R & RStudio Introduction

(last updated on 03/22/2016)

Installation Notes

- OS X: Install (in this order) XQuartz, R, and RStudio.
- Windows: Install (in this order) R and RStudio.

For OS X users only, obtain XQuartz at `http://www.xquartz.org/`.

To obtain R, navigate to `https://www.r-project.org/`.

To obtain RStudio, navigate to `https://www.rstudio.com/products/rstudio/#Desktop` and select "DOWNLOAD RSTUDIO DESKTOP."

Setting Up RStudio

You will want to create a directory/folder that you can use solely for R files for this course. Use the menu bar to find the "Preferences" menu. Then select the "General" tab. Finally, type (or browse to select) the path of the default working directory. For example, first create a directory called "LB118SS16" in your "Documents" directory and then enter the path of this directory in the Preferences.

Recommended Tutorial

`http://www.cyclismo.org/tutorial/R/`

General Tips

- R is case sensitive. This means that `A` and `a` are different symbols.
- If something seems to not be working while using RStudio, try hitting the `Esc` key. For example, if your code results in an infinite loop, hit the `Esc` key to interrupt the loop.
- R commands are of the form `this_command()`. The parentheses at the end of the command are important. Some commands take one or more arguments such as `this_command(arg1=option1, arg2=option2)`.
- To find commands, you can use the autocomplete feature by hitting the `TAB` key. To cycle through previously executed commands, use the up arrow.

- To get help on a command, type `help(command)`, where `command` is the one you want to learn about. Alternatively, type `?command`. Some commands need to be enclosed in double quotes when using `help()` or `?`.

Where To Read More

`https://www.r-project.org/`

The Assignment Operation & Entering Data

- Type `x <- 2` to assign the value 2 to the variable x. Experiment with operations like $x + 1$ or x^3. What happens if you type $x < -x + 1$?
- Type `y <- c(1,2,1,3)` to assign the values of an array to the variable y. The command `c()` *combines* the elements into a *list*. To reference the 3rd entry in this list, type `y[3]`.
- Type `myfile <- read.csv(file="myfile.csv",head=TRUE,sep=",")` to assign the contents of a .CSV file to the variable $myfile$.
- To create a .CSV file, use a spreadsheet program (e.g. MS Excel) or use a text editor. If you use a text editor, each row entry is separated by a comma and the next row is indicated by starting a new line. Do not end your lines with commas. Save the file with the extension ".csv".

Shortcuts For Simple Tasks

- Type `1:20` to produce an array of the numbers 1 to 20.
- Type `seq(1,20,2)` to produce an array of the odd numbers $1, 3, \ldots, 19$.
- Type `help("seq")` to get help about the `seq()` function.
- Type `ls()` or `objects()` to list the currently defined variables.

Arrays

- Type `x <- c(1,2,3,4)` to create an array called x. You can reference it's third entry by typing `x[3]`. An array of numbers are reffered to as *vectors* in R. The `c()` command will concatenate vectors. For instance, `x <- c(x,x)` appends `x` to itself.

- Type `1/x` to print the vector whose entries are the reciprocals of the entries of x. Note: this is printed, not stored. If you want to store it, create a new variable and assign the value, e.g. type `y <- 1/x`.

R and Your File System

- Type `dir()` to list the contents of your current directory/folder.
- Type `getwd()` to list the current directory/folder.
- Type `setwd(PATH)` to set the current directory, where `PATH` is the path, e.g. `/Users/Isaac_Newton/AwesomeRCode/`. Note, on a Windows machine, you need to *escape* the backslash symbol by typing '\\' instead of '\'.

More Advanced Stuff

- Use `source()` and `sink()` to execute source files and divert output to a file, respectively.
- Separate multiple commands with a `;`.
- Comments are indicated by a `#`.
- `objects()` lists the currently stored objects (e.g. variables, arrays, functions). `rm(object1,object2)` removes `object1` and `object2`.
- Good practice: create a new directory for each R session. When you exit you have the option to save the objects in a file called `.RData`. The comand history is automatically saved in a file called `.Rhistory`.

Plotting Data and Functions

- Type `x <- (1:10)` to create a vector called `x` that contains the integers from 1 to 10. Type `y <- x^2`. Type `plot(x,y)`. Try it!
- Type `x <- seq(1,10,0.1)`. Type `y <- x^2`. Type `plot(x,y)`. Try it!
- Shortcut: type `plot(sqrt(x))`.
- Type `plot(sqrt(x), type="l")`. (The type is a lowercase letter "L".)

- Type `plot(sin(x), type="l", xlab="time", ylab="cells")`.
- Alternate way to plot: type `plot(sin,type="l",0,4*pi)`.
- The commands `points()` and `lines()` are used to add points or curves ("lines") to the currently displayed plot.
- Alternate way to generate the domain: type `x <- seq(-10,10,length = 1000)`. Then to plot the function $f(z) = 1/(1+z^2)$ on the interval $[-10, 10]$, for instance, type `plot(1/(1+x^2),type="l")`.
- Type `f <- function(x) x^2`. This will define a function called `f`. To evaluate the function at, for instance, 42, type `f(42)`.
- Type `my_cubic <- function(t) t^3 - 3*t + 1` to define the function $f(t) = t^2 - 3t$. To plot this function on the interval, say, $-2 \leq t \leq 7$, type `plot.function(my_cubic, -3, 7)`. You can also type `plot(my_cubic, -3, 7)` to achieve the same result. In both cases, however, just type the name of the function; do not include the parentheses or the variable.

Examples: Plotting a Function with Specified Domain & Range

- Type `f <- function(t) t + sin(t)` to define the function $f(t) = t + \sin t$. Type `plot(f, 0, 4*pi)` to plot $f(t)$ versus t on the interval $0 \leq t \leq 4\pi$.
- To specify the range (the y-values), use the following:
  `plot(f, 0, 4*pi, ylim = range(-5,10))`.
  To add color, use
  `plot(f, 0, 4*pi, ylim = range(-5,10), col = "blue")`.
- To add a second graph to an existing plot, use the option `add = TRUE`. For example, type `g <- function(t) t^2 * exp(-t/4)` to define the function $g(t) = t^2 e^{-t/4}$. Then type
  `plot(g, col = "red", add = TRUE)` to add the graph of g to the graph of f.