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.

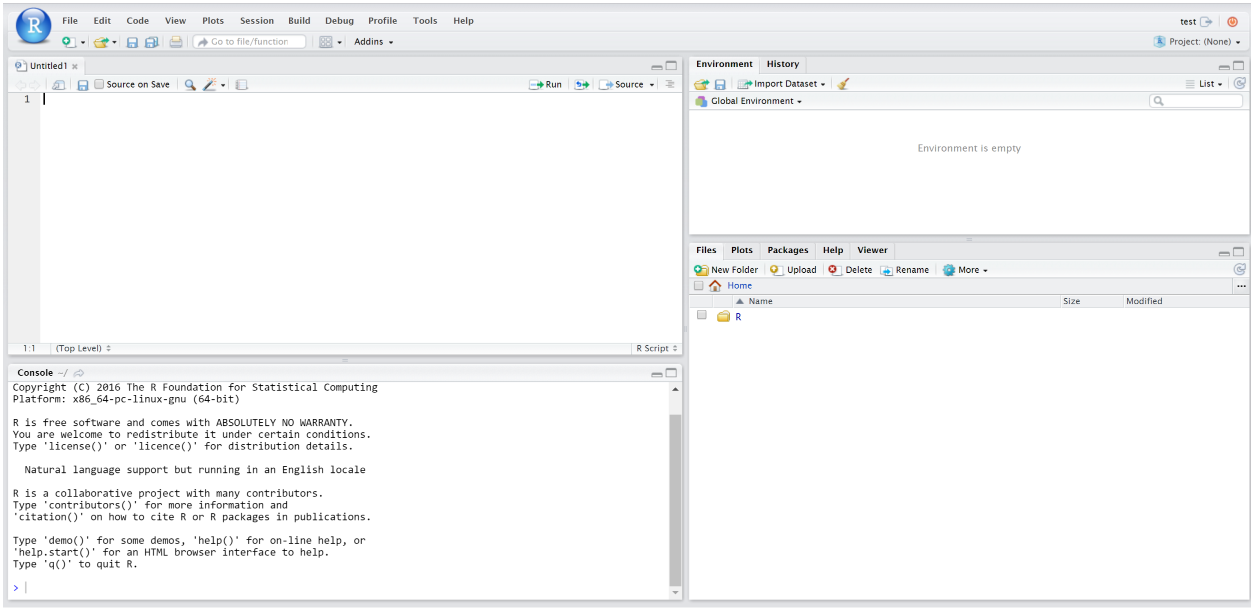
Installing R and *RStudio*

This is optional. You already have RStudio available to you via an account on the Calvin *RStudio* server. Though the next lines give some guidance, I will not provide further help in getting it installed on your local computer but, instead, will generally assume you are running it from the server through a browser window.

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



```
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

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 Quarto files (`.qmd`). 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. A Quarto 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. Quarto 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, the advantage of using the script editor is that code entered there can be saved, as well as easily edited and re-run.

2. R comes "out of the box" with many tools ready for importing, displaying, and processing data. It is also a programming language, so it offers a programmer the ability to add new functionality, or even alter the functionality of existing commands. Even end-users who do not engage in programming ourselves can benefit from the work of others through the addition of **packages**. One package we will use a lot is called **mosaic**. Enter this line in your R Script file, then click the *Run* button at the top right of the script editor to *send* the command to the console, where many messages will appear. Alternatively, with the cursor on the line of code, use the keyboard shortcut Ctrl/Cmd + Enter. On the first occasion, the messages appearing in the console are verbose; repetition of the command, however, is frivolous.

```r
# load the mosaic package
require(mosaic)
```

3. At its most basic, R can be used as a calculator. Enter `8 + 3` in the console. Then put this same code in the script editor and *Run* it. 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.

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

4. 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:

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

#calculation
x + y

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

5. Take a look at the Environment tab in the top right pane—the values of x, y, and z are displayed. Any named structure you create will appear in this tab. All such objects can be cleared by selecting the broom icon.

6. 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
   ```

7. Named structures can store not only 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
   ```

   The names a and b are assigned to containers of multiple values.

8. Use mean() and sd() to find the mean (or average) and standard deviation of the numbers in a. Note: The tilde ∼ character is located in the top left corner of your keyboard. Having loaded the **Mosaic** package, many (but not all) of the commands we use this character.

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

9. Plots appear in the Plots tab in the lower right. Plot the values of a against the values of b:

   ```
   #plot a against b
   gf_point(b ~ a)
   ```

10. 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 ?length and the help page for the length() command will appear in the Help tab on the bottom right. Having read the help, can you explain the behavior of this command?

    ```
    length(b)
    ```

11. The Files tab shows all the files in your workspace on the server. Save your script file via *File > Save As...* in a specific destination, most likely your Home directory, then close the script file. Use the Files tab in the upper right pane to see/navigate to where the file is saved. Clicking on the script file reopens it in the script editor.

12. 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*.