Lab 2: Intro to R

David Sichinava

October 11, 2019

Topics

- Read external files
- Simple tables
- Simple arithmetic operations

Instruction:

Follow the assignment step-by-step. Name your *.rmd* file in a following format: *your_surname_lab*2.*rmd*. For example,

```
sichinava_lab2.Rmd
```

Assignments:

Reading external data to R

First of all, create a working directory for your work in your computer. Add a subfolder *labs* and create a folder for the first lab: *lab_*2.

Open 'R-studio' and create new *R*-notebook. Assign a name and save it into *lab_*2 folder.

You can analyze variety of data formats in R including MS Excel, text files, Stata, SPSS, SAS, SQL tables, spatial data formats (šhp), javascript notation documents (.json) and so forth. Files can be stored into your computer or „somewhere" in the internet.

Usually, datasets are stored in tabular format, where each row represents a *case* and a column represents *variable*. Of course, there exist numerous other paradigms of organizing data (e.g. tidy data, document databases, relational tables), however we won't cover them throughout this course.

R's base functions easily read tab and comma separated data files. Usually, these data are stored in a text files having *.csv*, *.txt* or *dat* extensions. They look like as follows:

```
### tab separated files

1   6   a
2   7   b
3   8   c
4   9   d
5   10  e

### comma separated files

v1,v2,v3
1,2,3
4,5,6
7,8,9
a,b,c
```

Insert code chunk in your notebook. Be sure that your code is active. Write down a code between the accent marks which will navigate you to your working directory. From this link download *Galton.csv* file and save it to your working directory.

As I mentioned, reading text files to R is pretty straightforward. Please note that our file is *comma delimited*, that is, the columns in the file are separated with commas. You can easily see that if you open the file with notepad. For reading csv file, you should use *read.csv* function.

```
galton <- read.csv("Galton.csv")
```

Note that the first *galton* is a name of a new table which will store the information into R environment. The function is *read.csv*. Note that file names should be enclosed with quotation marks.

Sir Francis Galton, famous British statistician, geographer (and apparently, racist and eugenicist) was interested in studying how human physical characteristics are influenced by the treats of one's ancestor. He collected data for about two hundred families and studied, how childrens' height were affected by parents' physical characteristics.

Let's examine our data. Use function *names* to check what variables the dataset does contain. How many variables are there? Write down their names into your markdown document.

```
names(galton)
```

Simple descriptive statistics in R

Perfect. Now let's examine variable *height*. In order to refer objects in R, you should use dollar sign. The code below will show you the whole variable *height* into your console:

```
galton$height
```

Let's calculate mean, median and standard deviation of variable *height*.

```r
mean(galton$height)

median(galton$height)

sd(galton$height)
```

As I mentioned, Galton was linking parents' physical treats to those of children. Write a code which would calculate mean, median and standard deviation for variables $father$ and $mother$. Write a very simple description of the dataset.

Data transformation in R

You might notice that heights are in the dataset are given in inches. There are 2.54 centimeters in one inch. Let's create a new variable $height_cm$ which will store newly calculated values for kids' height.

Creating new variables in R are straightforward: you tell R that it should create a variable named $height_cm$ in table $galton$ which is $height$ times 2.54. To put it simple:

```r
galton$height_cm <- galton$height*2.54
```

Now summarize newly created variable and calculate its mean, median and standard deviation.

Create new variables for the centimeter equivalent for $father$ and $mother$ and calculate mean, median and standard deviation of these new variables.

Often we need to group certain values of a variable together, for example, to create a new variable which instead of actual ages will show age groups (e.g. whether the person is millennial, x-gener, baby-boomer, etc.). We will this on the following example. Let's create a new variable which would have four values: 1 would stand for children who are shorter than 150cm, 2 would correspond people whose height is between 150 and 169, 3 those who are 170-189 cm tall and 4 would stand for the tallest (190+) people. Remember we need to create a new variable $height_gr$ first.

```r
galton$height_gr <- galton$height_cm
```

Nice. Here we will tell R to replace $height_gr$ with 1 where $height_cm$ is less than 150. That is, when *filtering* the dataset we can refer to any variable existing in R environment. Let's see how this works:

```r
galton$height_gr[galton$height_cm < 150] <- 1
```

As you can see, square brackets indicate the code which filters the observations. You can have more complicated syntax as well:

```r
galton$height_gr[galton$height_cm>=150 & galton$height_cm<170 ] <- 2

galton$height_gr[galton$height_cm>=170 & galton$height_cm<190 ] <- 3

galton$height_gr[galton$height_cm>=190] <- 4
```

Easy, right? You can also attach labels, that is, textual meanings to the categories! In order to do that, you should turn the variable from *numeric* into *factor* type and then add labels.

```
galton$height_gr <- factor(galton$height_gr, labels=c("Short", "Medium", "Tall", "Very tall"))
```

Now let's do the same for mother's height. Please note that you should use mother's height in centimeters in order to calculate the new groups.

Frequency tables in R

Now let's see how many short, medium, tall and very tall individuals were in Galton's dataset. In order to do that, you should make a table of $height_gr$ variable. A table, simple frequency table.

```
table(galton$height_gr)
```

Similarly make table of newly created variable which shows categories for mothers' height.

Phew. Ready. Want to submit my assignment as soon as possible!

Zip the whole folder for the second lab. Name the file according to the following format: *surname_lab2.zip* like this:

```
shubladze_lab2.zip
```

Upload the file to this link before the start of our next meeting.
Good luck!