Exercises

Exercise 1 : Getting to know R and RStudio

Read Chapter 1 to help you complete the questions in this exercise. We'll also bounce occasionally to Chapter 2 for a few questions (links will be provided).

1. Start RStudio on your computer. Create a new RStudio Project (select File –> New Project on the main menu). Create the Project in a new directory by selecting 'New Directory' and then select 'New Project'. Give the Project a suitable name in the 'Directory name:' box (maybe 'intro_2_r'? - see Section 1.9 of the Introduction to R book for a discussion on naming files and directories). Choose where you would like to create this Project directory by clicking on the 'Browse' button. Finally create the project by clicking on the 'Create Project' button. This will be your main RStudio Project file and directory which you will use throughout this course. If you're confused see Section 1.6 of the introduction to R Book which covers RStudio Projects or watch the 'RStudio Projects' video here.

2. Now create a new R script inside this Project by selecting File –> New File –> R Script from the main menu (or use the shortcut button). Before you start writing any code save this script by selecting File –> Save from the main menu. Call this script 'exercise_1' or something similar (remember, files names should not contain spaces!). Click on the 'Files' tab in the bottom right RStudio pane to see whether your file has been saved in the correct location. Ok, at the top of almost every R script (there are very few exceptions to this!) you should include some metadata to help your collaborators (and the future you) know who wrote the script, when it was written and what the script does (amongst other things). Include this information at the top of your R script making sure that you place a `#` at the beginning of every line to let R know this is a comment. See Section 1.10 for a little more detail.

3. Explore RStudio making sure you understand the functionality of each of the four windows. Take a look at Section 1.3 of the Introduction to R book for more details or watch this video. Take your time and check out each of the tabs in the windows. The function of some of these tabs will be obvious whereas others won't be useful right now. In general, you will write your R code in the script editor window (usually top left window) and then source your code into the R console (usually bottom left) by clicking anywhere in the relevant line of code with your mouse and then clicking on the 'Run' button at the top of the script editor window. If you don't like clicking buttons (I don't!) then you can use the keyboard shortcut 'ctrl + enter' (on Windows) or 'command + enter' (on Mac OSX).

4. Now to practice writing code in the script editor and sourcing this code into the R console. Let's display the help file for the function `mean`. In your script type `help('mean')` and source this code into the console. Notice that the help file is displayed in the bottom right window (if not then click on the 'Help' tab). Examine the different components of the help file (especially the examples section at the end of the help file). See Section 2.5 of the Introduction to R book for more details on using the help functions.

5. The content displayed in the bottom right window is context dependent. For example if we write the code `plot(1:10)` in our script and then source it into the R console the bottom right window will display this plot (don't worry about understanding the R code right now, hopefully this will become clear later on in the course!).

6. Next, let's practice creating a variable and assigning a value to this variable. Take a look at Section 2.2 of the Introduction to R book for further information on how to do this or if you prefer watch the Objects in R video. Create a variable called `first_num` and assign it the value `42`. Click on the 'Environment' tab in the top right window to display the variable and value. Now create another variable called `first_char` and assign it a value `"my first character"`. Notice this variable is now also displayed in the 'Environment' along with it's value and class (`chr` - short for character class).

7. Remove the variable `first_num` from your environment using the `rm()` function. Use the code `rm(first_num)` to do this. Check out the 'Environment' tab to ensure the variable has been removed. Alternatively, use the `ls()` function to list all objects in your environment.

8. Let's see what happens if we assign another value to an existing variable. Assign the value `"my second character"` to the variable `first_char` you created in Q6. Notice the value has changed in the 'Environment'. To display the value of `first.char` enter the name of the variable in the console. Don't to forget to save your R script periodically!

9. OK, let's leave RStudio for a minute. Using your favourite web browser, navigate to the R-project website and explore links that catch your eye. Make sure you find the R manuals page and the user contributed documents section. Download any manuals that you think you might find useful (some are listed in the course manual) and save them on your computer (or USB drive).

10. Click on the 'Getting Help' link on the R-Project website. Scroll down and use 'Rseek' (under 'R Help on the Internet') to search for the term 'mixed model p values' (this is a controversial subject!) and explore anything that looks interesting. Also experiment with the 'R site search' and 'Nabble R Forum' links. Learning how to search for help when you run into a problem when using R is an acquired skill and something you get better at over time. One note of caution, often you will find many different solutions to solving a problem in R, some written by experienced R users and others by people with less experience. Whichever solution you choose make sure you understand what the code is doing and thoroughly test it to make sure it's doing what you want.

11. OK, back to RStudio. Sometimes you may forget the exact name of a function you want to use so it would be useful to be able to search through all the function names. For example, you want to create a design plot but can only remember that the name of the function has the word 'plot' in it. Use the `apropos()` function to list all the functions with the word plot in their name (see Section 2.5.1 of the Introduction to R book). Look through the list and once you have figured what the correct function is then bring up the help file for this function (Hint: the function name probably has the words 'plot' and 'design' in it!).

12. Another strategy would be to use the `help.search()` function to search through R's help files. Search the R help system for instances of the character string 'plot'. Take a look at Section 2.5.1 for more information. Also, see if you can figure out how to narrow your search by only searching for 'plot' in the `nlme` package (hint: see the help page for `help.search()`).

13. R's working directory is the default location of any files you read into R, or export from R. Although you won't be importing or exporting files just yet (that's tomorrows job) it's useful to be able to determine what your current working directory is. So, read Section 1.7 of the Introduction to R book to introduce yourself to working directories and figure out how to display your current working directory.

14. Let's finish up by creating some additional useful directories in your Project directory. If you've followed the **Data** instructions when downloading datasets for this course you will already have a directory called `data` in your Project (if you didn't then take a look at the instructions under **Data** to create this directory). Now let's create another directory called `output` where you'll save data files and plots you generate later on during this course. This time, instead of clicking on the 'New Folder' icon in RStudio we'll create a new directory using R code directly in the R console (you can also interact with your computer's operating system in all sorts of useful ways). To do this use the `dir.create()` function to create a directory called `output` (See Section 1.8 of the Introduction to R book for more details). If you fancy it, you can also create a subdirectory in your `output` directory called `figures` to store all your fancy R plots for your thesis. You can list all the files in your directories using the `list.files()` function. Can you figure out how to list the directories as well? (hint: see `?listfiles` or Section 1.8 of the course book).

15. Don't to forget to save your R script. Close your Project by selecting File –> Close Project on the main menu.

End of Exercise 1