An introduction to R
WS 2013/2014

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

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

Course outline – Day 3

Review session

- Vectors (+ some more useful commands)
- Data frames
- How to get data into R

Reading and writing data

- More on data frames
- NA, Inf, NaN, NULL

Basic statistics with R

- Some distributions
- Example: Normal distribution
- Important functions
- Random numbers

Solutions to exercise sheet 1 (clarification)

- Exercise 5 – signif, exp(), expm1()

Review Session

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 could you get the indices for ALL minima?

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

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	hermaphrodite	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", "hermaphrodite", "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)

(Default) parameters for read.table() (page 31):

read.table(file, header = FALSE, sep = “”, dec = “.”, row.names, fill = FALSE, ...)

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)
```

Reading and writing data

More on data frames

We will work through the example from the lecture notes

(pp 25-29)

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
```

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

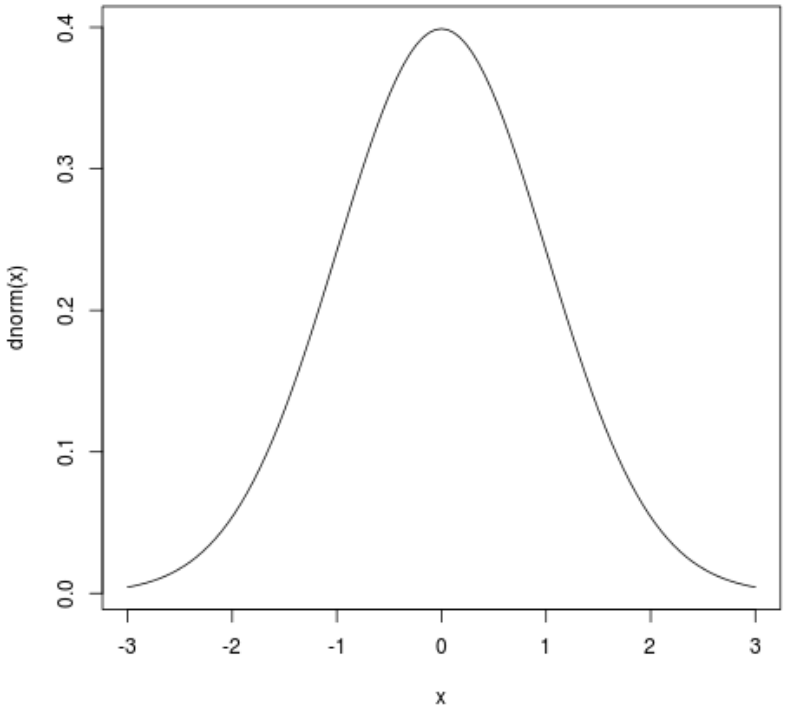
```
>dnorm(x, mean = 0, sd = 0)
>dnorm(x)
```

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:

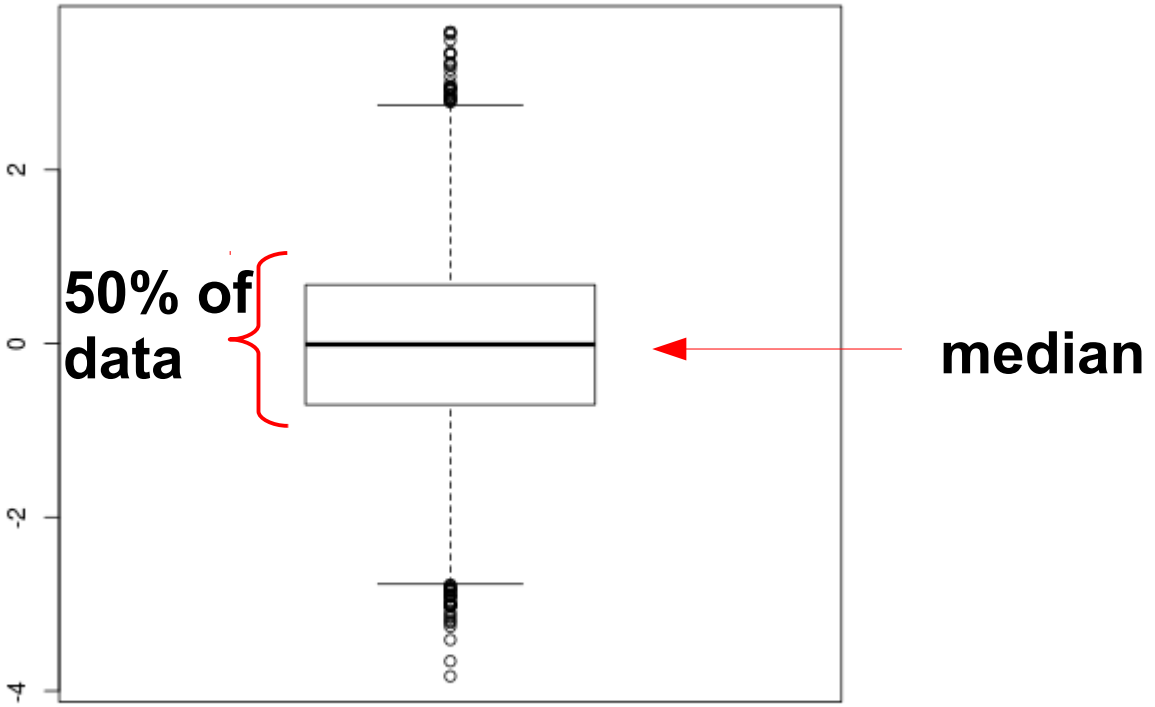
```
> v <- c(1:4)
> v
[1] 1 2 3 4
> quantile(v)
  0%  25%  50%  75% 100%
1.00 1.75 2.50 3.25 4.00
> summary(v)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.75    2.50    2.50    3.25    4.00
```

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)
```