UNIVERSITÄT LEIPZIG

erc European Research Council

Introduction to R and first steps in categorical data analysis

Natalia Levshina © 2017

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R
5. First inspection of data
6. Work with ParTy

What is R?

- statistical computing environment (from *t*-test to generalized linear models, and more…)

  - core distribution “base”

  - add-on packages (> 10K as of June 2017)

- programming language
- tools for creation of publication-quality plots

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
5. First inspection of data
6. Work with ParTy

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

- The population of Finland is approximately 5.5M people, and the population of Burkina Faso is about 17.3M. Create two R numeric vectors with 1 element in each, named Finland and Burkina_Faso, with the corresponding population values. Compute their sum.

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
```

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

1. Click on the cross 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
```

2. Next session: restart R or, if you have many different workspaces, click on the R from the directory; alternatively:

```
> load("yourDirectory/yourFile.RData")
```

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
5. First inspection of data
6. Work with ParTy

Important data types in R

- Numeric vectors
- Character vectors
- Factors
- Data frames
- Contingency tables
- Matrices
- Distance 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"
[…]
```

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

Create a character vector with the names of your fellow students. Create a vector with their heights (in cm). 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
```

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"   […]
```

Distance matrices

```
> eurodist
```

[output omitted: distances between several European cities]

My journey yesterday:

	Frankfurt	Stockholm	Tampere
Frankfurt	0	1186	1572
Stockholm	1186	0	395
Tampere	1572	395	0

My journey

```
> mydist <- rbind(Frankfurt = c(0, 1186, 1572),
Stockholm = c(1186, 0, 395), Tampere = c(1572,
395, 0))

> colnames(mydist) <- rownames(mydist)

> mydist

            Frankfurt Stockholm Tampere
Frankfurt         0      1186    1572
Stockholm      1186         0     395
Tampere        1572       395       0

> is(mydist)

[1] "matrix"    "array"     "mMatrix"
"structure" "vector"
```

From matrix to distance matrix

```
> mydist <- as.dist(mydist)
> mydist
              Frankfurt Stockholm
Stockholm      1186
Tampere        1572       395
> is(mydist)
[1] "dist"
```

… and back

```
> m <- as.matrix(mydist)
> m
            Frankfurt Stockholm Tampere
Frankfurt         0      1186    1572
Stockholm      1186         0     395
Tampere        1572       395       0
```

Exercise

- Make your own distance matrix, depending on where you have travelled from.

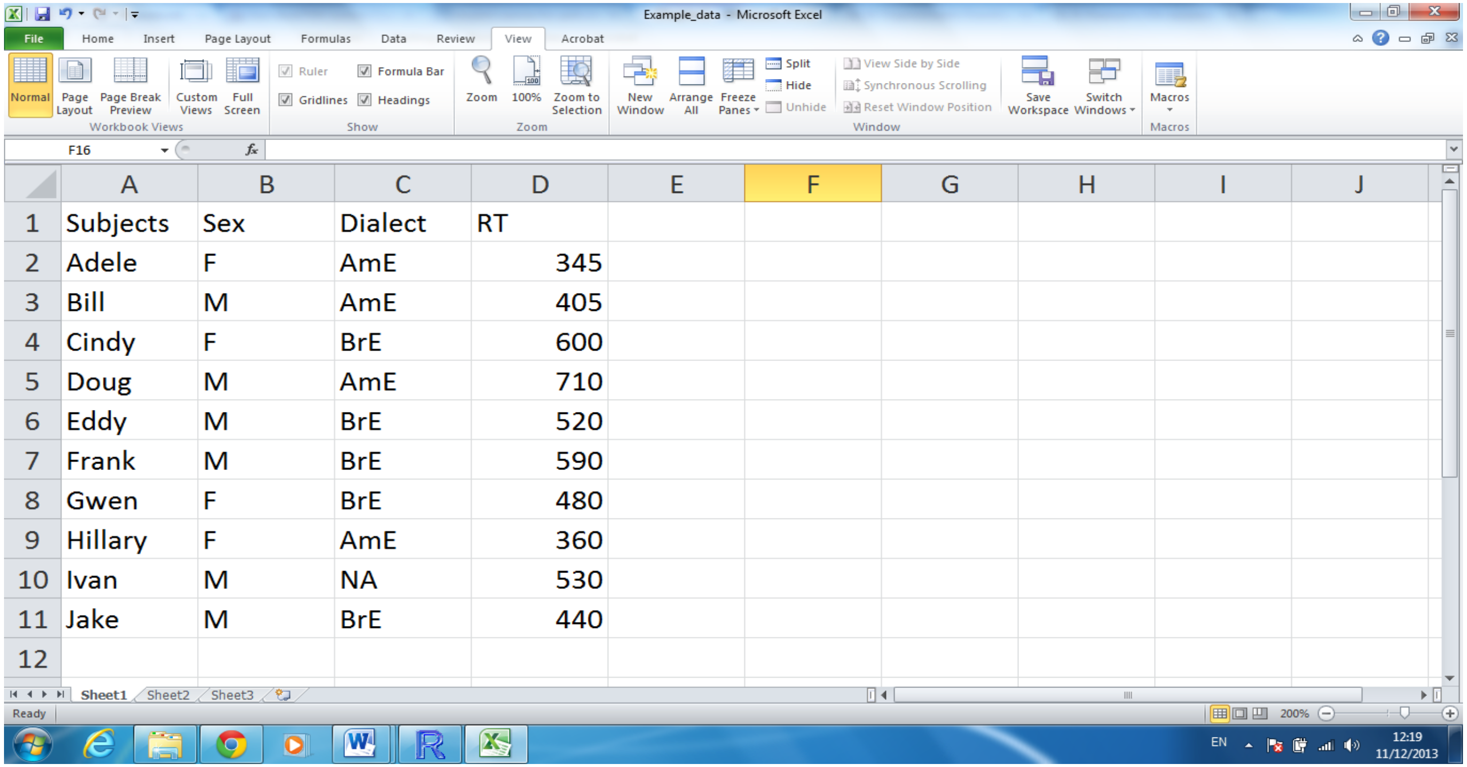
Quest

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 20th 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 3rd 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
5. First inspection of data
6. Work with ParTy

Importing your data to R



Subjects	Sex	Dialect	RT
Adele	F	AmE	345
Bill	M	AmE	405
Cindy	F	BrE	600
Doug	M	AmE	710
Eddy	M	BrE	520
Frank	M	BrE	590
Gwen	F	BrE	480
Hillary	F	AmE	360
Ivan	M	NA	530
Jake	M	BrE	440

Importing your data into R

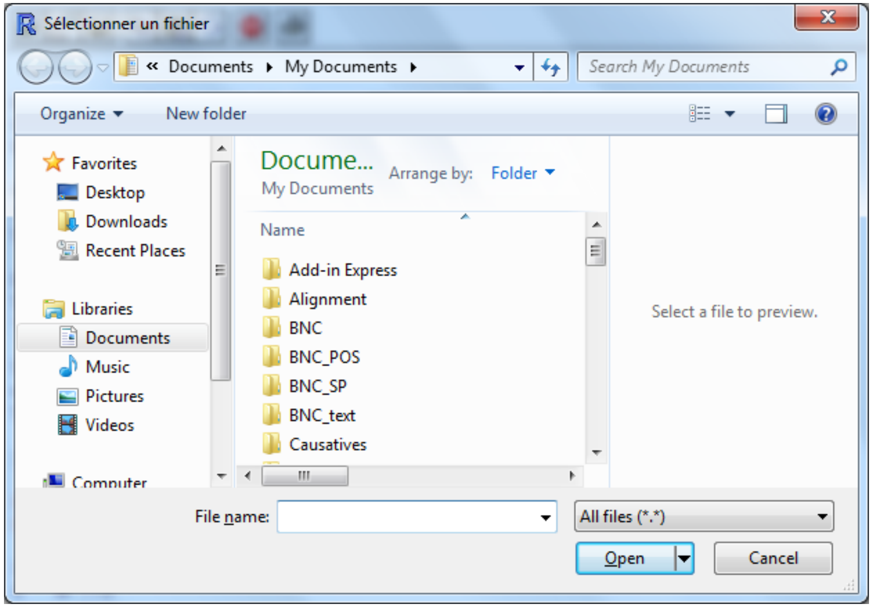
1. Create a similar table in Excel (or OpenOffice Calc). Don’t forget to create a header. In case of missing values, put NA. No empty cells!

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



Sélectionner un fichier

Documents ▸ My Documents ▸

Select a file to preview.

File name:

All files (*.*)

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

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R
5. First inspection of data
6. Work with ParTy

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

- Read the information about the built-in dataset sleep.
- Find how many and which subjects had actually a decrease in sleep.
- Make a subset of all subjects that belong to group 2.

English Lexicon Project data

```
> str(ELP)
'data.frame':     880 obs. of  5 variables:
 $ Word   : Factor w/ 880 levels
"abbreviation",..: 631 747 200 773 821 134 845 140
94 354 ...
 $ Length : int  7 10 10 8 6 5 5 8 8 6 ...
 $ SUBTLWF: num  0.96 4.24 0.04 1.49 1.06 3.33 0.1
0.06 0.43 5.41 ...
 $ POS    : Factor w/ 3 levels "JJ","NN","VB": 2 2
3 2 2 2 3 2 2 2 ...
 $ Mean_RT: num  791 693 960 771 882 ...
```

2 ways to tabulate a variable

```
> attach(ELP)
> summary(POS)
JJ  NN  VB
159 532 189
> table(POS)
POS
 JJ  NN  VB
159 532 189
```

Proportions and percentages

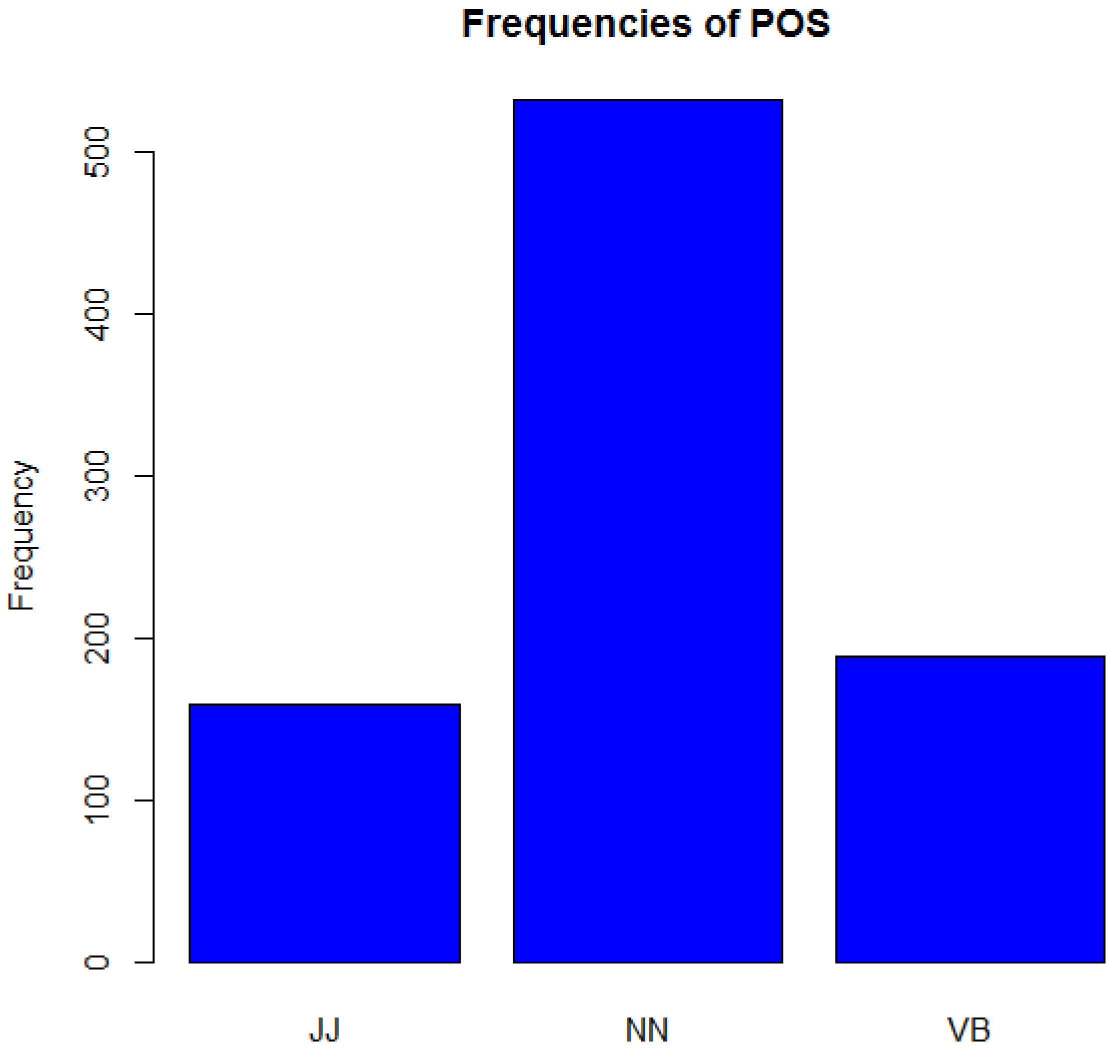
```
> prop.table(table(POS))
POS
       JJ        NN        VB
0.1806818 0.6045455 0.2147727
> prop.table(table(POS))*100
POS
      JJ       NN       VB
18.06818 60.45455 21.47727
```

Bar plot with counts

```
> barplot(table(POS))
```

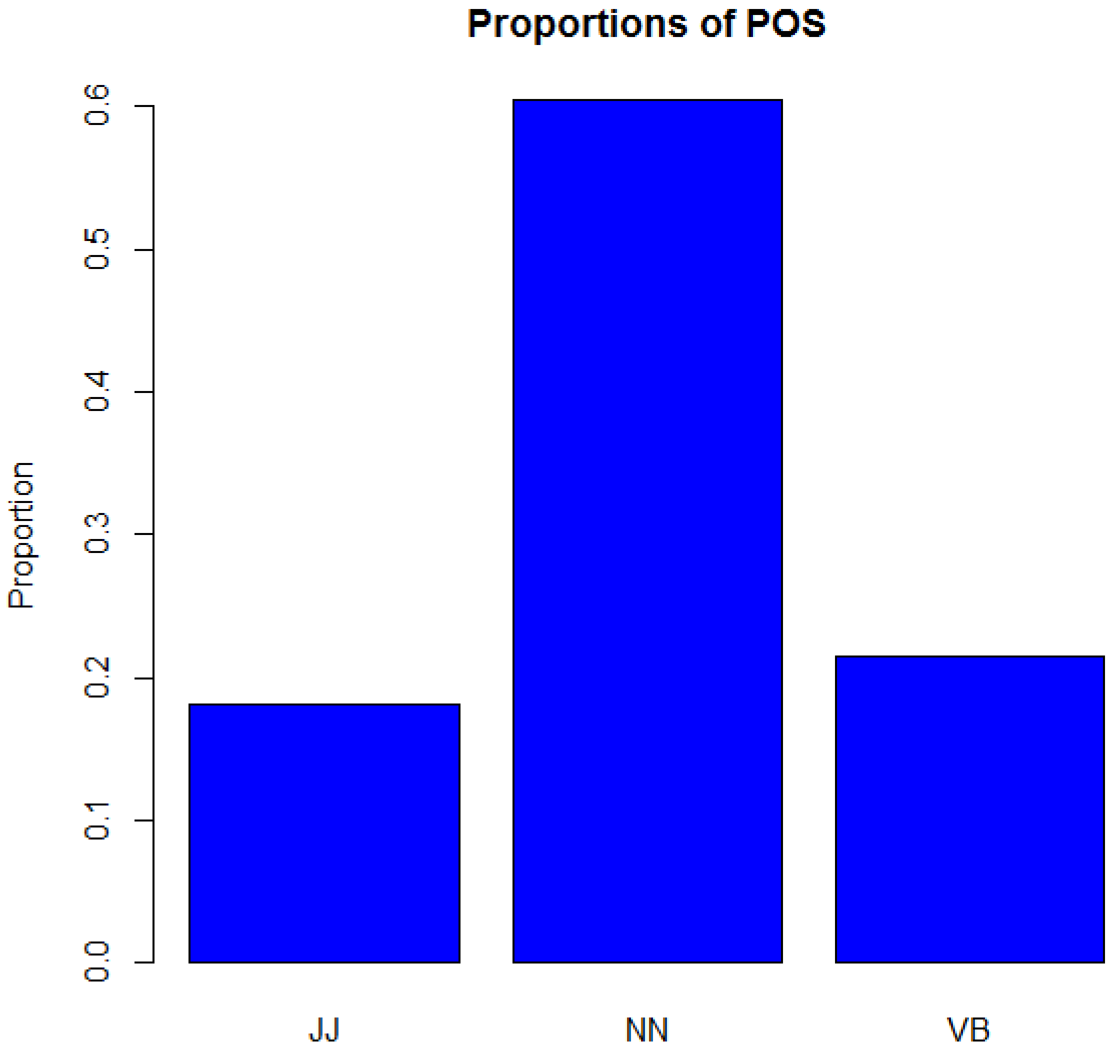
A fancier version:

```
> barplot(table(POS), col = "blue", main =
"Frequencies of POS", ylab = "Frequency")
```


Frequencies of POS
Frequency
0
100
200
300
400
500
JJ
NN
VB

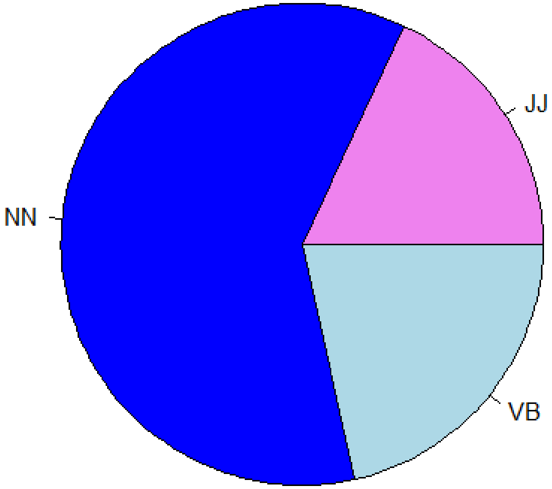
Bar plot with proportions

```
> barplot(prop.table(table(POS)), col = "blue",
main = "Proportions of POS", ylab = "Proportion")
```


Proportions of POS
Proportion
0.0
0.1
0.2
0.3
0.4
0.5
0.6
JJ
NN
VB

Pie chart

```
> pie(table(POS), col = c("violet", "blue",
"lightblue"))
```


JJ
NN
VB

Exercise: Your colours

- Create a factor with your and your fellow students' favourite colours.
- Compute the proportions of each of the colours. Which one is the most popular in the group?
- Create a pie chart with the colours corresponding to each colour category.

Nerds and geeks

```
> str(nerd)
'data.frame':     1316 obs. of  5 variables:
 $ Noun    : Factor w/ 2 levels "geek","nerd": 2 2
2 2 2 2 2 2 2 2 ...
 $ Num     : Factor w/ 2 levels "pl","sg": 1 1 1 1
1 1 1 1 1 1 ...
 $ Century : Factor w/ 2 levels "XX","XXI": 1 2 1
1 1 2 2 1 2 1 ...
 $ Register: Factor w/ 4 levels
"ACAD","MAG","NEWS",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Eval    : Factor w/ 3 levels
"Neg","Neutral",..: 2 2 2 2 2 2 2 2 2 2 ..
```

Noun by Century

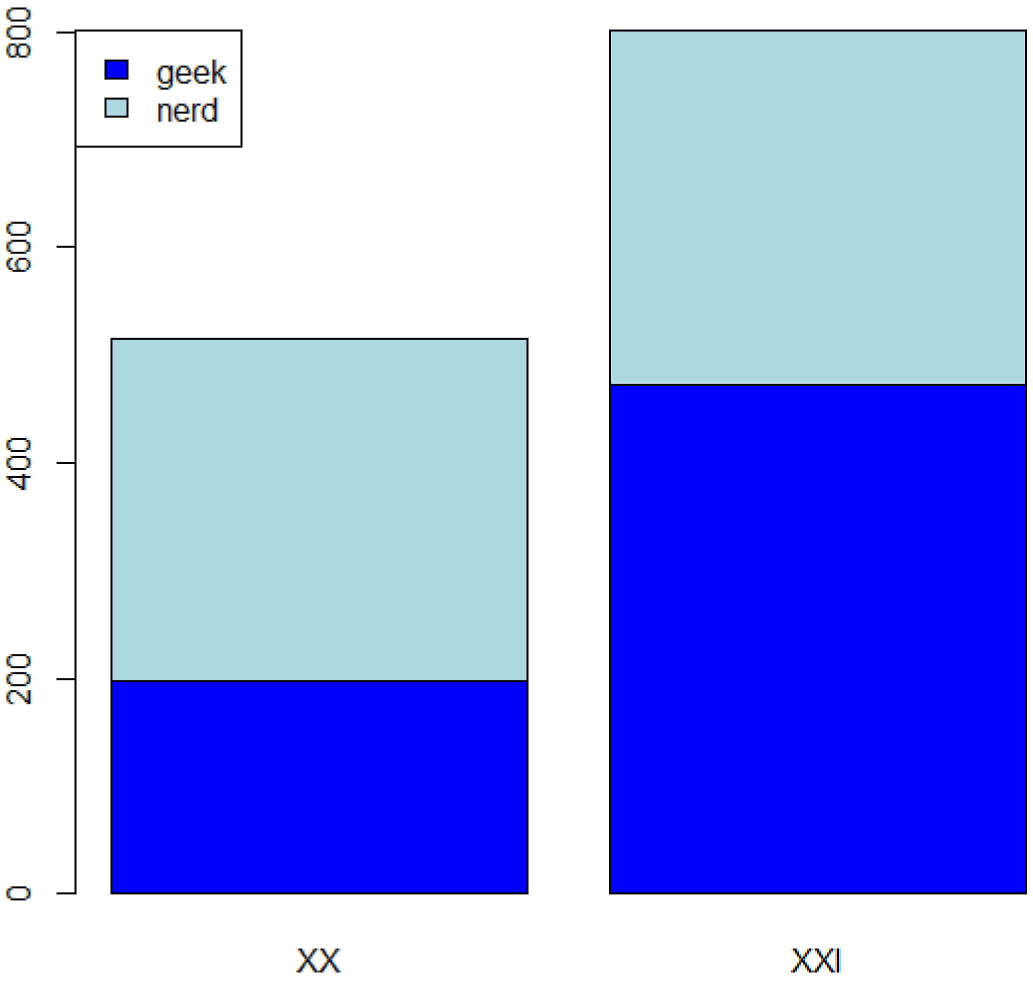
```
> attach(nerd)
> table(Noun, Century)
      Century
Noun    XX XXI
  geek 197 473
  nerd 318 328
```

Proportions for two-dimensional tables

```
> prop.table(table(Noun, Century)) #all cells sum up to 1
      Century
Noun          XX       XXI
  geek 0.1496960 0.3594225
  nerd 0.2416413 0.2492401
> prop.table(table(Noun, Century), 1) #rows sum up to 1
      Century
Noun          XX       XXI
  geek 0.2940299 0.7059701
  nerd 0.4922601 0.5077399
> prop.table(table(Noun, Century), 2) #columns sum up to 1
      Century
Noun          XX       XXI
  geek 0.3825243 0.5905119
  nerd 0.6174757 0.4094881
```

Bar plots with 2 variables

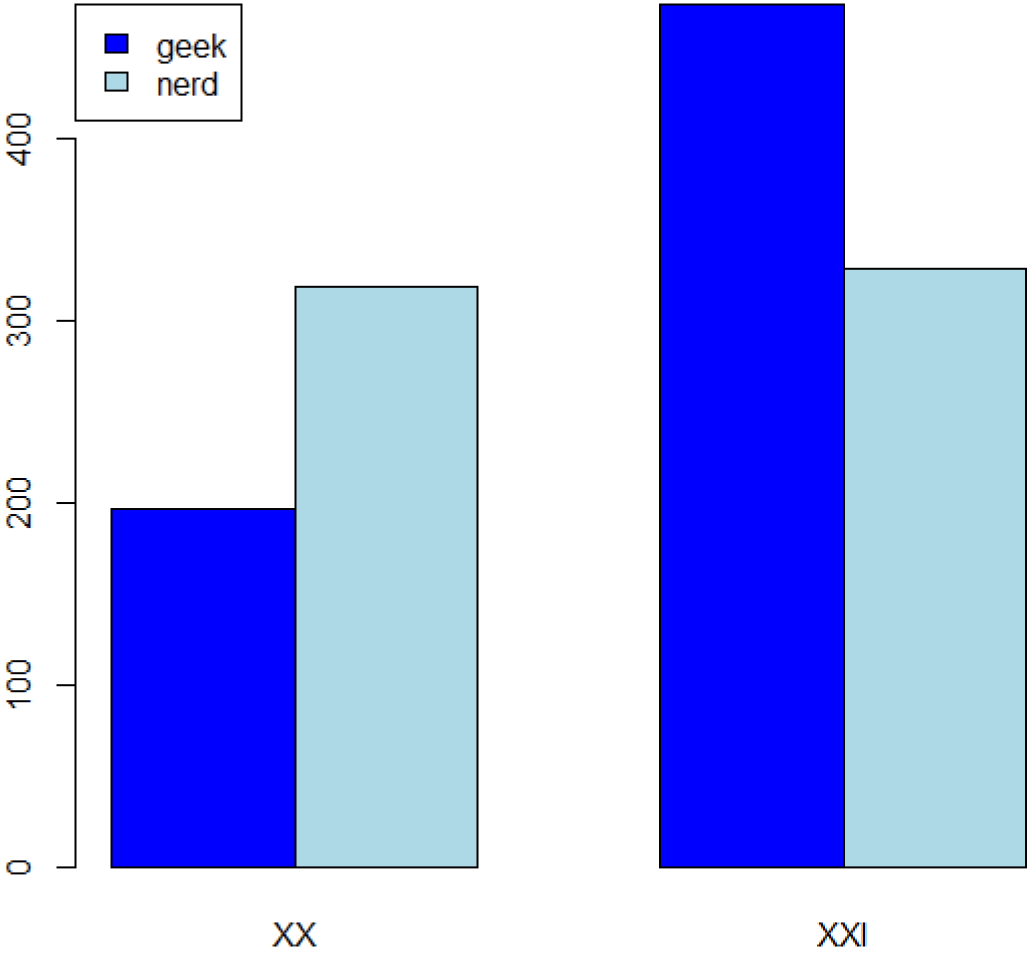
```
> barplot(table(Noun, Century), col = c("blue",
"lightblue"))
> legend("topleft", legend = c("geek", "nerd"),
fill = c("blue", "lightblue"))
```



0
200
400
600
800
geek
nerd
XX
XXI

Bar plots with unstacked bars

```
> barplot(table(Noun, Century), col = c("blue",
"lightblue"), beside = TRUE)
> legend("topleft", legend = c("geek", "nerd"),
fill = c("blue", "lightblue"))
```


geek
nerd
0
100
200
300
400
XX
XXI

Detach the data frames

```
> detach(ELP)
> detach(nerd)
```

Outline

1. Introduction to R
2. Basics of R syntax
3. Main objects in R
4. Creating and importing your data into R
5. Editing your data in R
6. Inspection of categorical data
7. Work with ParTy

Work with ParTy

1. Choose a linguistic construction in English (e.g. preposition in + NP, imperative).

2. Find the correspondences in different translations. Take at least two languages.

3. Create a dataset in Excel with the original sentences (rows) and translations in the target languages (columns).

4. Import the dataset into R as a data frame. Attach it.

5. Summarize the data in the translations as relative frequencies and percentages.

6. Visualize the data (bar plots, pie charts).

7. Save the plots.