Introduction to Data Preparation and Description with R

Before you start

Comprehensive documentation on R is available from The Comprehensive R Archive Network (CRAN) website: http://ftp.heanet.ie/mirrors/cran.r-project.org/. For the learning process to be successful you, the student, are expected to explore the language over and beyond the topics discussed in class and this website provides a starting point, but there are many other excellent resources on the internet. One of those is datacamp.com, where you may like to complete the free tutorials available for R.

NOTE: In specifying commands, we use descriptive text in angular brackets as placeholders for actual strings. For example, if in a line to type at the command prompt you see the string `<name of function>`, then the string in angular brackets, including the brackets, should be replaced with a literal string, e.g. `mean`.

Exercise 1: Running R and Basic Functions

In this exercise you will familiarise yourself with the R tools and perform some basic tasks with the R language.

1. **Running R:** Run RStudio from the Start Menu in Windows (if you want to run it on Linux or Mac, it is available for those operating systems as well). The window that opens will contain an R console sub-window. Commands are typed into this console and can be saved at the end of the session so that the interaction history and any loaded data remain available for future sessions.

2. **Inserting comments:** In the command line (the '>' character is the prompt shown by R) type in the following:

   ```
   > # this is a comment
   ```

   The '#' at a beginning of a line in R indicates that the line is a comment and can be ignored by the interpreter. The comments can be used in R scripts (more about them later) like in any other programming language, but can also be useful to remind yourself of what you were doing in a session, when it is reloaded upon re-starting R.

3. **Saving the session:** The default filename for the saved R image is .RData. To save the image during the session type:

   ```
   > save.image()
   ```

   This will save the image to the default file. If you want to give the image file a different name, pass it in to the function:

   ```
   > save.image(" my_image_file")
   ```

   Finally, to leave the session you should type:

   ```
   > q()
   ```

   This command will ask whether you want to save the session and if you confirm the image

will be saved to the default file.

Try out all the described commands.

4. **Getting help.** If you know the name of a function, you can get more information about it by typing:

   ```
   > ?<name of function>
   ```

   For example:

   ```
   > ?mean
   ```

   If you don't exactly know what you are looking for, i.e. are not sure about the spelling of a function, you can type:

   ```
   > help.search(<approximate name>)
   ```

   or use the shortcut:

   ```
   > ??<approximate name>
   ```

   For example:

   ```
   > ??fozzy
   ```

   Try out all the described commands.

5. **Listing session data.** To list all the data loaded in your session, type:

   ```
   > ls()
   ```

   If you call the list (ls) function before creating or importing any data, it will not list anything, but let's create a vector of data:

   ```
   > xdata <- c(1:6)
   ```

   And call the list function again:

   ```
   > ls()
   ```

   This time vector's name should be displayed (it is the only item on the list). To see the contents of `xdata`, type:

   ```
   > xdata
   ```

   Try out all the described commands.

6. **Working directory.** To find out what the working directory is, use this function:

   ```
   > getwd()
   ```

   Try out the described command.

7. **Working with libraries.** One of the things that make R attractive for researchers is the wealth of libraries that are available for it. Some of the libraries are provided with the installation, and to be used just have to be declared, as in:

   ```
   > library("MASS")
   ```

   Other libraries need to be installed before they can be declared and used. For example:

   ```
   > install.packages("ks")
   ```

   Try out all the described commands.

Exercise 2: Basic Statistics

In this exercise you will perform some simple statistical operations and visualisations.

NOTE: In specifying commands, we use descriptive text in angular brackets as placeholders for actual strings. For example, if in a line to type at the command prompt you see the string `<name of function>`, then the string in angular brackets, including the brackets, should be replaced with a literal string, e.g. `mean`.

1. **Median, mean and mode.** In the R console, type in the following to define a vector of values:

   ```
   > xdata <- c(1, 2, 3, 4, 5, 5, 6, 6, 6, 7, 8, 8)
   ```

   R has functions to calculate the median and the mean:

   ```
   > median(xdata)
   ```

   ```
   > mean(xdata)
   ```

   There is no inbuilt function for mode, but you can install the following package with a function that returns the mode: `> install.packages("modeest")`

   After calling the install function, you must interact with the GUI to find a mirror download site. `> library("modeest")`

   Then call the function: `> mlv(xdata, method="mfv")`

   The function determines the most likely value by the method of most frequent value, which is the mode.

2. **Creating and manipulating a dataset.** For this part of the exercise, you must download the script file `cardata.R`. From the **File** menu in R, click **Open script**, then navigate to the downloaded file and open it. Having the script file open allows you to execute lines from it in the R Console. Place the cursor on the first line of the opened script, then run it by clicking **Code**, then **Run Selected Line(s)** (there is also a keyboard shortcut for this, **Ctrl**+**Enter**). Once the line has been executed examine the console and make sure that the vector of IDs has been created by typing:

   ```
   > i
   ```

   Repeat the process for all the lines in the file (the cursor will move automatically through the lines in the file as they are executed).

   Now we can create a dataset from the vectors:

   ```
   > cardata <- data.frame(i, k, b, p)
   ```

   Have a look at your dataset in a table by typing:

   ```
   > cardata
   ```

   Give the variables more understandable names:

   ```
   > names(cardata) <- c("ID","Colour","Make", "NCTPass")
   ```

   Display the table again to check that the attribute names have been assigned: `> cardata`

   Extract a subset of the table by using each of the three methods:

   ```
   > cardata[c("ID","NCTPass")]
   ```

   ```
   > cardata[c(3:4)]
   ```

   ```
   > cardata$Make
   ```

We can order the values of the variable `Make` (albeit a bit arbitrarily) by using the function `ordered`:

```
> cardata$Make <- ordered(cardata$Make, levels=c("dacia", "mazda",
"toyota", "VW", "mercedes", "BMW"))
```

Now, to see the result of this ordering, try:

```
> cardata$Make
```

3. **Distribution: bar chart.** Now that we have our data in place, we are ready to create a bar-chart. We are going to create a bar chart showing the counts for the different values of the Make variable. The first step is to extract information about the counts and that is done by the simple but important function `table`:

   ```
   > counts <- table(cardata$Make)
   ```

   To check the result of the call, you can display the `counts` vector:

   ```
   > counts
   ```

   Now we can get R to create our bar-chart:

   ```
   > barplot(counts, main="Make distribution", xlab="Make")
   ```

   The picture will be displayed in a new *device* window.

4. **Distribution: frequency histogram.** To learn about creating a histogram and the various R parameters that can be used with that type of chart, follow the tutorial here.

5. **Inter-quartile range and boxplot.** Use the dataset imported in the datacamp tutorial to find the interquartile range for some variables, e.g. WEIGHT and HEIGHT:

   ```
   > IQR(chol$WEIGHT)
   ```

   etc.

   Can you conclude anything from the values that are being returned?

   We can use the same dataset to plot boxplots. We will plot boxplots for non-smokers, pipe-smokers and cigarette smokers. The smoking status, being a nominal variable, cannot be box-plotted and we use the variable to divide the data into three groups, plotting a boxplot for each of those.

   ```
   > boxplot(chol$WEIGHT chol$SMOKE, data=chol, main="weight",
   xlab="smokes", ylab="weight", ylim=c(0,120))
   ```

   Notice the parameters that are being passed into the boxplot function: the x and the y axis labels (`xlab` and `ylab`) and the plot title (`main`). The data is the dataset we downloaded, `chol`. As the number of possible configuration parameters is very large and applicable in many contexts, rather than employing positional identification for them, the parameter passing syntax is of the form `<param name>=<param value>`. The first, data related parameter is positional and in the case of the boxplot specifies the variable to be plotted, against the variable that defines the categories for which the plot is to be made. In our case the variable to be plotted is chol$WEIGHT and we use the smoking status for categorisation. Try the plot for the other numerical variables (HEIGHT, AGE) instead of WEIGHT.

6. **Variance and standard deviation.** The functions in R for variance and standard deviation, respectively, are `var(<variable name>)` and `sd(<variable name>)`. Try them out with the downloaded data.

7. **Scatterplots.** Look up the documentation for function `plot` and use the downloaded data

to produce 2 different plots.