UNIVERSITÄT LEIPZIG

European Research Council

Introduction to R

Natalia Levshina © 2018

Leipzig University

December 4 2018, Ghent

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R

What is R?

- statistical computing environment (from *t*-test to generalized linear models, and more…)
    - core distribution “base”
    - add-on packages (> 12K as of March 2017)
- programming language
- tools for creation of publication-quality plots (e.g. ggplot2)

Where to get R?

- Distribution and packages: CRAN (Comprehensive R Archive Network) http://cran.r-project.org/
- Information: http://www.r-project.org/

RStudio

- Highly recommended (easy to manage projects, packages, data, graphs, etc.)!
- Available from http://www.rstudio.com/products/RStudio/

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R

Input and output

```
2 + 2
```

```
[1] 4
```

```
sample(100, 25) #random sampling of 25 elements
from integers 1 to 100
```

```
[1] 49 45 70 51 54  5  7 19 60 82 35 55  6 76 93
89 44
```

```
[18]  8 48 87 53 34 86 96 63
```

Basic arithmetic functions

```
25^2
[1] 625
625^0.5
[1] 25
sqrt(625)
[1] 25
log(5)
[1] 1.609438
```

Creation of objects

```
a <- 3
a
[1] 3
a + 5
[1] 8
```

Exercise

- Create two numeric vectors with 1 element in each:
  a) the population of Ghent
  b) the population of Bruges
- Compute their sum.
- Compute their difference.
- By how many times is the population of Ghent larger than that of Bruges?

Beware: = and ==

```
a = 3 # creates an object a with the value 3, an alternative to "a <- 3"
a == 3 # tests if a equals 3
[1] TRUE
a == 10 # tests if a equals 10
[1] FALSE
```

Exercise

- Perform an R test whether the population of Ghent is equal to that of Bruges, using the vectors.

R is case-sensitive!

```
b <- 7
a + b
[1] 10
a + B
Error: object 'B' not found
```

Managing your objects

```
ls() #returns a list of objects
 [1] "a"         "b"

rm(b) #removes an object
ls()
 [1] "a"
```

Saving your workspace

Click on the cross button or type

```
q()
```

Select the action (to save or not to save).

```
getwd() #to find out where your workspace
will be saved
[1] "C:/Users/Your/Directory"

setwd("C:/Users/Your/Directory") #to change
it, if you like
```

Getting help

```
?cor #to open a help file with information about function 'cor'


??correlation #returns a list of functions that contain this expression
```

Exercise

- Get help on the function summary().

Errors

```
x <- 1:10 # creates a numeric vector with integers from 1 to 10

x

 [1]  1  2  3  4  5  6  7  8  9 10

meann(x) # we want to compute the mean value of x: a typo

Error: could not find function "meann"

mean(x) # correct

[1] 5.5
```

Warning messages

```
mytable <- rbind(c(1, 2), c(3, 4)) #create a 2-by-2 table
mytable
     [,1] [,2]
[1,]    1    2
[2,]    3    4
chisq.test(mytable)
Pearson's Chi-squared test with Yates' continuity correction
data: mytable
X-squared = 0, df = 1, p-value = 1
Warning message:
In chisq.test(mytable) : Chi-squared approximation may be incorrect
```

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R

Important data types in R

- Numeric vectors
- Character vectors
- Factors
- Data frames
- Contingency tables
- Matrices

Numeric vectors

```
vnum <- 1:5 # a vector of integers from 1 to 5
vnum
[1] 1 2 3 4 5
is(vnum)
 [1] "integer"                    "numeric"
"vector"
```

[….]

If not a sequence:

```
RT <- c(455, 773, 512, 667) #reaction times in an
experiment
RT
[1] 455 773 512 667
```

Character vectors

```
sex <- c("f", "m", "m", "f")
sex
[1] "f" "m" "m" "f"
is(sex)
[1] "character"              "vector"
[…]
```

Matrices

```
m <- cbind(1:5, 10:6)
m
     [,1] [,2]
[1,]    1   10
[2,]    2    9
[3,]    3    8
[4,]    4    7
[5,]    5    6
is(m)
[1] "matrix"    "array"   […]
```

Factors

```
sex.f <- factor(sex)
sex.f
[1] f m m f
Levels: f m

is(sex.f)
 [1] "factor"                    "integer"
[…]
```

Data frames

```
mydf <- data.frame(sex, RT) #char. vectors turn into factors
mydf
  sex  RT
1   f 455
2   m 773
3   m 512
4   f 667
is(mydf)
[1] "data.frame" "list" […]
```

Exercise

1. Create a character vector with the names of your fellow students.
2. Create a vector with their heights (in cm).
3. Combine the vectors in one data frame.

Contingency tables

- Let's add another factor to the dataframe, *dialect*:

```
mydf$dialect <- c("BrE", "AmE", "AmE", "BrE")
mydf
  sex          RT     dialect
1   f  455            BrE
2   m  773            AmE
3   m  512            AmE
4   f  667            BrE
table(mydf$sex, mydf$dialect)

     AmE BrE
  f    0      2
  m    2      0
```

Exercise

1. Add a factor to your data frame with the answers to the question, “Do you like beer?” (“Yes” or “No”)

2. Add another factor with the gender (“m” or “f”, or …)

3. Cross-tabulate the factors. Do you think there’s a gender bias?

Summarizing the data

```
summary(mydf)
 sex          RT           dialect
 f:2   Min.   :455.0   Length:4
 m:2   1st Qu.:497.8   Class :character
       Median :589.5   Mode  :character
       Mean   :601.8
       3rd Qu.:693.5
       Max.   :773.0
str(mydf)
'data.frame': 4 obs. of  3 variables:
 $ sex    : Factor w/ 2 levels "f","m": 1 2 2 1
 $ RT     : num  455 773 512 667
 $ dialect: chr  "BrE" "AmE" "AmE" "BrE"
```

Selecting observations

```
mydf[1,]
  sex  rt   dialect      #the fist row
1   f 455   BrE

mydf[,2]
[1] 455 773 512 667 #the second column

mydf[1,2]
[1] 455 #the element in the fist row,  second
column
```

Using logical operators

```
mydf[mydf$sex == "f",]
  sex  RT   dialect
1   f 455   BrE
4   f 667   BrE
mydf[mydf$sex != "m", ]
  sex  RT   dialect
1   f 455   BrE
4   f 667   BrE
mydf[mydf$RT < 500,]
  sex  RT   dialect
1   f 455   BrE
```

Exercise

- Make a subset of your data frame with all colleagues taller than 170 cm.
- How many rows (students) does the data frame contain?

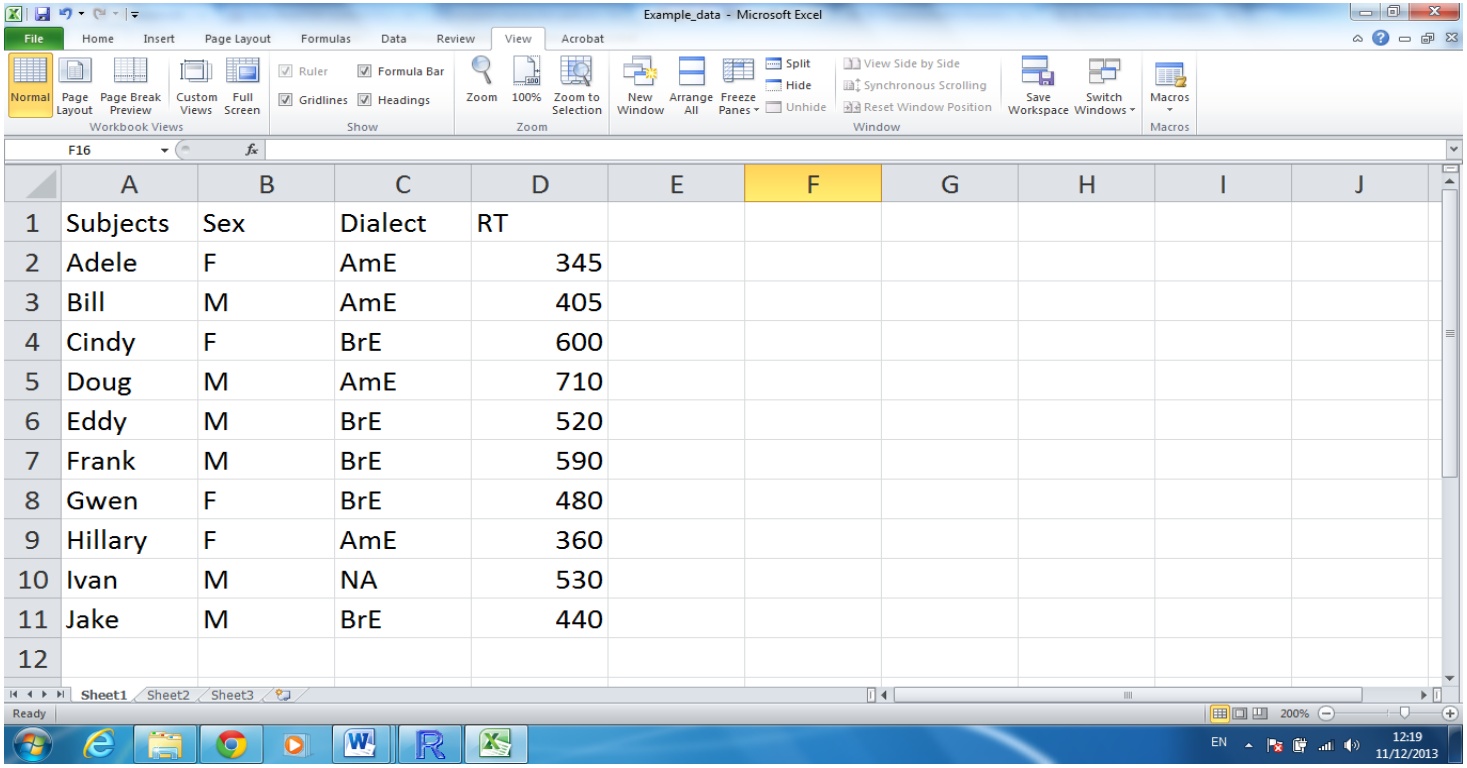
A quest

1. Compute the square root of 1681.
2. Type in R: set.seed(x), where x is the result of step 1.
3. Create a random sample of 100 numbers from 1 to 100.
4. Find the 20^{th} element. This will be your y.
5. Take the yth letter in the English alphabet. Write down the letter.
6. Open the help page of the function read.table and find the subsection “See also”. Find the first R function mentioned in that subsection. Remove the first letter and write down the result.
7. Find R citation information using citation(). Take the 3^{rd} word and write down the letter.
8. Put all words together!

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R

Importing your data into R



	A	B	C	D	E	F	G	H	I	J
1	Subjects	Sex	Dialect	RT						
2	Adele	F	AmE	345						
3	Bill	M	AmE	405						
4	Cindy	F	BrE	600						
5	Doug	M	AmE	710						
6	Eddy	M	BrE	520						
7	Frank	M	BrE	590						
8	Gwen	F	BrE	480						
9	Hillary	F	AmE	360						
10	Ivan	M	NA	530						
11	Jake	M	BrE	440						
12										

Importing your data into R

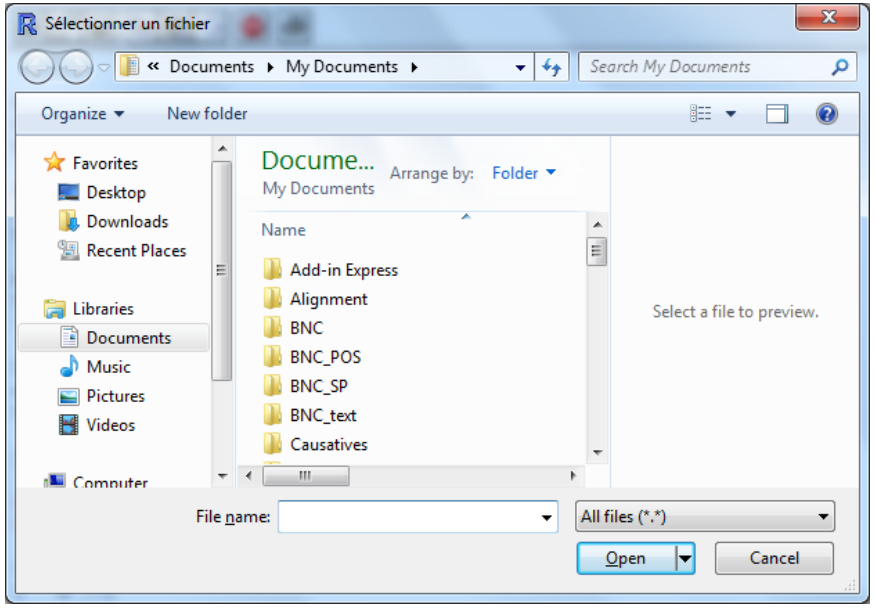
1. Create a similar table in Excel (or OpenOffice Calc). Don't forget to create a header. In case of missing values, put NA. No empty cells!

2. Save the file as a tab delimited text file (.txt).

3. Read the file in R:

```
mydata <- read.table(file = file.choose(), header = TRUE)
```

Interactive choice



Exercise

Create the following table in Excel (or OpenOffice Calc) and import it in R as a data frame under the name *Linguists*.

Last name	**First name**	**Framework**	**Born**	**Died**
de Saussure	Ferdinand	Structuralism	1857	1913
Chomsky	Noam	Generative Linguistics	1928	NA
Lakoff	George	Cognitive Linguistics	1941	NA

A tip

- If you have a table with white spaces, apostrophes, etc., use this bullet-proof code:

```
mydata <- read.table(file = file.choose(),
header = TRUE, sep = "\t", comment = "",
quote = "")
```

Exercise

- Open a fragment of a Universal Dependencies corpus (ud_sample) as a table.
- Make a subset with all common nouns (column: upos, value: NOUN).
- Make a subset with all subjects (column: dep_rel, value: nsubj)
- Cross-tabulate all parts of speech with all syntactic functions (dependencies).