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 2

Review session

- Assignments
- Functions/Commands
- Parameters/Arguments
- Vectors/Matrices (→ exercise solutions)

Reading and writing data

- Libraries/Packages
- Data frames
- Reading and writing data frames

Printing and Plotting

- Basics and outlook on day 3

Solutions to exercise sheet 1

Review Session

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
```

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)
```

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

Argument 1 passt auf mehrere formale Argumente

Reading and writing data

Libraries

R commands are organized in packages (libraries)

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()
```

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	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)
)
```

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

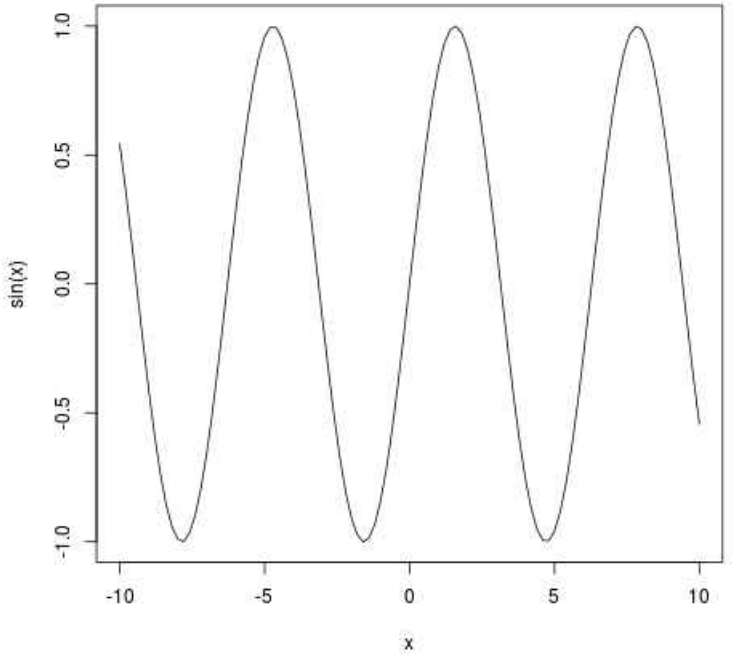
Workflow - Script

```
#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)
```

Printing and Plotting

Printing and Plotting

```
>x <- 3:7
>print(x)
>x
>print(sqrt(2),digits=5)
>y <- 42
>cat("And the answer is ",y,".\n")
>plot(sin, from=-10, to=10)
```



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

Advantage:
You can basically do any kind of graphic

Disadvantage
You can basically do any kind of graphic...
→ many, many parameters and packages...

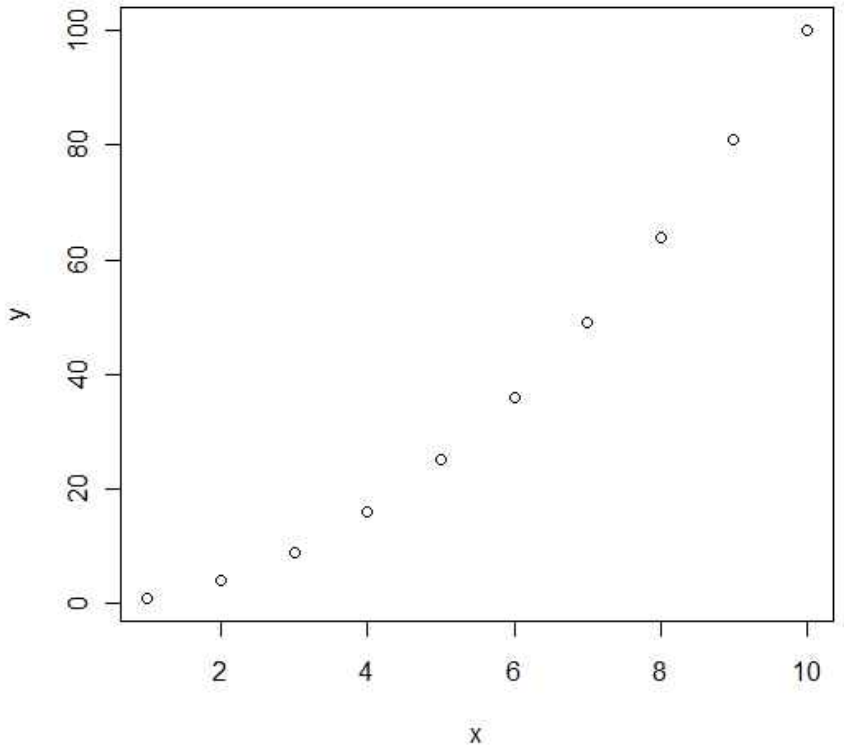
High-level plotting functions

Standard high-level plotting function: plot()

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

Example:

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



High-level plotting functions

Standard high-level plotting function: plot()

plot(*fun*)

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

Example:

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