R course for beginners

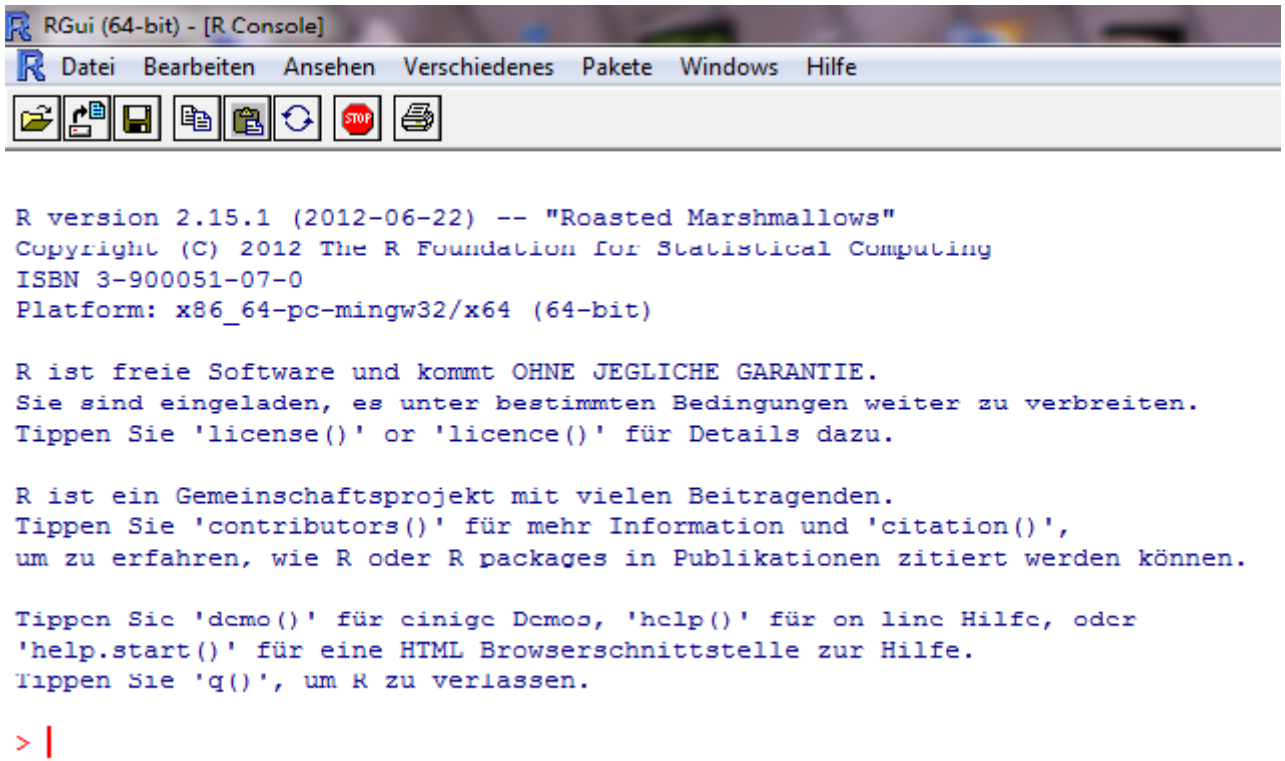
Session 1:
Read a file, manipulate its contents and save your results

Christine Gläßer ---- ZMBH ---- Room 504 ---- +49(0)6221-54 6824 ---- c.glaesser@zmbh.uni-heidelberg.de

Where are we?

- Your data is stored in a folder called „R_basic“

- You start your R session



Where are we?

- Find out where you are located

command: getwd()
Getwd stands for „get working directory“

Where are we?

- Find out where you are located

command: getwd()
Getwd stands for „get working directory“

() in R denotes a function

Function: Process doing a specific task

Where are we?

- Find out where you are located

command: getwd()
Getwd stands for „get working directory“

- You want to work in your folder „R_basic“, since it'll be easier to read and save your data

Command: setwd()
Setwd stands for „set working directory“

Where are we?

- Find out where you are located

command: getwd()
Getwd stands for „get working directory“

- You want to work in your folder „R_basic“, since it'll be easier to read and save your data

Command: setwd()
Setwd stands for „set working directory“

- What files are stored in „R_basic“?

Command: dir()

Reading in a file

Load data „sleep_data_simple.txt“

It is a sample data set, describing a sleep experiment

---> group X had 8 hours of sleep
---> group Y had 4 hours of sleep
---> both groups were tested for their cognitive abilites directly after sleep

We will use this dataset later for a simple t-test, now only for reading-in

Reading in a file

Command: read.delim(INFILE)

INFILE stands for the file you'd like to read in (in our case: sleep_data_simple.txt)

Many other commands for reading files...

…. read.csv: read a csv/text file with semicolon separations
…. read.delim: read a text file containing tab separations
…. read.table: read a text file with white space separations

---> check **?read.table**
---> „?“ before a command always directs you to the documentation of that specific command

Command

Package

```
read.table                package:utils                R Documentation

Data Input

Description:

     Reads a file in table format and creates a data frame from it,
     with cases corresponding to lines and variables to fields in the
     file.

Usage:

     read.table(file, header = FALSE, sep = "", quote = "\"'",
                dec = ".", row.names, col.names,
                as.is = !stringsAsFactors,
                na.strings = "NA", colClasses = NA, nrows = -1,
                skip = 0, check.names = TRUE, fill = !blank.lines.skip,
                strip.white = FALSE, blank.lines.skip = TRUE,
                comment.char = "#",
                allowEscapes = FALSE, flush = FALSE,
                stringsAsFactors = default.stringsAsFactors(),
                fileEncoding = "", encoding = "unknown", text)

     read.csv(file, header = TRUE, sep = ",", quote = "\"",
              dec = ".", fill = TRUE, comment.char = "", ...)

     read.csv2(file, header = TRUE, sep = ";", quote = "\"",
               dec = ",", fill = TRUE, comment.char = "", ...)

     read.delim(file, header = TRUE, sep = "\t", quote = "\"",
                dec = ".", fill = TRUE, comment.char = "", ...)

     read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
                 dec = ",", fill = TRUE, comment.char = "", ...)

Arguments:

    file: the name of the file which the data are to be read from.
          Each row of the table appears as one line of the file.  If it
          does not contain an _absolute_ path, the file name is
          _relative_ to the current working directory, 'getwd()'.
          Tilde-expansion is performed where supported.  This can be a
          compressed file (see 'file').

          Alternatively, 'file' can be a readable text-mode connection
          (which will be opened for reading if necessary, and if so
          'close'd (and hence destroyed) at the end of the function
          call).  (If 'stdin()' is used, the prompts for lines may be
          somewhat confusing.  Terminate input with a blank line or an
          EOF signal, 'Ctrl-D' on Unix and 'Ctrl-Z' on Windows.  Any
          pushback on 'stdin()' will be cleared before return.)

          'file' can also be a complete URL.  (For the supported URL
          schemes, see the 'URLs' section of the help for 'url'.)
:
```

Usage with default settings

Description of arguments

Ready to run examples

##: comment, will be ignored if copy-pasted to R command line

```
Examples:

    ## using count.fields to handle unknown maximum number of fields
    ## when fill = TRUE
    test1 <- c(1:5, "6,7", "8,9,10")
    tf <- tempfile()
    writeLines(test1, tf)

    read.csv(tf, fill = TRUE) # 1 column
    ncol <- max(count.fields(tf, sep = ","))
    read.csv(tf, fill = TRUE, header = FALSE,
             col.names = paste0("V", seq_len(ncol)))
    unlink(tf)

    ## "Inline" data set, using text=
    ## Notice that leading and trailing empty lines are auto-trimmed

    read.table(header = TRUE, text = "
    a b
    1 2
    3 4
    ")
```

Reading in a file

Is it possible to give a specific name to the read-in content?

Reading in a file

Is it possible to give a specific name to the read-in content?

This name is a variable, representing the content our our file
We need that, since otherwise the content won't be accessible in R
It would simply vanish somewhere in our storage in a location labelled with a name we don't know

A variable is a place holder for the content it contains

Reading in a file

Is it possible to give a specific name to the read-in content?

Command: sleepdata <- read.delim(„sleep_data_simple.txt“)

Alternative: sleepdata = read.delim(„sleep_data_simple.txt“)

The arrow and the equal both direct the file's content to the variable sleepdata
You can use both, but in the R community, there's an „operator war“ going on...

Reading in a file

Is it possible to give a specific

Why quotation marks?

The file name is a string, a sequence of characters
If it is something else (e.g. a variable like sleepdata),
you don't need the quotation marks

Command: sleepdata <- read.delim(„sleep_data_simple.txt“)

Alternative: sleepdata = read.delim(„sleep_data_simple.txt“)

The arrow and the equal both direct the file's content to the variable sleepdata
You can use both, but in the R community, there's an „operator war“ going on...

Reading in a file – reading it's contents

a) Look at the data

sleepdata

Reading in a file – reading it's contents

a) Look at the data

sleepdata

b) What is the third value of group Y?

Sleepdata[1,1]
Row Column

sleepdata[3,2]
----> sleepdata[,2] will give you only Y as list

Reading in a file – reading it's contents

a) Look at the data

sleepdata

b) What is the third value of group Y?

Sleepdata[1,1]
Row Column

sleepdata[3,2]
----> sleepdata[,2] will give you only Y as list

c) What are the column's names? What are the row's names?

Command: colnames(sleepdata)
rownames(sleepdata)

Reading in a file – adding a line

Sleepdata has two columns, eight rows:

```
> sleepdata
  X8.hours.sleep.group..X. X4.hours.sleep.group..Y.
1                        5                        8
2                        7                        1
3                        5                        4
4                        3                        6
5                        5                        6
6                        3                        4
7                        3                        1
8                        9                        2
>
```

We'd like to add the following data:

For group X: 1
For group Y: 2

Command: rbind
Rbind stands for „row bind“

Reading in a file – adding a line

Build an extended matrix:

sleepdata.add <- rbind(sleepdata, 0)

```
> sleepdata
  X8.hours.sleep.group..X. X4.hours.sleep.group..Y.
1                        5                        8
2                        7                        1
3                        5                        4
4                        3                        6
5                        5                        6
6                        3                        4
7                        3                        1
8                        9                        2
> sleepdata.add<- rbind(sleepdata, 0)
> sleepdata.add
  X8.hours.sleep.group..X. X4.hours.sleep.group..Y.
1                        5                        8
2                        7                        1
3                        5                        4
4                        3                        6
5                        5                        6
6                        3                        4
7                        3                        1
8                        9                        2
9                        0                        0
> 
```

Reading in a file – adding a line

Now assign values „1“ to group X and „2“ to group Y

e.g. **sleepdata.add[9,1] <- 1**

New matrix:

```
> sleepdata.add
  X8.hours.sleep.group..X. X4.hours.sleep.group..Y.
1                        5                        8
2                        7                        1
3                        5                        4
4                        3                        6
5                        5                        6
6                        3                        4
7                        3                        1
8                        9                        2
9                        1                        2
>
```

Basic mathematical operations

a) Addition

Adding <- 1+2

What is the sum of group X + group Y per row?

Sumxy <- sleepdata.add[,1]+sleepdata.add[,2]

What is the total sum of X + Y?

Command: sum

Check **?sum**

Basic mathematical operations

b) Subtraction

Subtraction <- 1-2

What is group Y, position 2, minus group X, position 3?

Basic mathematical operations

b) Subtraction

Subtraction <- 1-2

What is group Y, position 2, minus group X, position 3?

c) Multiplication

Multiplication <- 2*3

d) Division

Division <- 2/3

Basic mathematical operations

e) What is the mean value of group X in sleepdata.add?

Command: mean

Check **?mean**

meanx <- mean(sleepdata.add[,1])

Basic mathematical operations

e) What is the mean value of group X in sleepdata.add?

Command: mean

Check **?mean**

meanx <- mean(sleepdata.add[,1])

f) What is the standard deviation of group Y in sleepdata.add?

Command: sd

Check **?sd**

stdy <- sd(sleepdata.add[,2])

Writing a file

Command: write.table()

```
write.table                package:utils                R Documentation

Data Output

Description:

     'write.table' prints its required argument 'x' (after converting
     it to a data frame if it is not one nor a matrix) to a file or
     connection.

Usage:

     write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
                 eol = "\n", na = "NA", dec = ".", row.names = TRUE,
                 col.names = TRUE, qmethod = c("escape", "double"),
                 fileEncoding = "")

     write.csv(...)
     write.csv2(...)

Arguments:
```

„x“ is the data we'd like to write; file=““““ stands for the file we'd like to create
Sep= „ “ means there's a space separation... we'd like to have a tab-separated file!

Writing a file

Command: write.table()

```
write.table                package:utils                R Documentation

Data Output

Description:

     'write.table' prints its required argument 'x' (after converting
     it to a data frame if it is not one nor a matrix) to a file or
     connection.

Usage:

     write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
                 eol = "\n", na = "NA", dec = ".", row.names = TRUE,
                 col.names = TRUE, qmethod = c("escape", "double"),
                 fileEncoding = "")

     write.csv(...)
     write.csv2(...)

Arguments:
```

A tab as separator can be created using the escape sequence „\t“

So, our command will be:

Writing a file

Command: write.table()

```
write.table                package:utils                R Documentation

Data Output

Description:

     'write.table' pri
     it to a data fra
     connection.

Usage:

     write.table(x, f
                 eol =
                 col.n
                 fileE

     write.csv(...)
     write.csv2(...)

Arguments:
```

Escape sequences

use an escape character to change
the meaning of the characters which follow
it, meaning that the characters can be interpreted
as a command to be executed rather than as data.

A tab as separator can be created using the escape sequence „\t“

So, our command will be:

Writing a file

write.table(sleepdata.add, „sleepdata.add_out.txt“, sep=“\t“)

However, output still has the row numbering:

```
"X8.hours.sleep.group..X."    "X4.hours.sleep.group..Y."
"1" 5    8
"2" 7    1
"3" 5    4
"4" 3    6
"5" 5    6
"6" 3    4
"7" 3    1
"8" 9    2
"9" 1    2
```

And why „“??? How to avoid the nasty line numbering being printed out???

Answers after lunch!

http://weihrausch.files.wordpress.com/2012/01/fotolia_29039500_s.jpg