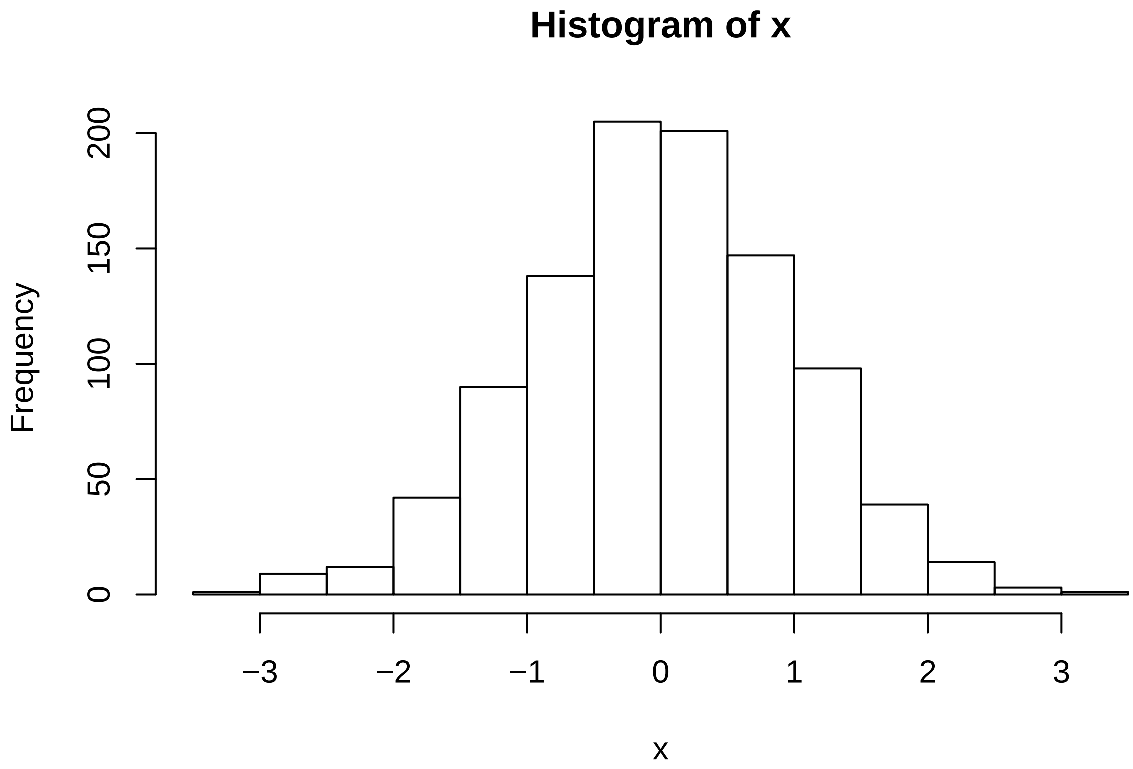
Adding a header:

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



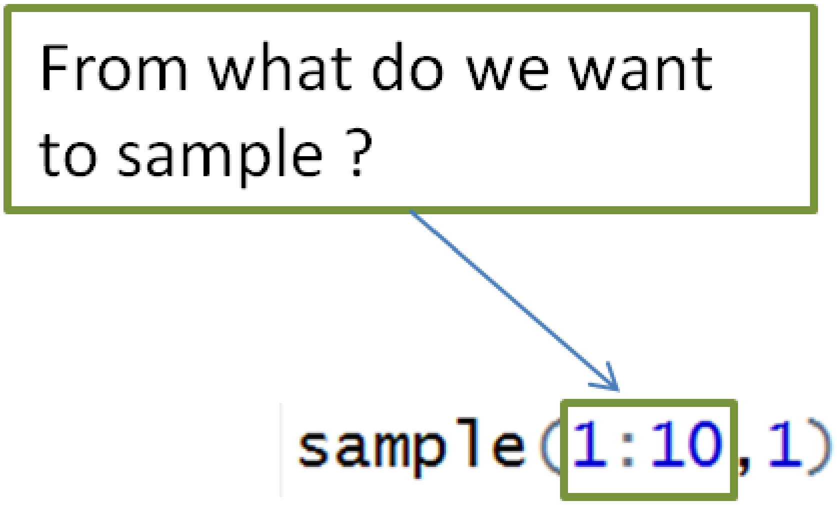
If we want a histogram, we can use the following command:

```
hist(x)
```



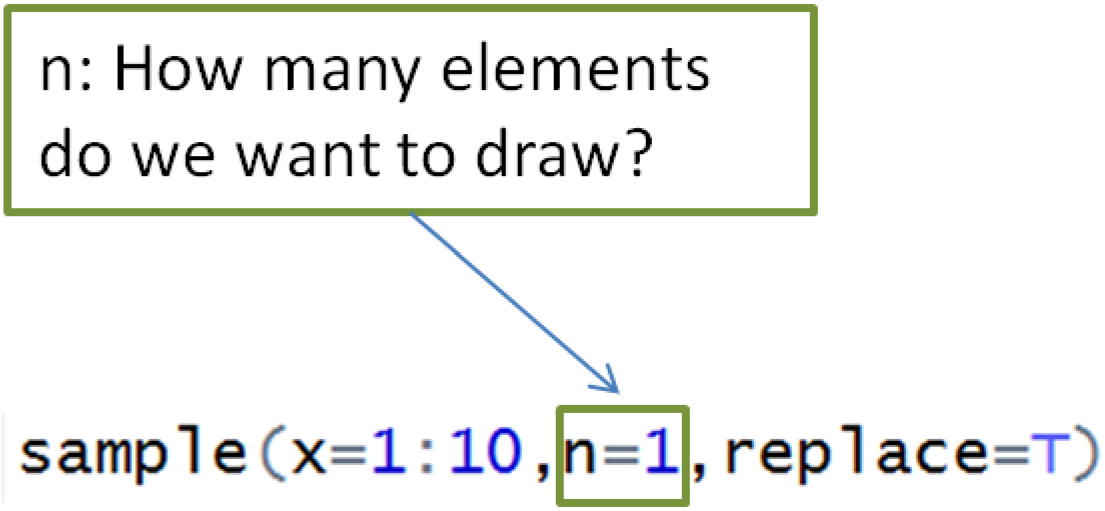
The sample function

Usage of the command sample



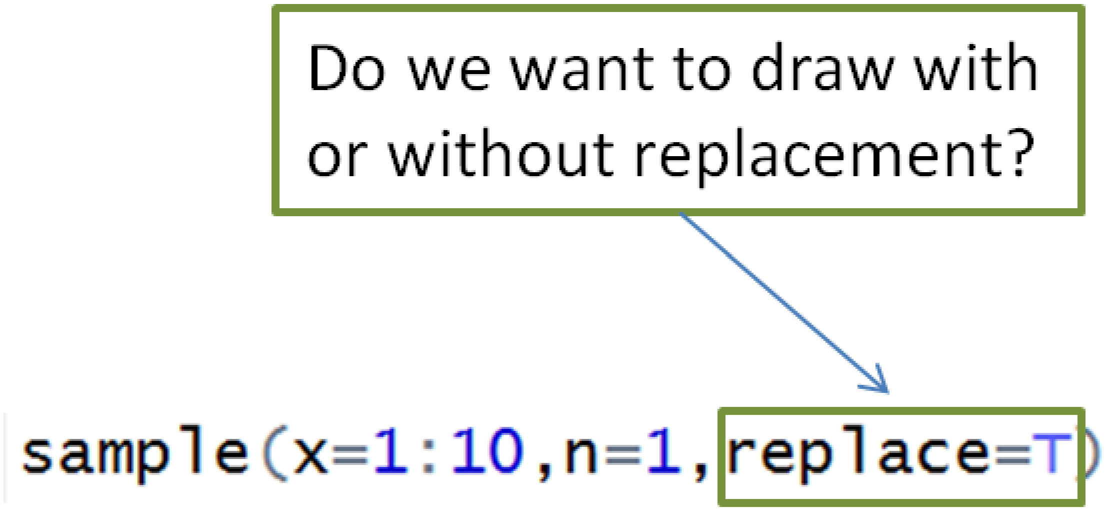
From what do we want to sample ?

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



n: How many elements do we want to draw?

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



Do we want to draw with or without replacement?

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

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

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

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

```
## [1] 9
```

Working Directory and Workspace

Declaring a working directory (you need to tell R where your data is saved).

```r
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()) # deletes all objects in your current workspace
```

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	male	high	123.26218
2	2	male	none	96.19531
3	3	male	low	94.21088
4	4	female	high	92.02308
5	5	male	average	114.18485
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 as 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 will be 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 Data

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

```
library("sampling")
data(belgianmunicipalities)
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 sure 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. You get the helpfile with:

```
?merge
```

There you will find code which can be copy-pasted to the console:

Use the following lines of code to produce a `data.frame` authors:

```
authors <- data.frame(
    surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
    nationality = c("US", "Australia", "US", "UK", "Australia"),
    deceased = c("yes", rep("no", 4)))
```

Use the following lines of code to produce a `data.frame` books:

```
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"))
```

Merge the do `data.frames` authors and books:

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

```
head(m1)
```

surname	nationality	deceased	title	other.author
McNeil	Australia	no	Interactive Data Analysis	NA
Ripley	UK	no	Spatial Statistics	NA
Ripley	UK	no	Stochastic Simulation	NA
Tierney	US	no	LISP-STAT	NA
Tukey	US	yes	Exploratory Data Analysis	NA
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)
```

```r
head(my.pop)
```

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

The dollar sign can also be used to access the columns

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

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

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

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

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

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

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

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

Apply family

Apply functions over array margins, ragged arrays or lists. To show that we first need an example data set:

```r
ApplyDat <- cbind(1:4,runif(4),rnorm(4))
```

To compute the mean for every row, we can use the `apply` command.

```r
apply(ApplyDat,1,mean)
```

```
## [1] 0.5919877 0.8588252 1.1218726 1.3765108
```

Mean for every column:

```r
apply(ApplyDat,2,mean)
```

```
## [1]  2.5000000  0.5027842 -0.0408870
```

Simple Example on Sampling

Summary of the dataset:

```r
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  
```

```r
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
```

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

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

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

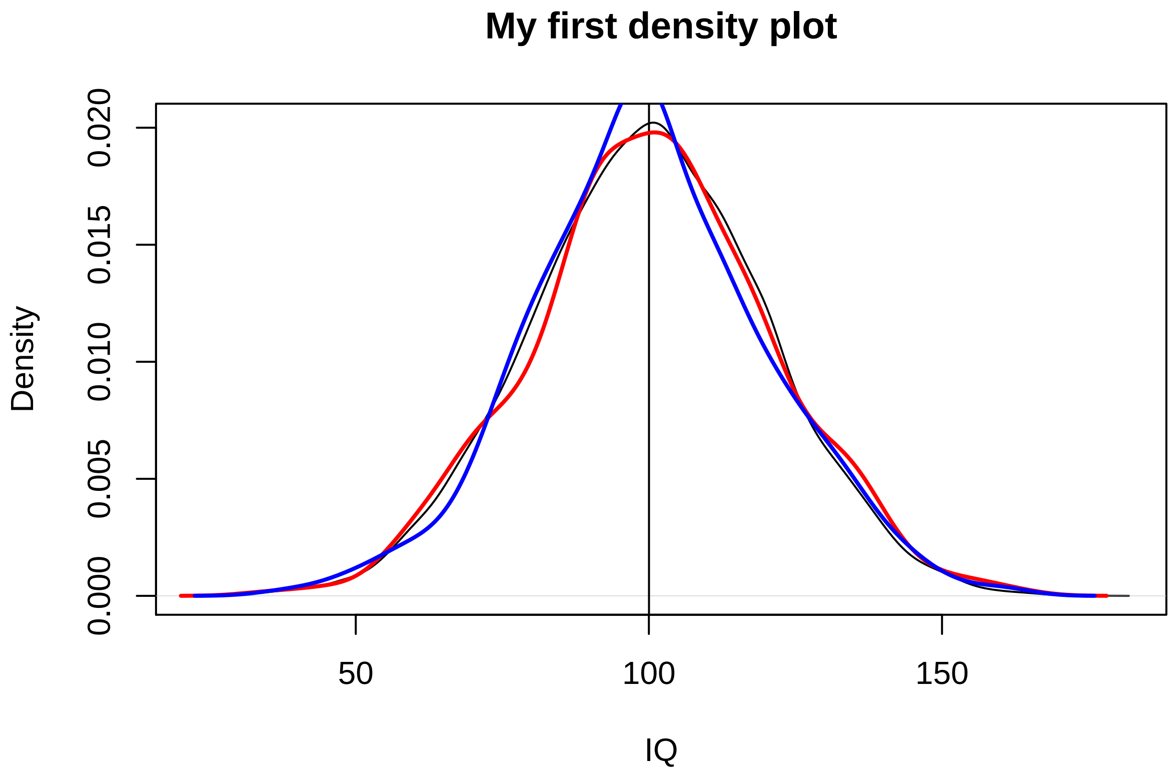
```r
s.SRS <- sample(1:nrow(my.pop),500,replace=T)
s.SRSWOR <- sample(1:nrow(my.pop),500,replace=F)
```

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

```
##        X               id           gender       education         iq        
##  Min.   :   8   Min.   :   8   female:260   average:149   Min.   : 35.36  
##  1st Qu.:2577   1st Qu.:2577   male  :240   high   :149   1st Qu.: 87.85  
##  Median :5172   Median :5172                low    :159   Median :100.68
##  Mean   :5025   Mean   :5025                none   : 43   Mean   :100.59
##  3rd Qu.:7448   3rd Qu.:7448                              3rd Qu.:113.98
##  Max.   :9997   Max.   :9997                              Max.   :162.88
```

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)
```

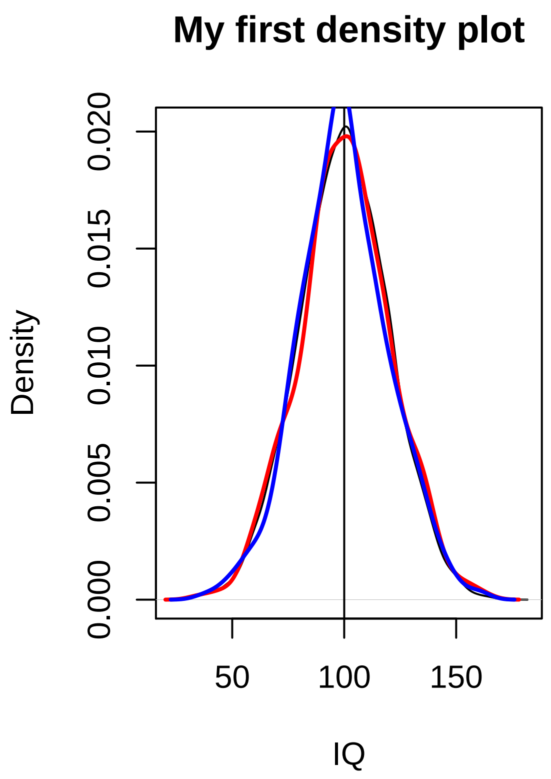


The package `sampling` is very useful to draw samples. An introduction to the package can be found here.

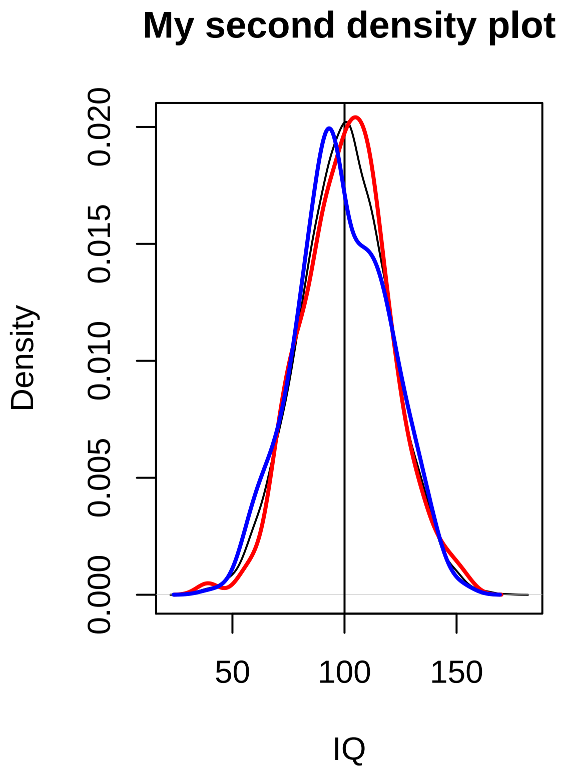
```
library("sampling")
s.SRS1 <- srswr(500,nrow(my.pop))
s.SRSWOR1 <- srswor(500,nrow(my.pop))
my.samp.SRS1 <- rbind(my.pop[s.SRS1!=0,]
                      ,my.pop[s.SRS1>1,])
my.samp.SRSWOR1 <- my.pop[s.SRSWOR1==1,]
```

```
par(mfrow=c(1,2))
plot(density(my.pop$iq),main = "My first density plot"
     , xlab = "IQ")
```

```r
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)
```



```r
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"

Links and resources for the workshop

- Kerns - Introduction to Probability and Statistics Using R
- Sharon Lohr (1999) - Sampling: Design and Analysis
- Ganninger - Design effects model-based versus design-based approach