Big Data Analytics

Lab 1

1 Description

Today we are going to get familiar with the R language. If you have already installed the software, jump to Section 3. Otherwise use the links provided in Section 2 in order to install the required softwares. You can also learn basics of R in your browser, without installing anything, however its recommended that you setup the software on your computer.

2 Install R

We will need a working environment for R. First, we need to install R distribution and thereafter an IDE.

1. **R Language** - `https://cran.r-project.org`
2. **R Studio** - `https://www.rstudio.com/products/rstudio/download/`. R Studio makes it much easier to work with R.

3 The R Programming Language

The best way to learn a new language is by practice. DataCamp offers subscription-based courses on R, see this `https://www.datacamp.com/courses/free-introduction-to-r`. The introductory course is available for free, the only requirement is to create an account. Go through all 6 chapters of the course, which outlines: Intro to basics, Vectors, Matrices, Factors, Data frames, Lists.

R (Wikipedia) is a programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

R is an interpreted language; users typically access it through a command-line interpreter. If a user types 2+2 at the R command prompt and presses enter, the computer replies with 4.

4 Basic R

It is a short version of the on-line available materials. Read all the instructions and run the code provided.

1. You can use R as a normal calculator. Just type some expression, like `5 + 6`, and it will evaluate that to you and return the result. Result is printed and lost by default. To save it for further reuse you have to assign it to a name, like here:

   ```
   x <- 5
   ```

2. If you want to evaluate a variable just use it:

   ```
   x
   ```

3. Names in R can store more than one value - vectors. The easiest way to create a vector of values is to use the `c()` function. Check the result of following:

   ```
   y <- c(1, 2, 3, 4, 5)
   ```

   We have created a vector of 5 numbers. As these are following number, the same can be achieved using:

   ```
   x <- seq(1, 5, 1)
   ```

   that goes from 1 to 5 by 1.

4. Vector can be used as any other value - you can perform arithmetic operations on them:

   ```
   y <- 2 * x + 1
   ```

   will create a vector with the same dimensions, with values matching the expression.

5. Vectors can hold any type of data. Other, very useful, is boolean data that is usually returned by evaluating conditions. You can create conditions by checking values with values, values with variables, etc.:

   ```
   temp <- x > 13
   ```

   it returns a logical vector where each field corresponds to a result of this condition applied to the field in the original vector.

6. Other types of vectors:

```
> a <- c(0.5, 0.6) # numeric
> a <- c(TRUE, FALSE) # logical
> a <- c(T, F) # logical
> a <- c("a", "b", "c") # character
> a <- 9:29 # integer
> a <- c(1+0i, 2+4i) # complex
```

7. Sometimes, usually when you are working with data collected somewhere, you might have missing values or values that are not available. In R they are represented by `NA` and you can check if something is missing by using the `is.na(...)` function that return `TRUE` if passed variable is not available or missing, and `FALSE` otherwise. Consider example:

   ```
   z <- c(1, 3, NA)
   is.na(z)
   ```

8. There are expressions that can not be evaluated, such as `0 / 0` or `Inf - Inf`, they as so-called Not a Number, NaN values. `is.na(x)` will return true for them as well, if you would like to check if something is NaN only, there another function - `is.nan(...)`. Check how it works.

9. R contains loads of predefined functions by default, it it impossible to remember how they work and what parameters they take. If you would like to check how something works or any other information about the function, you can just retrieve the manual using `help()` function:

   ```
   help(mean)
   ```

   Above will return the manual for the `mean` function.

10. Vectors are not the only types of objects that we can work on. Other are:

    - vector - list of objects of the same type,
    - matrix - 2D object that stores objects of the same type,
    - array - nD object that stores the same type of data,
    - function - code that takes an input and outputs other R objects,
    - list - object that stores objects of different types,
    - data.frame - matrix-like object where columns might have different types.

11. Check how the `rep` function works and use it to get `x` variable 5 times end-to-end.

12. Do the same to repeat each element 3 times.

13. You can read data from user using `scan()` function. Check the help and use this function to load `100, 20, 30, 50, 60` into `w` variable.

14. Create an array:

    ```
    A <- array(1:25, dim=c(5,5))
    A # it prints out the array
    ```

    `1:25` is another way of creating a sequence of numbers, `dim=c(5,5)` argument defines the dimensions of the array and it shapes the data to fill the dimensions.

15. You can slice the array in the following way:

    ```
    A[1, 4] # first row, forth column
    ```

    That returns the element from the first row and forth column. Indices in R start from 1. Of course it is possible to retrieve parts of the arrays that are bigger than a single cell:

    ```
    A[1, 1:3] # first row, columns 1 to 3
    A[1:3, 1:3] # first 3 rows and first 3 columns
    A[c(1,3, 5), 2] # rows 1, 3 and 5 from column 2
    ```