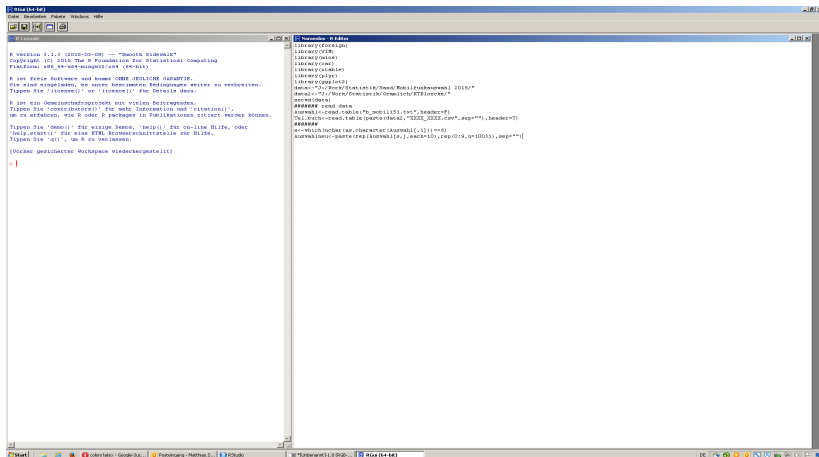# Tutorial: Sampling, Weighting and Estimation
Day 1

Stefan Zins, Matthias Sand
and Jan-Philipp Kolb

GESIS - Leibniz Institute
for the Social Sciences

- Open Source

- You can work with several datasets at the same time

- You can create your own objects, functions and packages

- Over 5,000 packages contributed by users available on CRAN

→ Rapid implementation of new (scientific) developments

→ Quick development of new tools that fit the user's demand

https://www.r-project.org

- most R-user prefer the graphical user interface (GUI) `RStudio`

https://www.rstudio.com

- `<-` assignment operator
- `#` can be used to comment your script
- `x<-rnorm(10,0,1)` creates a vector with ten standardnormal-distributed values
- `mean(x)` calculates the mean of variable `x`; `length(x)` returns the number of observations in `x`

```
mean(x)
```

```
[1] -0.3840465
```

```
length(x)
```

```
[1] 10
```

Getting Help

- *?command*
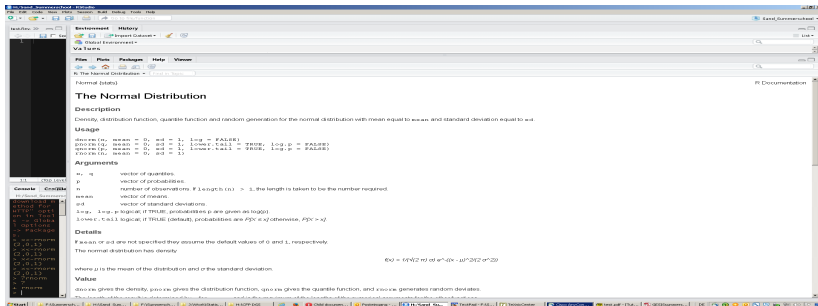
- CRAN
- Quick-R
- stackoverflow.com

numeric	`x<-c(1,2)`
logical	`x<-c(T,F)`
character	`x<-c("A","B")`
factor	`x<-as.factor(c("White","Black"))`

`str()` returns the type of your data

```
x<-as.factor(c("White","Black"))
str(x)

Factor w/ 2 levels "Black","White": 2 1
```

Indexing a Vector:

```
A1<-c(1,2,3,4)
A1[1]
```

[1] 1

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

[1] 1 2 3

```
A1[-2]
```

[1] 1 3 4

Indexing a data frame:

```
A2<-4:1
AA<-cbind(A1,A2)
AA[1,]
```
```
A1 A2
 1  4
```
```
AA[,1]
```
```
[1] 1 2 3 4
```
```
AA[1:3,2]
```
```
[1] 4 3 2
```
```
AA[,-1]
```
```
[1] 4 3 2 1
```

Indexing an 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

A3[,,2]

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

Indexing a list

```
A4<-list(A1,c("Summer","Winter"))
A4

[[1]]
[1] 1 2 3 4

[[2]]
[1] "Summer" "Winter"

A4[[1]]

[1] 1 2 3 4
```

```
1:5
[1] 1 2 3 4 5
rep("A",times=10)
 [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
rep(1:3,times=2,each=3)
 [1] 1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3
seq(-5,5,by=2.5)
[1] -5.0 -2.5  0.0  2.5  5.0
```

Function	Distribution	Important parameter
runif()	Uniform distribution	n, min, max
rnorm()	Normal distribution	n, mean, sd
rpois()	Poisson distribution	n, lambda
...	...	...

Function	Meaning	Example
length()	Length	length(x)
max()	Maximum	max(x)
min()	Minimum	min(x)
sd()	Standard deviation	sd(x)
var()	Variance	var(x)
mean()	Mean	mean(x)
median()	Median	median(x)

These functions do only need one argument
Other functions need to be specified by further arguments:

quantile()	90% Quantile	quantile(x,.9)
sample()	Draw a sample	sample(x,1)

Function	Meaning	Example
length()	Length	length(x)
max()	Maximum	max(x)
min()	Minimum	min(x)
sd()	Standard deviation	sd(x)
var()	Variance	var(x)
mean()	Mean	mean(x)
median()	Median	median(x)

These functions do only need one argument
Other functions need to be specified by further arguments:

quantile()	90% Quantile	quantile(x,.9)
sample()	Draw a sample	sample(x,1)

- R is a modular program with many functions included in basic R

Function	Meaning	Example
length()	Length	length(x)
max()	Maximum	max(x)
min()	Minimum	min(x)
sd()	Standard deviation	sd(x)
var()	Variance	var(x)
mean()	Mean	mean(x)
median()	Median	median(x)

These functions do only need one argument
Other functions need to be specified by further arguments:

quantile()	90% Quantile	quantile(x,.9)
sample()	Draw a sample	sample(x,1)

- R is a modular program with many functions included in basic R

- more specific functions are embedded in further packages

`install.package("sampling")`
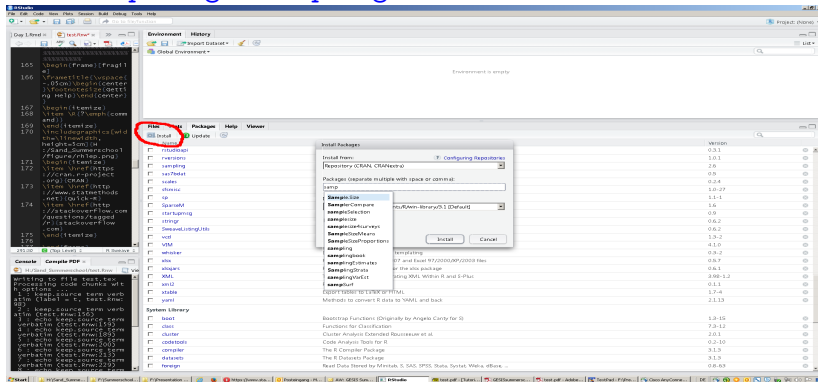
`library(sampling)` *or* `require(sampling)`

Library	Subject
`foreign`	reading and writing of data in numerous formats (e.g. *.dta*, *.sav*)
`sampling`	drawing and weighting samples
`survey`	analysis of complex survey samples
`xlsx`	read and write data in Excell-Format
`xtable`	export tables to LaTex and HTML
`mice`	multiple imputation by chain equation
`reshape`	alter structure of datasets
`car`	applied regressions
`VIM`	visualization and imputation of Missing Values
`lattice`	high-level data visualization
`ggplot2`	grammar for graphics in `R`

```
set.seed(42)
x <- rnorm(1000,0,1)
plot(x)
```
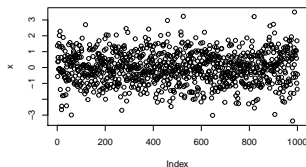
`set.seed()` is used to specify a starting point

```
hist(x)
```
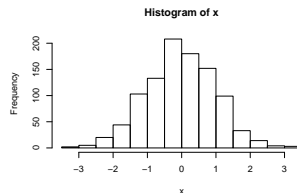

⇒ we will use *42* as seed-value for future exercises to obtain comparable results

sample {base}   R Documentation

Random Samples and Permutations

Description

sample takes a sample of the specified size from the elements of x using either with or without replacement.

Usage

sample(x, size, replace = FALSE, prob = NULL)

> x: From what do we want to sample ?

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

sample {base}   R Documentation

Random Samples and Permutations

Description

sample takes a sample of the specified size from the elements of x using either with or without replacement.

Usage

sample(x, size, replace = FALSE, prob = NULL)

n: How many elements
do we want to draw?

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

sample {base}   R Documentation

Random Samples and Permutations

**Description**

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

```
sample(x, size, replace = FALSE, prob = NULL)
```

> Do we want to draw with or without replacement?

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

```
id <- 1:10000
set.seed(42)
education <- sample(c("none","low","average","high"),10000,
+                   replace = T,prob = c(.072,.356,.289,.283))
gender <- sample(c("male","female"),10000,
+                 replace = T,prob = c(.488,.512))
iq <- rnorm(10000,100,20)
my.pop <- data.frame(id,gender,education,iq)
head(my.pop)

  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
```

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

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

$\Rightarrow \sigma^2$

```
set.seed(42)
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)

      id          gender      education         iq
 Min.   :   3   female:257   average:132   Min.   : 45.95
 1st Qu.:2322   male  :243   high   :134   1st Qu.: 85.38
 Median :4804                low    :192   Median :100.00
 Mean   :4896                none   : 42   Mean   : 99.60
 3rd Qu.:7434                              3rd Qu.:113.20
 Max.   :9966                              Max.   :165.63

nrow(unique(my.samp.SRS))

[1] 487
```

```
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)
```

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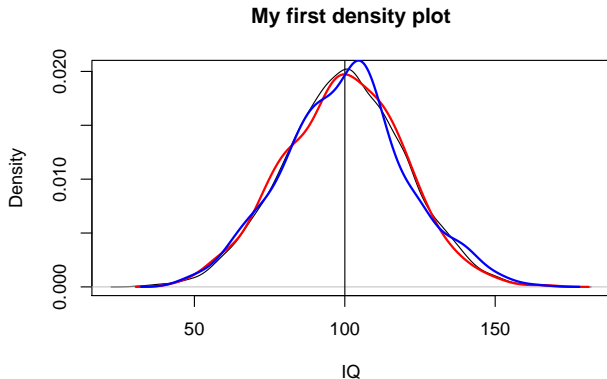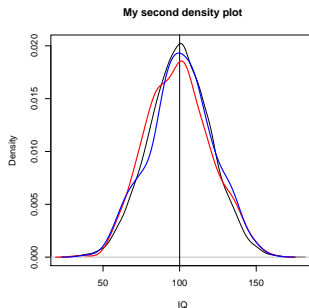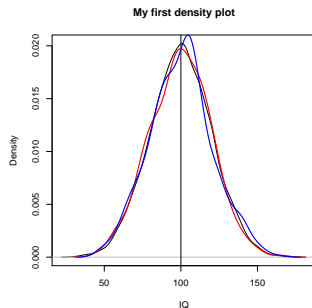
```
library(sampling)
set.seed(42)
s.SRS1 <- srswr(500,nrow(my.pop))
s.SRSWOR1 <- srswor(500,nrow(my.pop))
my.samp.SRS1 <- rbind(my.pop[s.SRS1!=0,]
+                      ,my.pop[s.SRS1>1,])
my.samp.SRSWOR1 <- my.pop[s.SRSWOR1==1,]
```

```
par(mfrow=c(1,2))
```


My first density plot

My second density plot
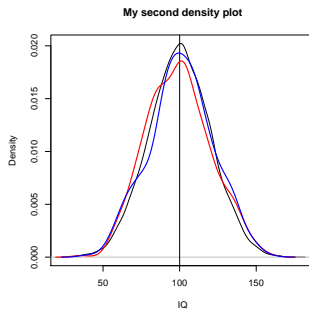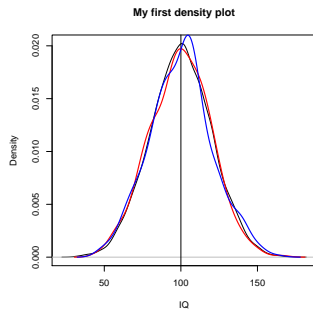
```
dev.off()
```

```
par(mfrow=c(1,2))
```



```
dev.off()
```

- should yield same results
⇒ routine differs in "starting point"

Declaring a working directory

```
path<-"H:/Sand_ Summerschool/Data Day1/"
setwd(path)
```

- 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

Example:

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

Writing/ saving data and results

```
write.table(my.pop,"Synthetic Data Day1.csv",
row.names = F, quote = F, dec = ".",sep = ",")
```
OR:
```
save(my.samp.SRS,s.SRS,my.samp.SRSWOR1,file = "Day1.Rdata")
```

⇒ See also: `write.csv` and `write.csv2` (sep = ";")

Reading/ loading data and results

```
d1 <- read.table("Synthetic Data Day1.csv",
header = F, dec = ".",sep = ",")
```
OR:
```
load("Day1.Rdata")
```