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 2

Review session day 1

Basics

- Matrices
- Data types in R

Lecture notes, pp 12-16

Basic statistics with R

- Some distributions implemented in R
- Examining the distribution of a set of data
- Random number generators

Lecture notes, pp 16-23

A sample R session

Review session

Libraries

R commands are organized in libraries (packages)

Examples: 'stats', 'datasets'

Loaded at start:

library(lib.loc=.Library)

Load packages:

library(*packagename*)

Further useful commands:

library(help="*packagename*")

installed.packages()

download.packages()

install.packages()

Pre-Defined Datasets

R comes with a huge amount of pre-defined datasets
Examples: 'cars', 'mtcars', 'chickwts', ...
→ can be used for exercises, demonstration of in-built functions...

How to use a dataset:

```
data(cars)
```

Help:

```
?cars
```

➔ Exercises

Organize your R session

General advice:

Organize your work in folders

Save your commands in scripts (text files)

Example:

```
# Define a vector with age values

age <- c(1,3,5,2,11,9,3,9,12,3)

# Define a vector with weight values

weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1)

# Calculate the mean weight value

mean(weight)

# Quit R session

q()
```

Organize your R session

General advice:

Organize your work in folders

Save your commands in scripts (text files)

Working directory:

getwd()

setwd()

Recipe:

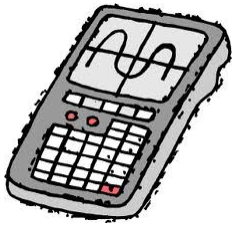
(1) Open your favourite text editor

(2) Save the file (*e.g.* example.R)

(3) Define first comments for your workflow

(4) Write your R commands and test them step-by-step

R as calculator



Basic arithmetic operations

2+3

7-4

3*5

7/3; 2^6

Comments

This is a comment

Integer vs. modulo divison

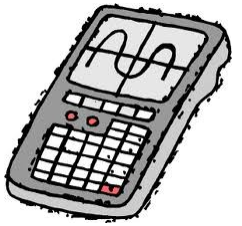
5 %/% 3 # “5 divided by 3 without decimal positions” → 1

5 %% 3 # “if you divide 5 by 3 – what's the rest?” → 2

Caution: German (Spanish, French..) decimal notation does not work!
> 1,2
Error: unexpected ',' in “1,”

> 1.2 ✔

R as calculator



Important functions

exp(1)

exp(log(5))

sin(pi/2)

cos(pi/2)

max(4,2,5,1)

min(4,2,5,1)

sum(4,2,5,1)

prod(4,2,5,1)

sqrt(16)

factorial(4)

choose(5,2)

factorial()
"4 factorial", 4!
→ 4*3*2*1

choose()
"5 choose 2", $\binom{a}{b}$

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}.$$

Help!

R console

```
help(solve)                 #help page for command “solve”

?(solve)                    #same as help(solve)

help("exp")

help.start()

help.search("solve")        #list of commands which could
                            #be related to string “solve”

??solve                     #same as help.search(“solve”)

example(exp)                #examples for the usage of 'exp'

example("*")                #special characters have to be in
                            #quotation marks
```

Assignments

General form:

variable <- *value*

Example:

x <- 5

“The ***variable*** 'x' is assigned the ***value*** '5'”

Valid variable names: contain numbers, '_', characters

NOT allowed: '.' followed by number at the beginning

.4you

Allowed:

my.variable, my_variable, myVariable

favourite_color, a, b, c, data2, 2data ...

Assignments

```
x <- 5          # The variable x is assigned the value 5

5 -> x          # The same assignment but unusual

x = 5           # The same assignment but unusual
```

Works with longer expressions:

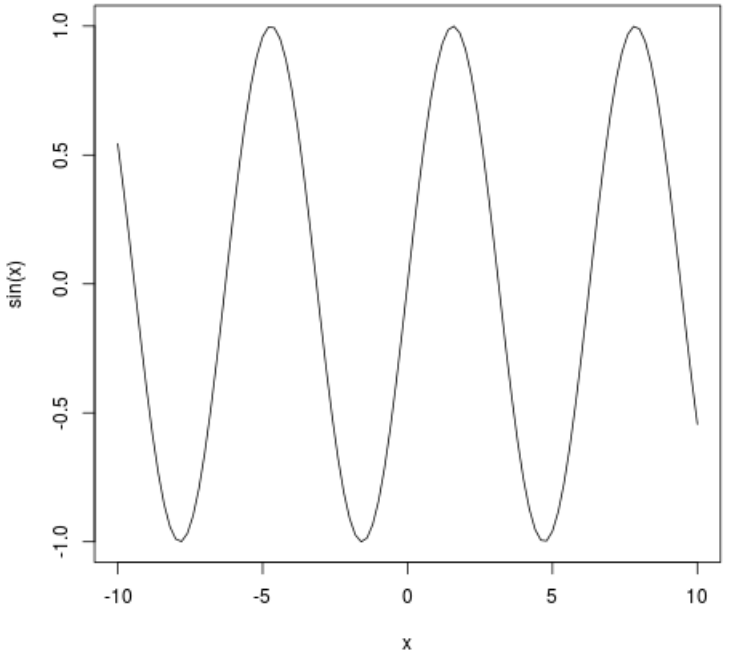
```
x <- 2

y <- x^2 + 3

y

[1] 7
```

… or to define functions:

```
myfunction <- sqrt

myfunction(81)

[1] 9
```

Printing and Plotting

```
x <- 3

print(x)

x

print(sqrt(2),digits=5)

y <- 42

cat("And the answer is ",y,".\n")

plot(sin, from = -10, to = 10)
```



Vectors

Vectors are enumerations of arbitrary objects

For example:

(2,5,3,7)

(1,4,7,10)

("green","blue","red")

("A","T","G","C")

("A","A","A","G","G")

Vectors

Vectors are enumerations of arbitrary objects

To create vectors, you can use functions in R: 'c()', 'seq()', 'rep()'

```
c(2,5,3,7)

seq(from=1,to=10,by=3)

seq(from=3,to=7)

seq(1,11,3)

seq(3,7)

seq(7,3)

3:7

c(2:5, 3:7)

rep(3,5)

rep(0:2,3)

rep(7:9,2:4)
```

Operations on vectors

You assess elements of a vector with the [ ]-Operator:

```
x <- c(12,15,13,17,11)

x[4]

[1] 17

x[3:5]

x[-2]

X[-(3:5)]
```

Standard operations on vectors are element by element:

```
c(2,5,3) + c(4,2,7)

[1] 6 7 10

2 + c(2,5,3)

c(2,5,3)^2
```

Operations on vectors

```
sum(5:7)

prod(4:6)

x <- 1:5

x[3:5]

x[-2]

x > 3

[1] FALSE FALSE FALSE TRUE TRUE
```

Useful commands on vectors:

```
length(x)

rev(x)

sort(x)

unique(x)
```

Basics – Part II

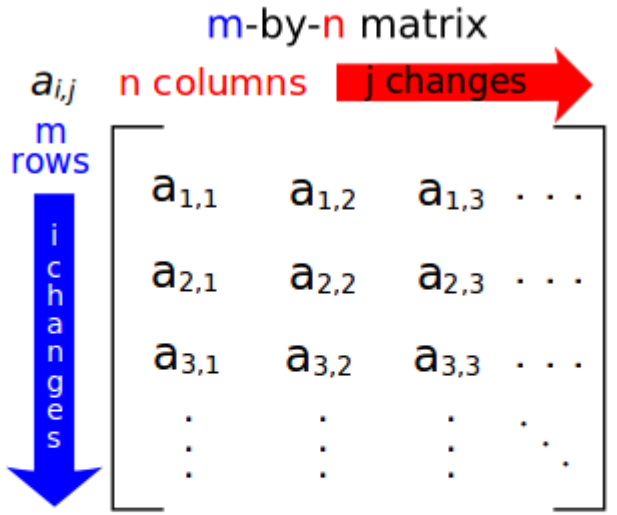
Matrix - Definition

Wikipedia:

In mathematics, a ***matrix*** (plural matrices) is a rectangular array of numbers, symbols, or expressions arranged in ***rows*** and ***columns***. The individual items in a matrix are called its ***elements*** or ***entries***. An example of a matrix with 2 rows and 3 columns is:

$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -6 \end{bmatrix}$$

2-by-3 matrix
2x3 matrix



Each element of a matrix is often denoted by a variable with two subscripts. For instance, a2,1 represents the element at the second row and first column of a matrix **A**.

Matrices - Basics

You can create matrices by:

1. matrix()
2. converting a vector into a matrix
3. binding together vectors

```
m <- matrix(data = 1:8, nrow=4, ncol=2)

m <- matrix(1:8,4,2)

z <- as.matrix(1:6)

cbind(1:3,5:7)

rbind(1:3,5:7,10:12)
```

Matrices - Basics

You can create matrices by:

1. matrix()
2. converting a vector into a matrix
3. binding together vectors

Indexing is 'row by column':

m[3,2]

m[2,]

m[,2]

m[2:3,1:2]

Matrices - Basics

Examples:

```
m <- matrix(data = 1:8, nrow = 4, ncol = 2)

m

     [,1] [,2]

[1,]    1    5

[2,]    2    6

[3,]    3    7

[4,]    4    8

m[2:3,1:2]

     [,1] [,2]

[1,]    2    6

[2,]    3    7
```

Matrices - Basics

Examples:

```
z <- as.matrix(1:6)

z

     [,1]

[1,]    1

[2,]    2

[3,]    3

[4,]    4

[5,]    5

[6,]    6
```

Matrices - Basics

Examples:

```
cbind(1:3,5:7)
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
rbind(1:3,5:7,10:12)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    5    6    7
[3,]   10   11   12
```

More in the exercises!

Functions/Commands

General form:

function()

Examples:

`sqrt()`

`exp()`

`c()`

`matrix()`

Functions can have pre-defined **parameters/arguments** with default settings

→ help page of the function

Parameters/Arguments

Example: matrix()

Which arguments can be used with this function?

?matrix()

matrix {base}

Matrices

Description

matrix creates a matrix from the given set of values.

as.matrix attempts to turn its argument into a matrix.

is.matrix tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)
```

matrix(data=(1:6),nrow=2, ncol=3)

Data types

Every variable in R has a ***class*** (e.g. matrix, list, data frame) and a ***data type*** (e.g. logical, numerical, complex, character)

Data type	Description	Examples
logical	TRUE or FALSE	TRUE, FALSE
numeric	integers and real numbers	5, -2, 3.1415, sqrt(2)
complex	complex numbers	2.1+3i, 5+0i
character	character string	“This is text”, “5”

Types can be converted:

as.logical(), as.numeric(), as.complex, as.character()

***Implicit* conversion:**

logical → numeric → complex → character

Data types

Every variable in R has a ***class*** (e.g. matrix, list, data frame) and a ***data type*** (e.g. logical, numerical, complex, character)

Data type	Description	Examples
logical	TRUE or FALSE	TRUE, FALSE
numeric	integers and real numbers	5, -2, 3.1415, sqrt(2)
complex	complex numbers	2.1+3i, 5+0i
character	character string	“This is text”, “5”

Check for data type:

is.logical(), is.numeric(), is.complex(), is.character()

```
mode()          # find out the data type
class()         # find out the class
```

Basic Statistics with R

Some distributions implemented in R

Distribution	R name
beta	beta
binomial	binom
Cauchy	cauchy
chi-square	chisq
exponential	exp
F	F
gamma	gamma
geometric	geom
hypergeometric	hyper
log-normal	lnorm
logistic	logis
multinomial	multinom
multivariate	mvnorm
normal	norm
Poisson	pois
Student's t	t
uniform	unif
distribution of the Wilcoxon rank statistic	wilcox
...	...

Some distributions implemented in R

For each distribution:

dxxx: density of the xxx distribution
pxxx: distribution function of the xxx distribution ('p' for probability)
qxxx: quantile function of the xxx distribution
rxxx: random number generator for the xxx distribution

Example: Normal distribution

dnorm(x, mean = μ, sd = ρ)

Standard normal distribution:

mean 0, standard deviation 1

dnorm(x, mean = 0, sd = 0)

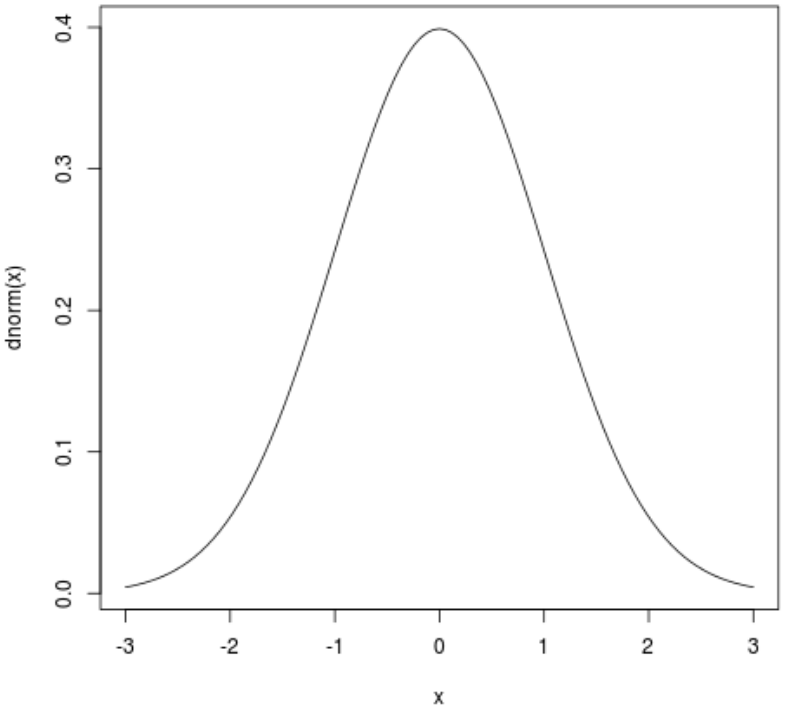
dnorm(x)

Example: Normal distribution

Recall:
plot()

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

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



Important functions

Imagine you have a vector v:

```
v <- c(1:4)

v

[1] 1 2 3 4

mean(v)

[1] 2.5

var(v)

[1] 1.666667

sd(v)

[1] 1.290994

median(v)

[1] 2.5
```

Important functions

Imagine you have a vector v:

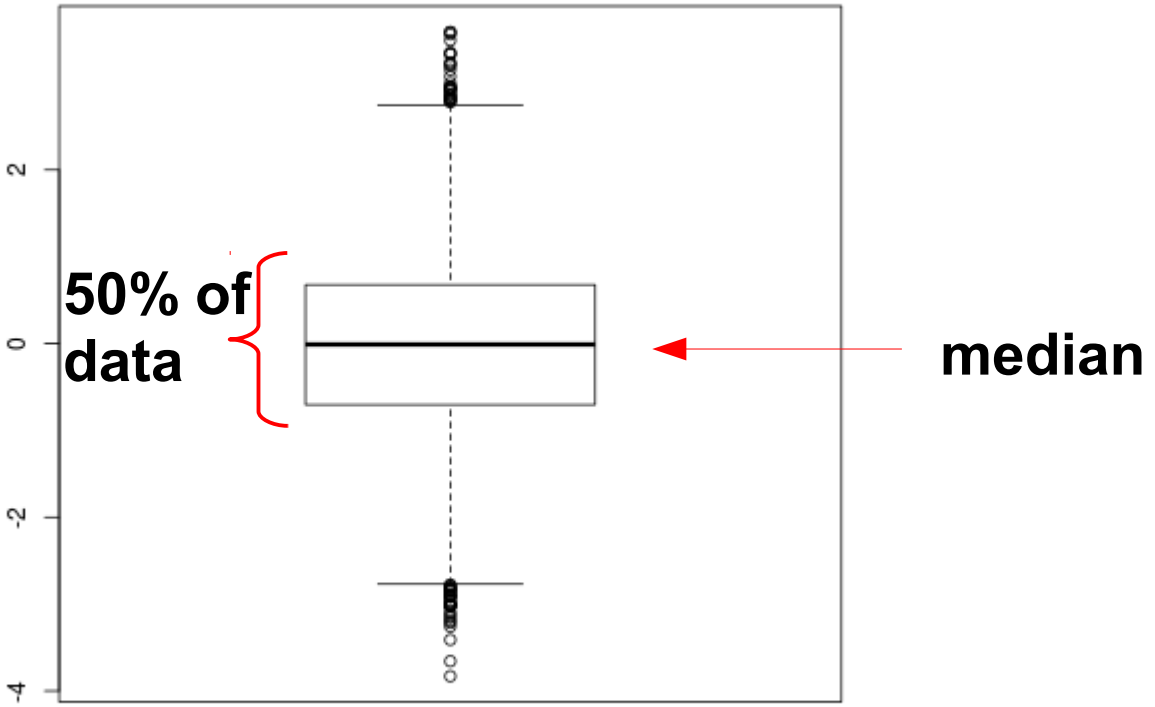
```
v <- c(1:4)

v

[1] 1 2 3 4

quantile(v)

  0%  25%  50%  75% 100%

1.00 1.75 2.50 3.25 4.00

summary(v)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.

   1.00    1.75    2.50    2.50    3.25    4.00
```

Box- and whisker plot (boxplot)

Get 10000 normally distributed values:

rnorm(10000)

boxplot(rnorm(10000))



Random numbers

Actually: Random numbers are not really random...

→ pseudo-random numbers

Reasons:

1. A normally distributed variable has a continuum of potential values – but computers only can represent a finite number of values
2. Results should be reproducible

Properties of pseudo-random numbers:

- Almost no regularities in the generated sequence
- Random sequence is reproducible
- Random sequence is generated quickly

Random numbers in R

If you want to reproduce your results: set.seed()

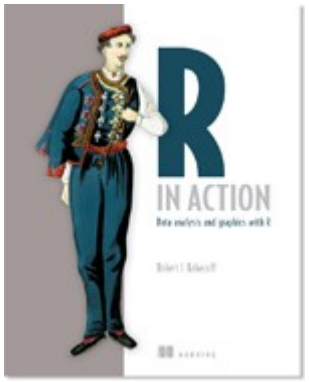
```
set.seed(1234)

rnorm(3)
```

Clarification:

The 'seed' can be every number (does not have to be '1234' – it is just used to make your results reproducible)

```
set.seed(1111)

rnorm(3)
```

A sample R session



R in Action
Data Analysis and Graphics with R
2nd edition (2011)
Robert I. Kabacoff
http://www.manning.com/kabacoff/
➔ Sample Chapter 1 (PDF)

Organize your R session

General advice:

Organize your work in folders

Save your commands in scripts (text files)

Example:

```
# Define a vector with age values

age <- c(1,3,5,2,11,9,3,9,12,3)

# Define a vector with weight values

weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1)

# Calculate the mean weight value

mean(weight)

# Quit R session

q()
```

What you should do now

If you have your own laptop or computer
1. Install R and RStudio (see tutorial on the web)
2. Read the first chapter of “R in Action” (course web page)
http://www.manning.com/kabacoff/SampleCh-01.pdf
3. Open a R session and try the commands we learned today and yesterday (lecture slides)
→ if you have trouble with installing R, ask us
4. Try to solve the exercises on **sheet01_2015.pdf**

If you don't have your own laptop or computer
1. Go to a computer room (C 00.005 or G 00.037)
2. Read the first chapter of “R in Action” (course web page)
http://www.manning.com/kabacoff/SampleCh-01.pdf
3. Open a R session and try the commands we learned
4. Try to solve the exercises on **sheet01_2015.pdf**

Keep in mind:
Programming needs a lot of practice!