An introduction to R
WS 2014/2015

Dr. Noémie Becker (AG Metzler)
Dr. Sonja Grath (AG Parsch)

Special thanks to: Prof. Dr. Martin Hutzenthaler
(previously AG Metzler, now University of Duisburg-Essen)
course development, lecture notes, exercises

Course outline – Day 4

Reading and writing data

- Data frames
- NA, Inf, NaN, NULL
- Editing data

Lecture notes, pp 24-35

Plotting

- High- and low-level plotting functions and arguments
- Mathematical symbols
- Interacting with plots
- Saving plots

Lecture notes, pp 36-62

Solution to the exercises

Reading and writing data

Data frames

General command: *data.frame*()

→ typical R representation of data sets

→ lists with constraint that all elements are vectors of the same length

name	gender	favourite_colour	income
Hans	male	green	800
Caro	female	blue	1233
Lars	male	yellow	2400
Ines	female	black	4000
Samira	female	yellow	2899
Peter	male	green	1100
Sarah	female	black	1900

How can you get your data into R?

Possibility 1

General command: data.frame()

→ type your data at the command line/within a script

group – name of the variable

name, gender, favourite_colour, income – column names

```
> group <- data.frame(
name = c("Hans", "Caro", "Lars", "Ines", "Samira", "Peter", "Sarah"),
gender = c("male", "female", "male", "female", "female", "male",
"female"),
favourite_colour = c("green", "blue", "yellow", "black", "yellow", "green",
"black"),
income = c(800, 1233, 2400, 4000, 2899, 1100, 1900)
)
```

Note that R uses the **equal sign** to specify named arguments to a function!

Possibility 2

- provide the data in a file (txt, csv)
- read in your data from that file

Typical call:

```
read.table("filename.txt", header=TRUE)

read.csv("filename.csv", header=TRUE)


write.table(dataframe, file="filename.txt")

write.csv(dataframe, file="filename.csv")
```

Example:
Workflow for reading and writing data frames

Steps:

1) Read in your data

2) Check your data

3) Perform your analyses

4) Write output

5) Close session

Data source:

data.txt

→ contains the data of the data frame we had before

Workflow - Script

```
# Load data
group <- read.table("data.txt", header=TRUE)
# Copy data into search path
attach(group)
# Get an overview of data
names(group)
str(group)
summary(group)
# ANALYSIS
# Remove data from search path
detach(group)
```

attach()/detach()

Copy data into search path:

```
attach()
```

Remove data from the search path:

```
detach()
```

Example:

```
data(mtcars)
summary(mtcars$mpg)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
summary(mpg)
Error in summary(mpg) : object 'mpg' not found
attach(mtcars)
summary(mpg)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
detach(mtcars)
```

attach()/detach()

Caution: Problem when more than one object has the same name!

Example:

```
# You define your own variable 'mpg'
mpg <- c(25,36,47)
data(mtcars)
attach(mtcars)
The following object(s) are masked _by_ '.GlobalEnv':

    mpg

mean(mpg)
[1] 36
mean(mtcars$mpg)
[1] 20.09062
mpg
[1] 25 36 47
```

Alternative to attach(): with()

```
with(mtcars, {
   summary(mpg)
})
```

Limitation of the with() function:

```
with(mtcars, {
   stats <- summary(mpg)
})
stats
Error: object 'stats' not found
```

Solution: <<- (saves object to the global environment)

```
with(mtcars, {
   nokeepstats <- summary(mpg)
   keepstats <<- summary(mpg)
})
nokeepstats
Error: object 'nokeepstats' not found
keepstats
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
```

More on data frames

We will work through the example from the lecture notes

(pp 26-29)

Steps:

1) Define your working directory	setwd()
2) Read in data (from *data.txt*)	read.table()
3) Check your data	names(), str(), summary()
4) Copy data into search path	attach()
5) Select subsets of your data	subset()
6) Split your data into a list of a subgroup	split()
7) Extend your data frame	merge()
8) Remove data from search path	detach()

Example
data.txt

name	gender	favourite_colour	income
Hans	male	green	800
Caro	female	blue	1233
Lars	male	yellow	2400
Ines	female	black	4000
Samira	female	yellow	2899
Peter	male	green	1100
Sarah	female	black	1900

NA, Inf, NaN, NULL

NA = not available

Inf = Infinity

NaN = Not a Number

Important command: is.na()

Example:

```
v <- c(1,3,NA,5)
is.na(v)
[1] FALSE FALSE TRUE FALSE
sum(v)
[1] NA
```

Ignore missing data: 'na.rm=TRUE'

```
sum(v, na.rm=TRUE)
[1] 9
```

Plotting

Plotting

There are three types of plotting commands:

High-level plotting functions create a new plot (usually with axes, labels, titles and so on)

Low-level plotting functions add more information to an existing plot, such as extra points, lines or labels

Interactive graphics functions allow you to interactively add information to an existing plot or to extract information from an existing plot using the mouse

High-level plotting functions

Function	Description
barplot()	Visualizes a vector with bars
boxplot()	Box- and whisker plot
contour()	The contour of a surface is plotted in 2D
coplot()	Conditioning-Plots
hist()	Histogram
mosaicplot()	Plot in form of a mosaic
pairs()	Produces a matrix of scatterplots
pie()	Circular pie charts
qqplot()	Quantile-quantile plot
…	

… many more – and R offers many packages for plotting (ggplot2, lattice...)

We will cover now: plot(), hist(), boxplot()

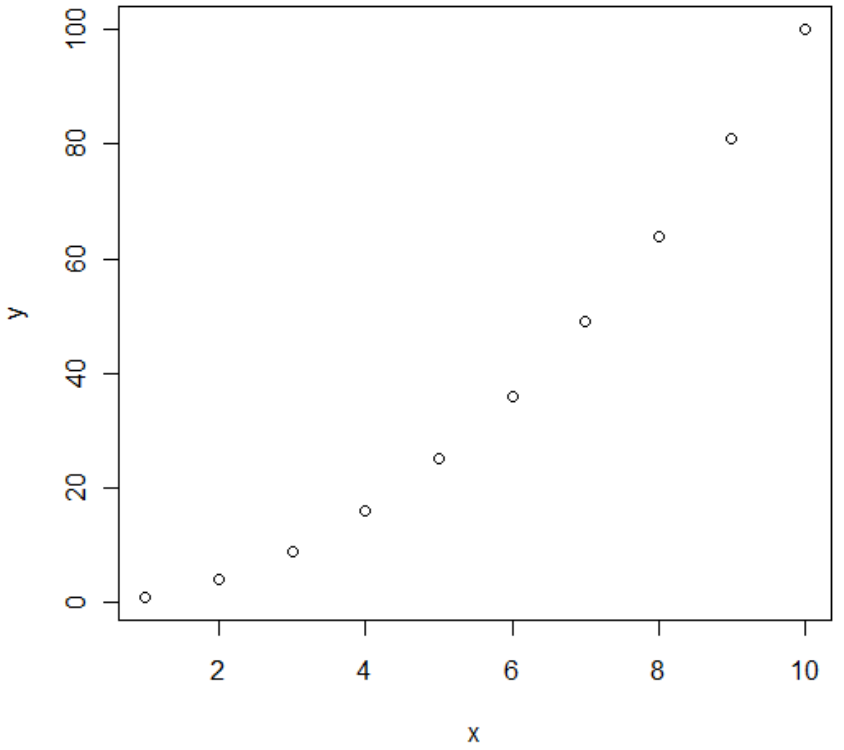
High-level function – plot()

- Standard high-level plotting function
- Behaviour of plot() depends on the type of its argument

plot(x,y)
If x and y are numerical vectors, then plot(x,y) produces a scatterplot of y against x

Example:

```
x <- 1:10
y <- x^2
plot(x,y)
```



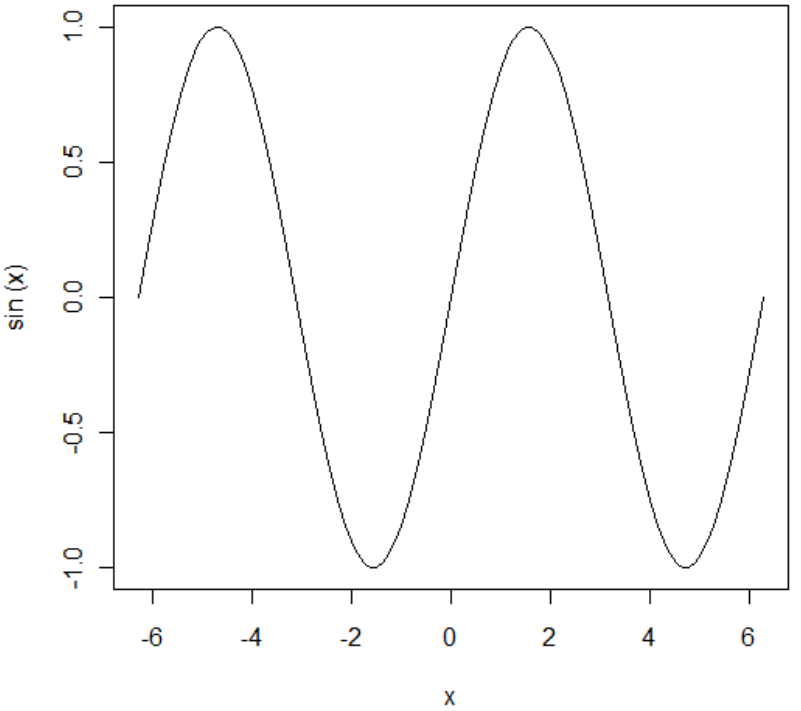
High-level function – plot()

- Standard high-level plotting function
- Behaviour of plot() depends on the type of its argument

plot(*fun*)
If *fun* is a function, then plot(*fun*, from=a, to=b) plots *fun* in the range [a, b]

Example 1:

```
plot(sin, from=-2*pi, to=2*pi)
```



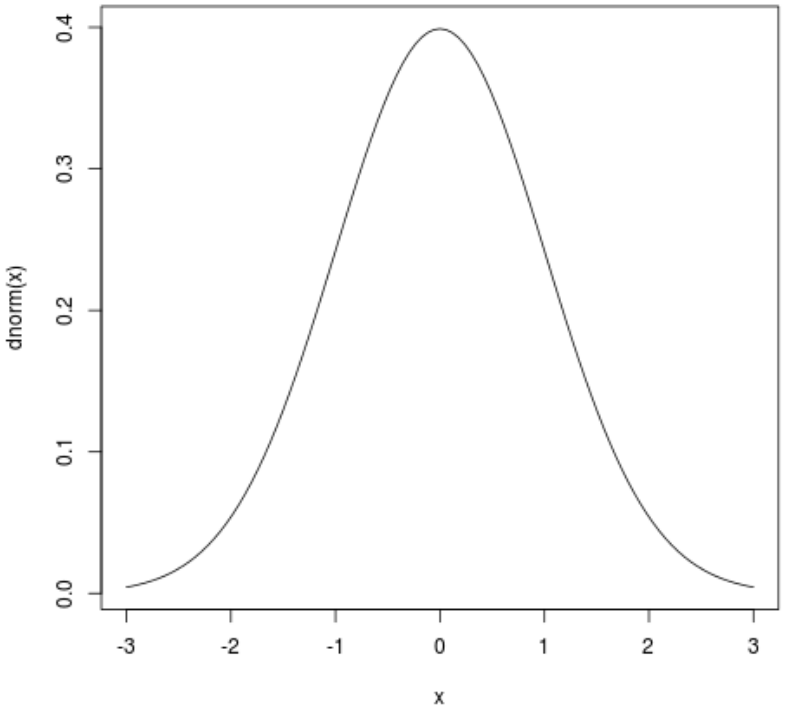
High-level function – plot()

- Standard high-level plotting function
- Behaviour of plot() depends on the type of its argument

plot(*fun*)
If *fun* is a function, then plot(*fun*, from=a, to=b) plots *fun* in the range [a, b]

Example 2:

```
plot(dnorm, from = -3, to = 3)
```

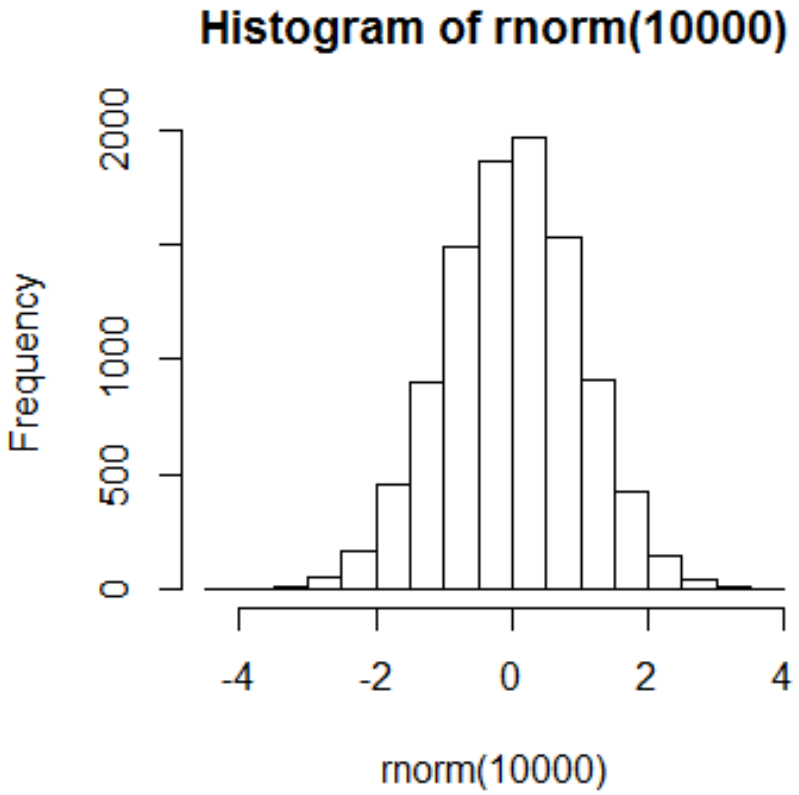


High-level function – hist()

- Histogram

Example 1:

```
hist(rnorm(10000))
```

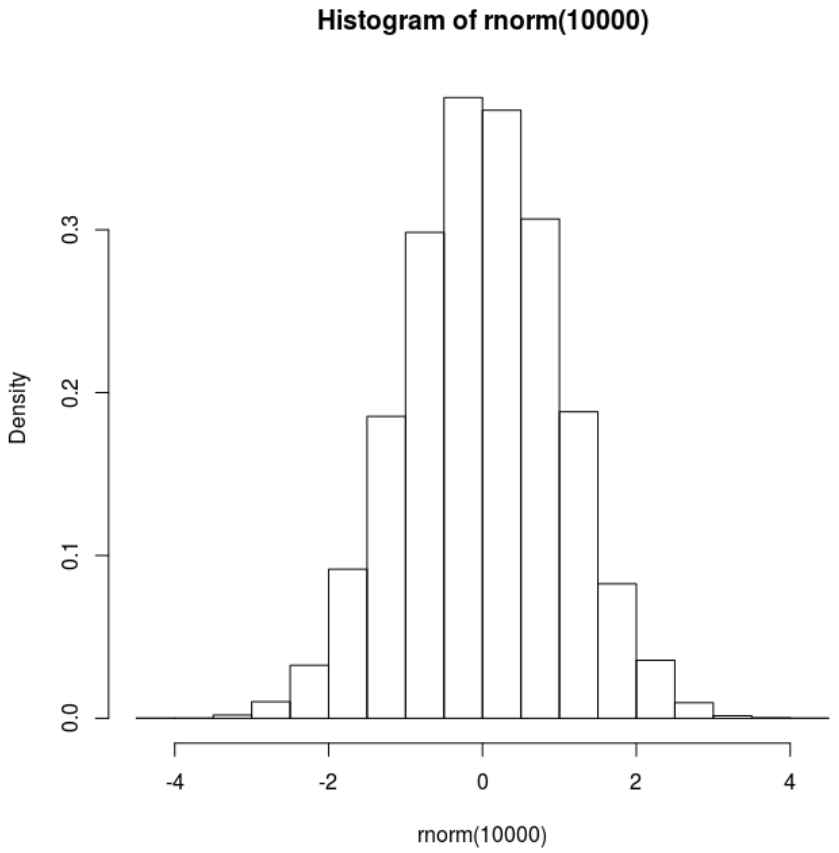


High-level function – hist()

➔ Histogram

Example 1:

```
hist(rnorm(10000), probability = TRUE)
```

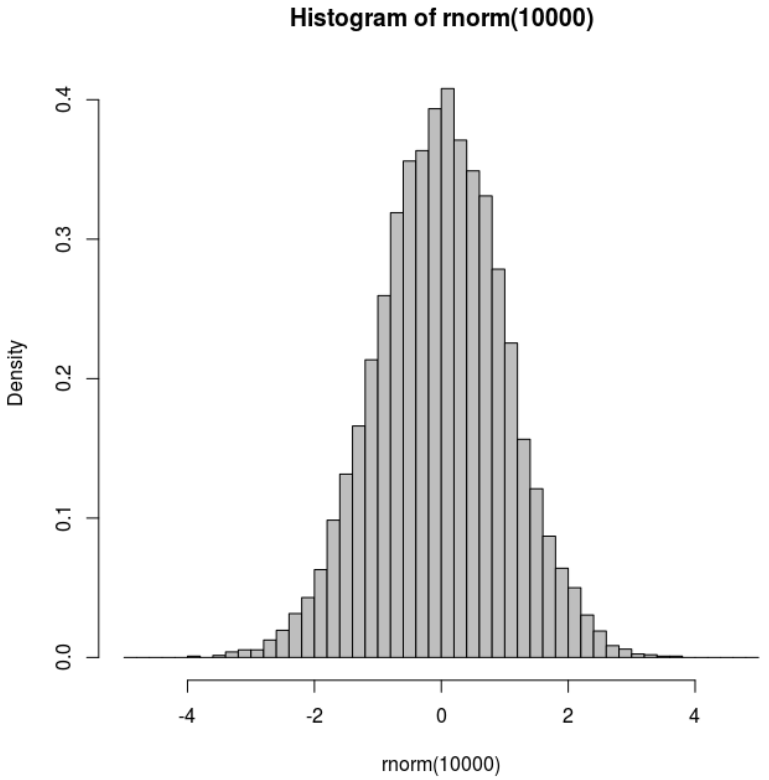


High-level function – hist()

- Histogram

Example 2:

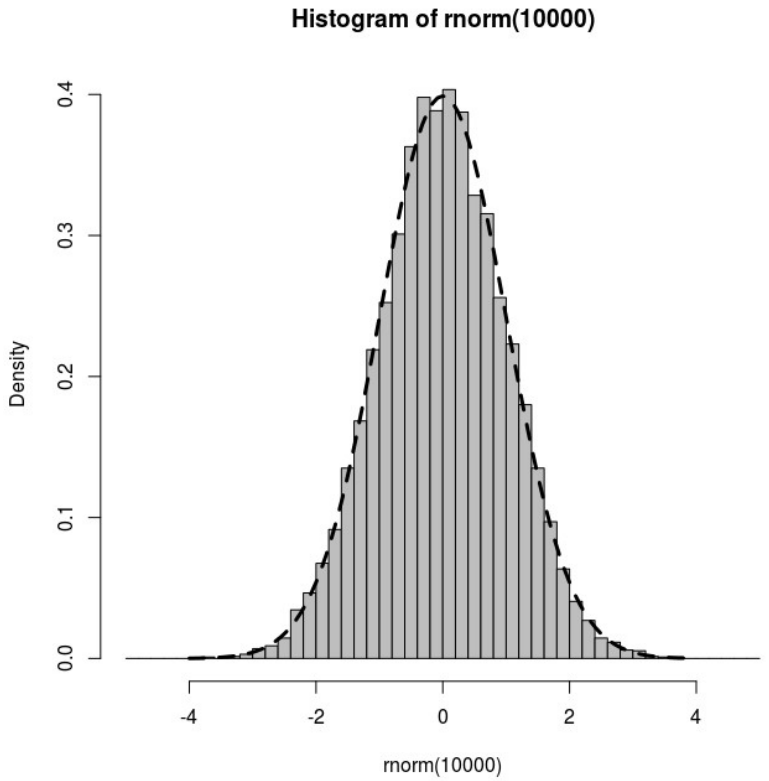
```
hist(rnorm(10000), probability=TRUE, col="grey",
breaks=seq(-5,5,by=0.2))
```



The histogram of 10000 simulated values is close to the density function

Example:

```
hist(rnorm(10000), probability=TRUE, col="grey",
breaks=seq(-5,5,by=0.2))
plot(dnorm, from=-4, to=4, add=TRUE, lwd=3,
lty="dashed")
```

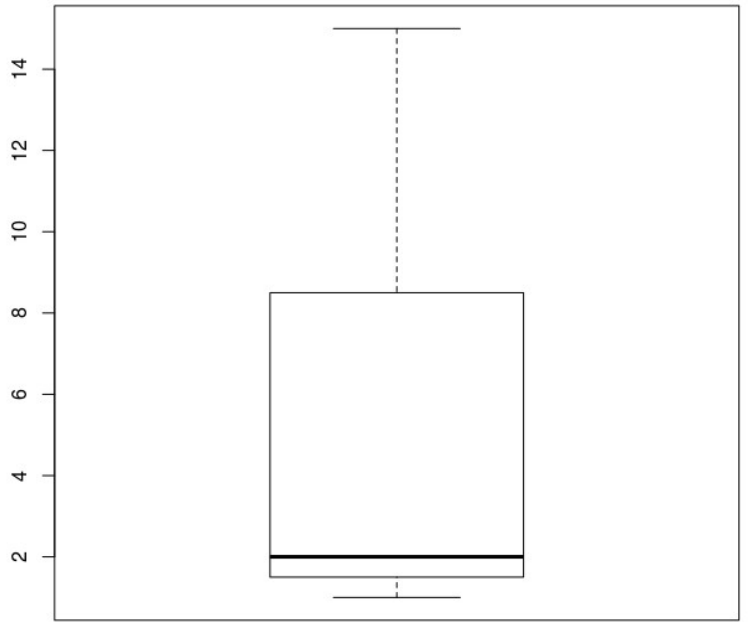


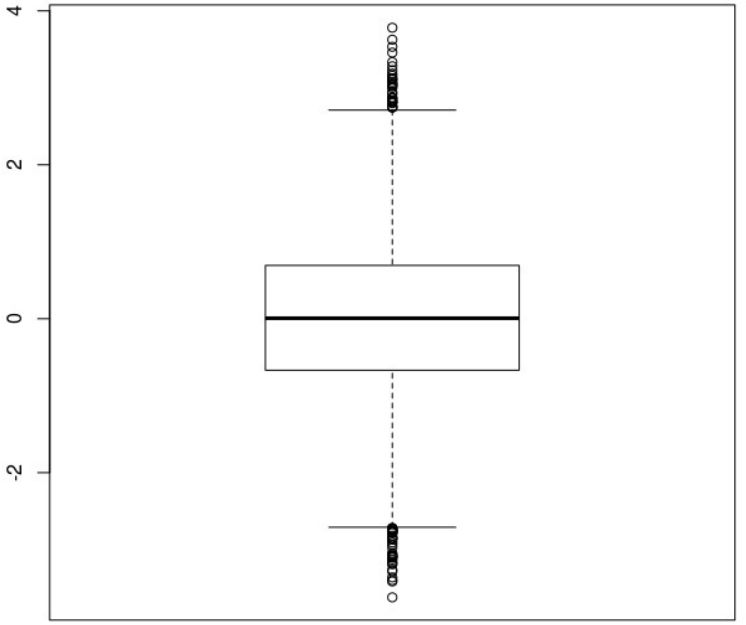
High-level function – boxplot()

- Box and whisker plot

Example:

```
boxplot(c(1,2,15))
boxplot(rnorm(10000))
```





Saving plots

➔ Several possibilities (see lecture notes pp 55/51)

(1) dev.print()

Example:

```
plot(...) # Begin a plot with an high-level plotting function
          #such as plot()
...       # Further low-level plotting function enrich the
          #plot
# After you are finished with the plot:
dev.print(device=pdf, file="filename.pdf"
```

→ filename.pdf now contains the plot you saw on the screen

Saving plots

(2) savePlot()

Usage:

```
savePlot(filename = "Rplot",
     type = c("wmf", "emf", "png", "jpg", "jpeg", "bmp",
               "tif", "tiff", "ps", "eps", "pdf"),
     device = dev.cur(),
     restoreConsole = TRUE)
```

Example:

```
savePlot(filename="Figure1.pdf", type="pdf")
```

→ Figure1.pdf now contains the plot you saw on the screen
→ It can be that not all types work for your system

Saving plots

(3) Plot directly into a file

Example:

```
x <- 1:10
y <- x^2
pdf("filename.pdf")
plot(x,y)
dev.off()
```

→ filename.pdf now contains the plot
→ the plot is not printed on screen
→ works for different devices

Important:

When you are done you have to close the printing device!

```
dev.off()
```

Exercise sheet 3