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 3

Review session day 2

Reading and writing data

- Lists
- Data frames
- NA, Inf, NaN, NULL
- Editing data
- Reading and writing data frames
- Examples of different input files
- Factors

Lecture notes, pp 24-35

Solution to the exercises

Review session

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

Some tricky but very useful commands on vectors:

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

x[x>12] <- 0

x[x==0] <- 2

sum(x==2)

[1] 3

x==2

[1] FALSE TRUE TRUE TRUE FALSE

as.integer(x==2)

[1] 0 1 1 1 0
```

Operations on vectors

More useful commands (see also lecture notes, page 9):

```
v <- c(13,15,11,12,19,11,17,19)
length(v)      #returns the length of v
rev(v)         #returns the reversed vector
sort(v)        #returns the sorted vector
unique(v)      #returns vector without multiple elements
some_values <- (v > 13)
which(some_values)     #indices where 'some_values' is TRUE
which.max(v)           #index of (first) maximum
which.min(v)           #index of (first) minimum
```

Brainteaser: How can you get the indices for ALL minima?

```
all_minima <- (v == min(v))
which(all_minima)
```

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)
```

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)
```

Parameters/Arguments

Example: matrix()

Which arguments can be used with this function?

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

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

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

matrix(1:6,2,3)

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

matrix(1:6,nr=2, nc=3)

matrix(d=(1:6),nr=2, nc=3)
```

Error in matrix(d = (1:6), nr = 2, nc = 3) :

argument 1 matches multiple formal arguments

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
```

Data types
Examples

```
x <- TRUE
mode(x)
[1] "logical"

as.numeric(TRUE)
[1] 1
as.numeric(FALSE)
[1] 0

v <- c("1","2","3","4")
v
[1] "1" "2" "3" "4"
mean(v)
[1] NA
Warning message:
In mean.default(v) : argument is not numeric or logical:
returning NA
as.numeric(v)       # explicit conversion character → numeric
[1] 1 2 3 4
mean(as.numeric(v))
[1] 2.5
```

Data types
Examples

```
v <- c("1","2","3","4")
v
[1] "1" "2" "3" "4"
mean(v)
[1] NA
Warning message:
In mean.default(v) : argument is not numeric or logical:
returning NA
as.numeric(v)        # explicit conversion character → numeric
[1] 1 2 3 4
mean(as.numeric(v))
[1] 2.5

# implicit conversion character → numeric does not work
2*v
Error in 2 * v : non-numeric argument to binary operator

# implicit conversion numeric → character works
z <- c(1,2,"3","4")
z
[1] "1" "2" "3" "4"
```

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)

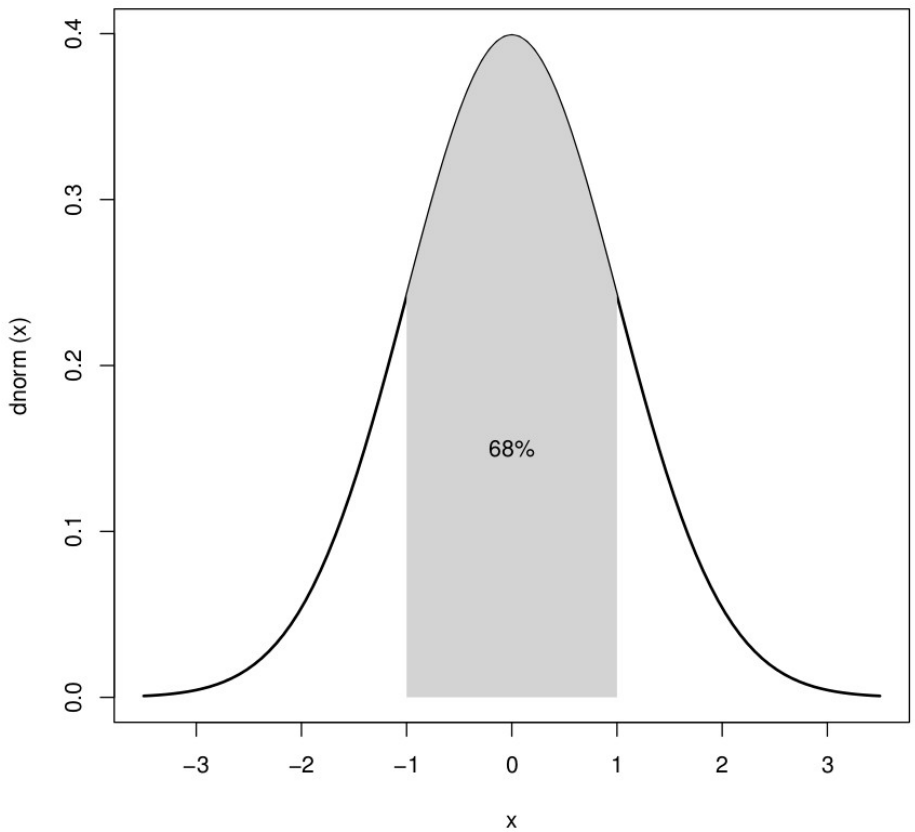
Standard normal distribution:
mean 0, standard deviation 1

```
dnorm(x, mean = 0, sd = 1)
dnorm(x)
```

Normal distribution
some useful facts

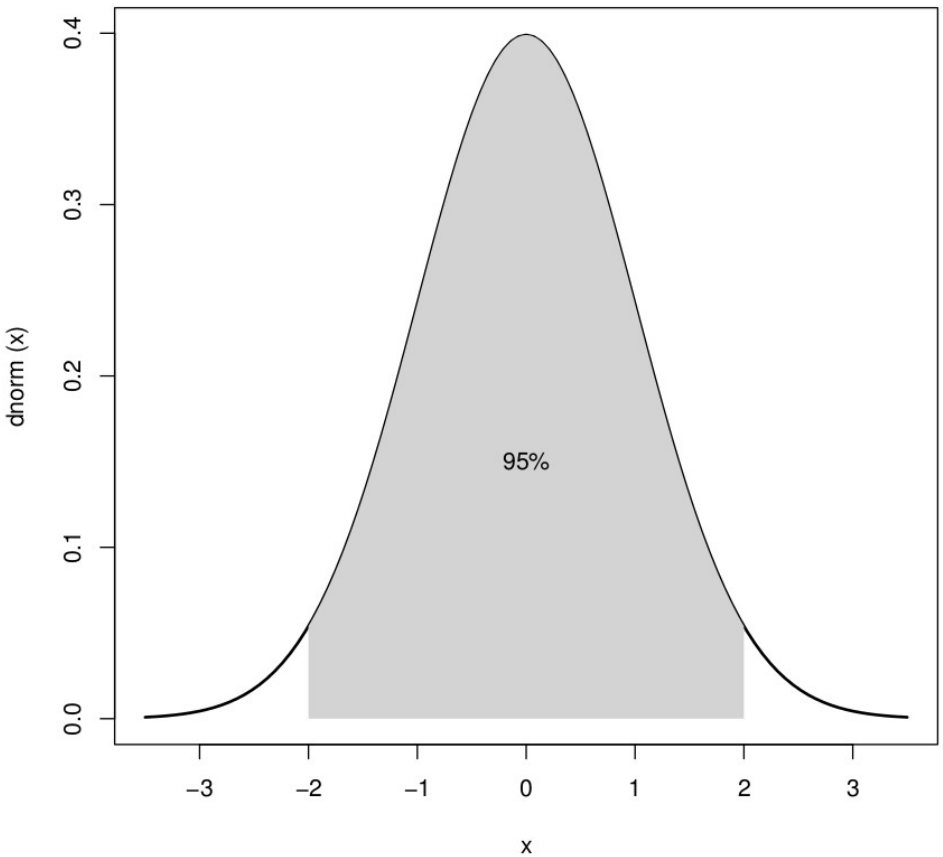
~68% of the mass of a standard normal distribution is within **one** standard deviation

```
pnorm(1) – pnorm(-1)
[1] 0.6826895
```



~95% of the mass of a standard normal distribution is within **two** standard deviations

```
pnorm(2) – pnorm(-2)
[1] 0.9544997
```



Reading and writing data

What is a data frame?

- Object with rows and columns
- Rows: different observations, measurements
- Columns: different variables

IMPORTANT:

All values of the same variable MUST go in the same column

Example: Data of expression study

3 groups/treatments: Control, Tropics, Temperate

4 measurements per treatment

Control	Tropical	Temperate
6.1	6.3	7.1
5.9	6.2	8.2
5.8	5.8	7.3
5.4	6.3	6.9

NOT a data frame!

Same data as data frame

Response	Treatment
6.1	control
5.9	control
5.8	control
5.4	control
6.3	tropical
6.2	tropical
5.8	tropical
6.3	tropical
7.1	temperate
8.2	temperate
7.3	temperate
6.9	temperate

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(): which()

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

Alternative to attach(): which()

```
with(mtcars, {
   summary(mpg)
   mean(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 tomorrow (pp 26-29)

Steps:

1) Define your working directory

2) Read in data (from *data.txt*)

3) Check your data

4) Copy data into search path

5) Select subsets of your data

6) Split your data into a list of a subgroup

7) Extend your data frame

8) Remove data from search path

NA, Inf, NaN, NULL

NA = not available

Inf = Infinity

NaN = Not a Number

Important command: is.na()

Examples:

```
v <- c(1,3,NA,5)

v[1] <- NA

is.na(v)

[1] TRUE FALSE TRUE FALSE
```

Ignore missing data: 'na.rm=TRUE'

```
sum(v, na.rm=TRUE)

[1] 8
```

Reading and writing data frames
more options

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")
```

Command	header	sep	dec	fill
read.table()	FALSE	`""`	`"."`	FALSE
read.csv()	TRUE	`","`	`"."`	TRUE
read.csv2()	TRUE	`";"`	`","`	TRUE
read.delim()	TRUE	`"\t"`	`"."`	TRUE
read.delim2()	TRUE	`"\t"`	`","`	TRUE

Exercise sheet 1