Exercise & solution sheet: Day 1

Vangelis Theodorakis, Fatemeh Behjati, Julien Gagneur, Marcel Schulz

09 April, 2021

Contents

1 Vectors

First, create three named numeric vectors of size 10, 11 and 12 respectively in the following manner:

1) One vector with the "colon" approach: *from:to*
2) One vector with the `seq()` function: *seq(from, to)*
3) And one vector with the `seq()` function and the `by` argument: *seq(from, to, by)*

For easier naming you can use the vector `letters` or `LETTERS` which contain the latin alphabet in small and capital, respectively. In order to select specific letters just use e.g. `letters[1:4]` to get the first four letters. Check their types. What is the outcome? Where do you think the difference comes from?

```
# Answer :

# A. Create vectors
vector.1 <- 1:10
names(vector.1) <- letters[vector.1]

vector.2 <- seq(1, 11)
names(vector.2) <- letters[vector.2]

vector.3 <- seq(1, 12, by = 1)
names(vector.3) <- letters[vector.3]

typeof(vector.1)
## [1] "integer"
typeof(vector.2)
## [1] "integer"
typeof(vector.3)
## [1] "double"
```

2 Factors

1) Create a character vector consisting of three annotations *Mutant-1*, *Mutant-2*, *Control*.
2) Using this annotation vector, create a factor where each annotation is repeated 4 times in a sequential manner (*Mutant-1*, *Mutant-2*, *Control*, *Mutant-1*, *Mutant-2*, *Control*, ...). In addition, the levels are the sorted annotation values.
3) Print the results.

```
# Answer :
#1)
annotation <- c("Mutant-1", "Mutant-2", "Control")
#2)
test.factor <- factor(rep(annotation, 4), levels = sort(annotation))
#3)
print(test.factor)
##  [1] Mutant-1 Mutant-2 Control  Mutant-1 Mutant-2 Control  Mutant-1 Mutant-2
##  [9] Control  Mutant-1 Mutant-2 Control
## Levels: Control Mutant-1 Mutant-2
```

3 Data tables

The purpose of this exercise is to get familiarized with data.table and try out some of its useful features.

3.1 Basic operations

Please follow the steps listed below:

1) load the library called *dslabs*

2) Access the database called *brexit_polls*. You can take a look at the the *help* documentation of this database (*?brexit_polls*) to learn about its content.

For example:

column name	Description
pollster	Pollster conducting the poll.
poll_type	Online or telephone poll.
samplesize	Sample size of poll.
remain	Proportion voting Remain.
leave	Proportion voting Leave.

3) Inspect this data by checking properties such as the class type, the number of rows and columns, its column names, the unique values in the *poll_type* column.

4) Create a new variable called *brexit_DT* and assign the data.table converted version of *brexit_polls*.

```
# Answer :
library(data.table)
library(dslabs)

print("class of brexit_polls is")
## [1] "class of brexit_polls is"
class(brexit_polls)
## [1] "data.frame"

print("dim of brexit_polls is")
## [1] "dim of brexit_polls is"
dim(brexit_polls)
## [1] 127   9

print("column names of brexit_polls are")
## [1] "column names of brexit_polls are"
colnames(brexit_polls)
## [1] "startdate"  "enddate"    "pollster"   "poll_type"  "samplesize"
## [6] "remain"     "leave"      "undecided"  "spread"

print("a small subset of data looks like")
## [1] "a small subset of data looks like"
brexit_polls[1:3, 1:5]
```

```
##    startdate    enddate pollster poll_type samplesize
## 1 2016-06-23 2016-06-23   YouGov    Online       4772
## 2 2016-06-22 2016-06-22  Populus    Online       4700
## 3 2016-06-20 2016-06-22   YouGov    Online       3766

print("tissue types in data:")
## [1] "tissue types in data:"
unique(brexit_polls$poll_type)
## [1] Online    Telephone
## Levels: Online Telephone

brexit_DT <- as.data.table(brexit_polls)
```

3.2 More exciting operations

Continue from the previous part and perform the following actions:

5) From *brexit_DT* get the counts of Online and Telephone polls
6) What are the mean and median values of the *samplesize*
7) Add a new column *remain_polls* to *brexit_DT* that holds the multiplication of *samplesize* to *remain*
8) What is the range of values in this newly created column?
9) How do the mean values of *undecided* look like when grouped by *pollster*? How do they look like when grouped by *poll_type*? What is this mean value when *pollster* is *YouGov*?
10) Remove the column *remain_polls* created in step 7.

```
# Answer :
#5
brexit_DT[, .N, by= poll_type]
##    poll_type  N
## 1:    Online 85
## 2: Telephone 42

#6
brexit_DT[, .(mean_samplesize= mean(samplesize),
              median_samplesize= median(samplesize))]
##    mean_samplesize median_samplesize
## 1:        1694.457              1693

#7
brexit_DT[, remain_polls:= samplesize * remain]

#8
brexit_DT[, range(remain_polls)]
## [1]  268.38 2585.00

#9
```

```
brexit_DT[, mean(undecided), by= pollster]
##                               pollster         V1
##  1:                            YouGov 0.14153846
##  2:                           Populus 0.00000000
##  3:                        Ipsos MORI 0.06571429
##  4:                           Opinium 0.14555556
##  5:                            ComRes 0.09400000
##  6:                               TNS 0.22777778
##  7:                Survation/IG Group 0.11000000
##  8:                     ORB/Telegraph 0.02000000
##  9:                         Survation 0.17857143
## 10:                      BMG Research 0.15000000
## 11:                               ICM 0.13464286
## 12:                               ORB 0.02857143
## 13: Greenberg Quinlan Rosner Research 0.16000000
## 14:  Populus/Number Cruncher Politics 0.13750000
## 15:                  YouGov/The Times 0.19000000
## 16:                         Panelbase 0.12000000
brexit_DT[, mean(undecided), by= poll_type]
##    poll_type         V1
## 1:    Online 0.14141176
## 2: Telephone 0.09619048
# YouGov
brexit_DT[pollster == "YouGov", mean(undecided), by= poll_type]
##    poll_type     V1
## 1:    Online 0.1384
## 2: Telephone 0.2200

#10
brexit_DT[, remain_polls := NULL]
```