Getting Started with R and RStudio

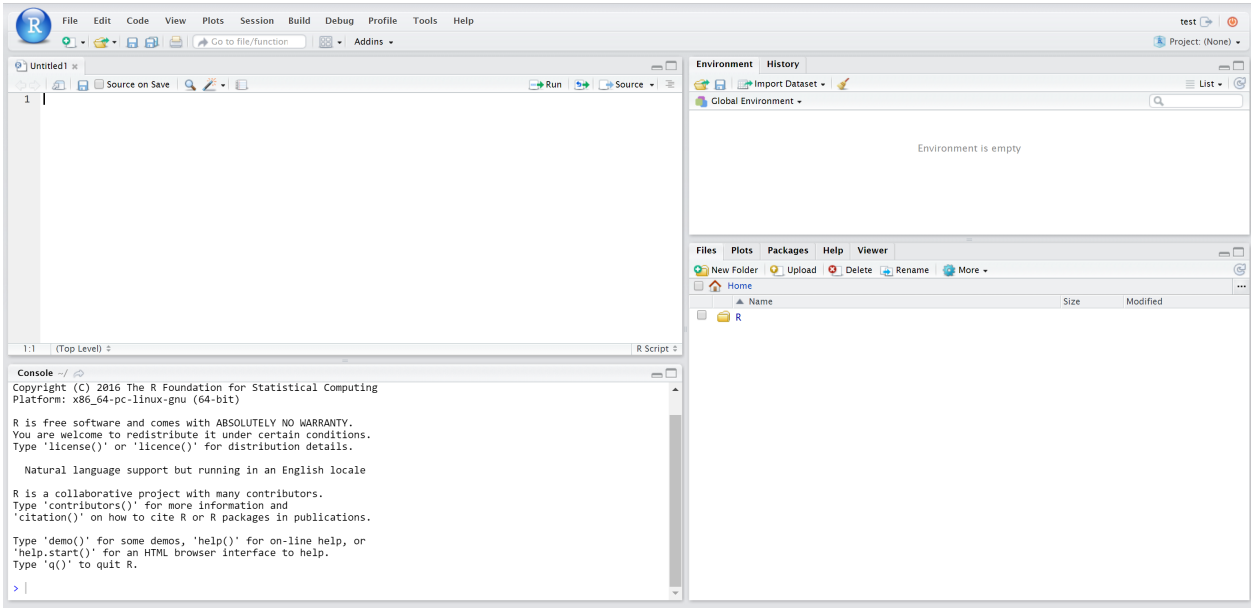
R is an open-source statistical language widely used in biostatistics and computational biology, among other fields of research. *RStudio* is a "front end" to R that simplifies important aspects of using R and makes it much easier to work within the R environment. Both R and *RStudio* run identically under the Mac OSX, Microsoft Windows, and Linux.

Installing R and *RStudio*

First, download R from http://cran.us.r-project.org/. Versions are available for Windows, Mac OS X, and Linux. Follow the instructions when running the installation program, selecting the default options when prompted.

RStudio can be downloaded from https://www.rstudio.com/products/rstudio/download/. Scroll down to "Installers for Supported Platforms" and select the appropriate version for your system. Leave all default settings in the installation options.

The RStudio Console



The *RStudio* environment is organized by panes, with the default layout shown above; the script editor and console are on the top and bottom left, and there are additional panes on the right. The script editor is used to create and edit files, such as R script files. Multiple files can be open at once, and will appear as separate tabs. If the script editor is not visible, open a new file via *File > New File > R Script* to make the script editor reappear. When commands are run in the script editor, the commands and the corresponding output appear in the console.

RStudio can be used to create several types of files; the major two types of files are R Script files (`.R`) and R Markdown files (`.Rmd`). An R Script file contains code that can be executed in the script editor. Saving code in a script file makes it easy to reuse or modify code at a later time. An R

Markdown file contains both code and plain text, and can be used to generate a PDF, HTML, or Word document that contains output (including figures or calculations) along with the text.

The following section introduces the rest of the *RStudio* layout and describes the use of the script editor to enter commands. R Markdown will be introduced in the first lab exercise.

R and *RStudio* Tutorial

1. Open a new R Script file via *File* > *New File* > *R Script*. While commands can be entered in either the script editor or the console, it is recommended to always use the script editor; code entered in the editor can be saved, as well as easily edited and re-run.

2. At its most basic, R can be used as a calculator. Enter `8 + 3` in the script editor and click the *Run* button at the top right of the script editor to send the command to the console, where the output will appear. Alternatively, with the cursor on the line of code, use the keyboard shortcut Ctrl/Cmd + Enter. Try entering several lines of arithmetic expressions and running them at once. The "#" symbol marks off text as 'comments', which are not run as code; this is a useful tool for making notes within the code.

```
#some arithmetic expressions
8 + 3
log(2)
((121/3) * (6^3))/(pi)
```

3. The previous calculations only produce output in the console. To save a value, assign it a name by using "=". Entering the name of a value will return the value. For example, run:

```
#create x and y
x = 8 + 3
y = log(2)

#calculation
x + y

#define and return z
z = x * y
z
```

4. Take a look at the Environment tab in the top right pane—the values of `x`, `y`, and `z` are displayed. Any created data structures or loaded datasets will appear in this tab. All objects can be cleared by selecting the broom icon.

5. Note that R is case-sensitive; `x` and `X` are not the same. If you try to run `X`, an error will be returned since no value has been named `X`. If the same name is used again to define a new value, R will overwrite the previous information. For example, redefine `x`:

```
#a semicolon (;) can be used to separate commands
x = 8 + 3; x
x = 21; x
```

6. Variables can not only contain single values, but also vectors or matrices of values. One simple way to create a vector is to use the `c()` command:

```
#define and return vectors a and b
a = c(4.1, 6.7, 8.2, 1.8); a
b = 2*a; b
```

7. Use `mean()` and `sd()` to find the mean and standard deviation of the numbers in `a`:

```
#calculate mean and standard deviation
mean(a)
sd(a)
```

8. Plots appear in the Plots tab in the upper right. Plot the values of `a` against the values of `b`:

```
#plot a against b
plot(a, b)
```

9. R has help pages that can sometimes be useful; they typically contain a basic description and include syntax information. To look up what a certain function does, use `?`; for example, run `?mean` and the help page for the `mean` will appear in the Help tab on the bottom right.

10. The Files tab shows all the files on the local computer; the "..." icon in the upper right corner opens up the directory in a separate window, which makes it easier to browse for a particular location. Save your script file via *File* > *Save As...* in a specific destination, such as the Desktop, then close the script file. Use the Files tab in the upper right pane to navigate to where the file is saved. Clicking on the script file reopens it in the script editor.

11. Bonus: the *RStudio* workspace can be customized easily. The panels can be rearranged by going to *Tools* > *Global Options* > *Pane Layout*. Other customization options (e.g., font size, themes, etc.) are available under *Global Options* > *Appearance*.