BIOL 4001 - Biostatistics

R Tutorial 1: Introduction to R and R Studio

Acknowledgment

Thank you to Michael C. Whitlock and Dolph Schluter. This tutorial is based on their Labs using R, which accompanies their book, *The Analysis of Biological Data* (3rd edition).

R Software

Implementing the statistical procedures in this course requires the use of computer software and there are many different statistical software packages available. This course uses R, which is a free software environment for statistical computing and graphics that is widely used by biologists and is available for Windows, Mac, and Linux operating systems.

R has a command line interface in which you submit lines of computer code that the software then implements. While this can be a little daunting when first learning to use R, this interface has a big advantage over menu-based software like Excel or Minitab because you can use scripts (saved text files of code) to recreate and adapt analyses. Also, as a complete computing language, it is highly customizable, and researchers have written thousands of accessible routines for implementing just about any statistical procedure you can think of.

R Studio

While it is possible to use R as a standalone package, it is easier to use front-end software that creates a more user-friendly interface. For this purpose, this course uses R studio, which is an integrated development environment (IDE) for R.

Installing R and R Studio

Go to Comprehensive R Archive Network Mirrors and choose the link for a location close to you, e.g., Simon Fraser University, Burnaby. Then, at the top of the screen under "Download and Install R," choose the download link for your computer operating system.

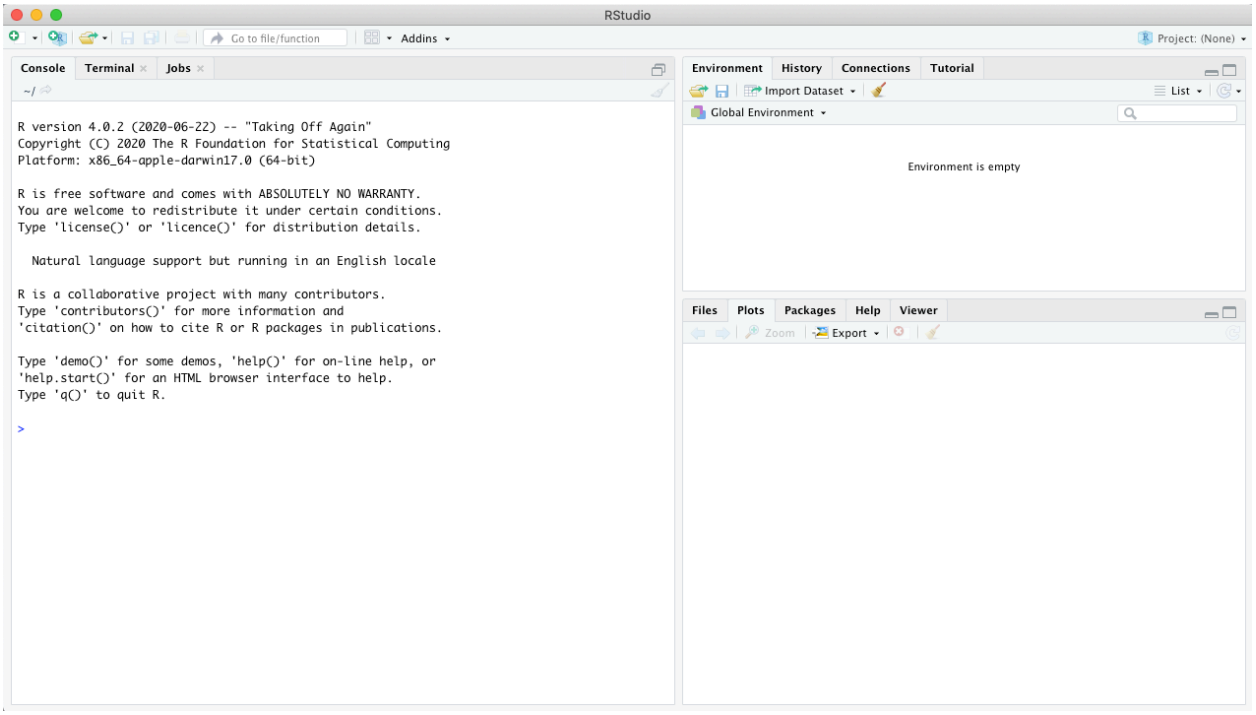
- Windows: At the top of the screen under "R for Windows," click on the link labeled "base," then click on the link "Download R X.X.X for Windows." Run the .exe file that downloads.
- Macs: Part way down the screen titled "R for Mac OS X," click on the link labeled "R-X.X.X.pkg" under "Latest release." (You may need an earlier release if you have an old Mac OS version.) Before opening the .pkg file that downloads, make sure you've first installed XQuartz.

Go to Download the RStudio IDE and click the "DOWNLOAD" button for "RStudio Desktop Free" on the left. Then, choose the download link for your computer operating system.

- Windows: Run the .exe file that downloads.
- Macs: Open the .dmg file that downloads and in the resulting window move the RStudio icon into the Applications folder.

Running R inside R Studio

Once you've installed R and R Studio, start R Studio. R Studio will automatically start R, which runs inside R Studio. (In other words, there is no need to start R manually outside R Studio.) You should see RStudio divided into three windows; this is what it looks like in Mac OS:



The "Console" window on the left is where you submit lines of computer code that R then implements. Results appear in the Console as text or in the "Plot" tab in the lower right window if the result is a plot.

Set Working Directory

You should also see a menu bar at the top of the screen with File, Edit, Code, View, …. In the menu bar, select Session > Set Working Directory > Choose Directory. Select a folder on your computer in which you will save your work for these tutorials. In the Console you should see "setwd" together with the path to your selected folder, e.g.:

```
setwd("~/Documents/BIOL 4001/RTutorials")
```

The word “setwd” is an R function and the round brackets are used to provide arguments to the function.

You can also type directly into the Console after the “>” prompt to enter commands, e.g.:

```
help(setwd)
```

This will activate the “Help” tab in the lower right window and displays the help information for the “getwd” and “setwd” functions. You should get in the habit of using “Help” any time you are stuck in R.

Use R as a Calculator

Some functions don’t take any arguments, e.g.:

```
1+1
```

This returns the answer labeled with “[1]” to denote that it is the first element in the answer (which is redundant in this case, but will be useful later):

```
[1] 2
```

Try the following basic arithmetic examples for further practice:

```
2*3
3^2
sqrt(4)
```

R Scripts

To save time typing commands into the Console manually, it is much easier to work with scripts, which are saved text files of code. In the menu bar at the top of the screen, select File > New File > R Script. The window on the left should split into two with the upper part labeled “Untitled1.” Copy/paste the basic arithmetic examples above into this window and select File > Save to save the script file as “RTutorial1.R” in the “RTutorials” folder you created earlier. You can then open this script file in the future to recreate or adapt your previous analyses.

Another advantage of script files in RStudio is that you can run lines of code directly from the script file. For example, click to place the cursor to the left of “2*3” in your script file. Then click the “Run” button 

at the top of the Console to implement this line of code. Click the “Run” icon again to implement the next line of code, and so on. Click/drag to highlight a series of lines of code to run all the selected code at once.

Comments

When working with R scripts, it is good practice to annotate the script with comments to remind yourself or other users what the code does. Denote comments using a hash symbol (#), which tells R to ignore anything that appears on the line after the #, e.g.:

```
# This is a comment
log(100) # default log function calculates natural log
log(100, base=10) # specify other logs using base argument
```

Copy/paste these lines of code into your RTutorial1.R script file and run them to see what happens.

Variables

R is much more than a simple calculator. Its real power comes into play when we use variables. Suppose we have a variable, x, that has the value 5. This is accomplished in R using the assignment operator, which consists of a "less than symbol" (<) and a "hyphen" (-) typed together to form a left-hand arrow (<-), e.g.:

```
x <- 5
x
```

which results in

```
[1] 5
```

Try the following examples for further practice:

```
x + 2
x <- 6
x
```

When naming variables, note that R has specific syntax requirements:

- A name has to start with a letter
- Subsequent characters can be letters or numbers or underscores, "_"
- Do not include spaces or symbols that have other meanings (e.g., "-" means "minus")
- R is case sensitive, so "x" is different than "X"

Vectors

We typically work with samples in statistics, for example, a sample of six temperature measurements in degrees Celsius: 21, 29, 20, 10, 23, 17. We can create this as a vector in R, e.g.:

```
tempC <- c(21, 29, 20, 10, 23, 17)
```

The “c” function combines values into a vector. Vectors are useful because we can do calculations with the entire vector at once, for example, calculating the corresponding temperatures in degrees Fahrenheit:

```
tempF <- tempC * 9/5 + 32
tempF
```

which results in

```
[1] 69.8 84.2 68.0 50.0 73.4 62.6
```

If we only want to work with a subset of the elements in a vector, we can use square brackets, e.g.:

```
tempC[2]   # 2nd element
tempC[3:5] # the 3rd, 4th, and 5th elements
tempC[-6]  # all but the 6th element
```

Try the following self-explanatory examples for further practice working with vectors:

```
mean(tempC)
sum(tempC)
length(tempC)
```