01-01: R Installation and Setup

1 - Purpose

- Download, install, and setup R and RStudio, the programs we will use for the rest of class
- Explain the components of RStudio
- Show how commenting works and the importance of commenting
- Choose a color scheme for RStudio

2 - Concepts

3 - Install programs

There are two programs you need to install for this class: **R** and **RStudio**. You will be creating scripts in R using RStudio. R is the programming language, RStudio provides a structured environment for the R programming language similar to the way Microsoft Word provides a structured environment for text editing. ***RStudio*** is patterned on other popular programming environments like Microsoft's Visual Studio.

3.1 - Download and Install R

First you need to install the *desktop version* of R.

The R for Windows download is here. Click on "Download R #.#.# for Windows".
The R for Mac download is here. Click on "R-#.#.#.pkg" on the left side of the page.

Open the file you downloaded and use the default installation options

And for those of you using Linux -- the R for Linux download instructions are here.

3.2 - Download and Install RStudio

You can download the RStudio Installer here. Download the appropriate file for your computer under *Installers*, open the file, and use the default installation options.

For Mac users there are some extra complexities:

1) You might be asked to install **Command Line Developer Tools** while installing RStudio. Go ahead and install the developer tools.
2) The download for RStudio is called ***RStudio-1.#.###.dmg***. Double-clicking the file will open the window below (*Fig 1*).

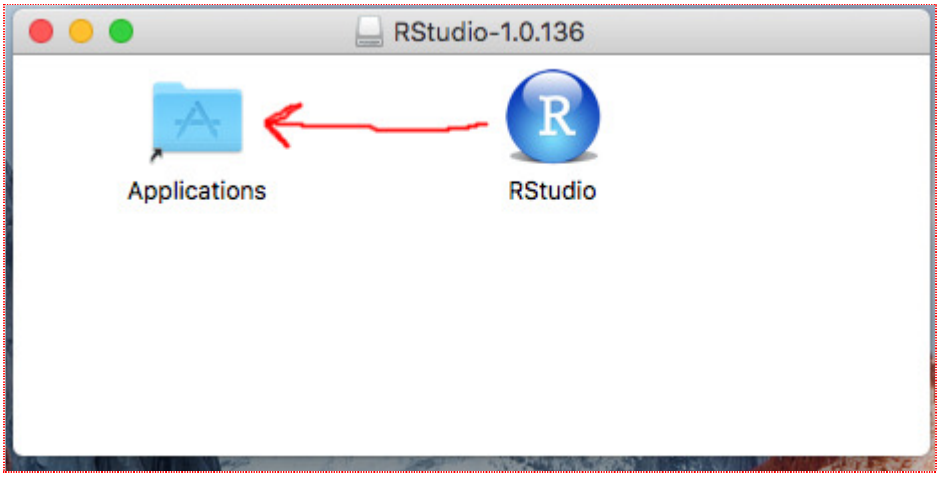

Fig 1: Opening RStudio DMG file

3) In the RStudio window above (*Fig 1*) drag the RStudio file to the Applications folder
4) **Unmount the RStudio device** by clicking the Eject button -- shown below (*Fig 2*)

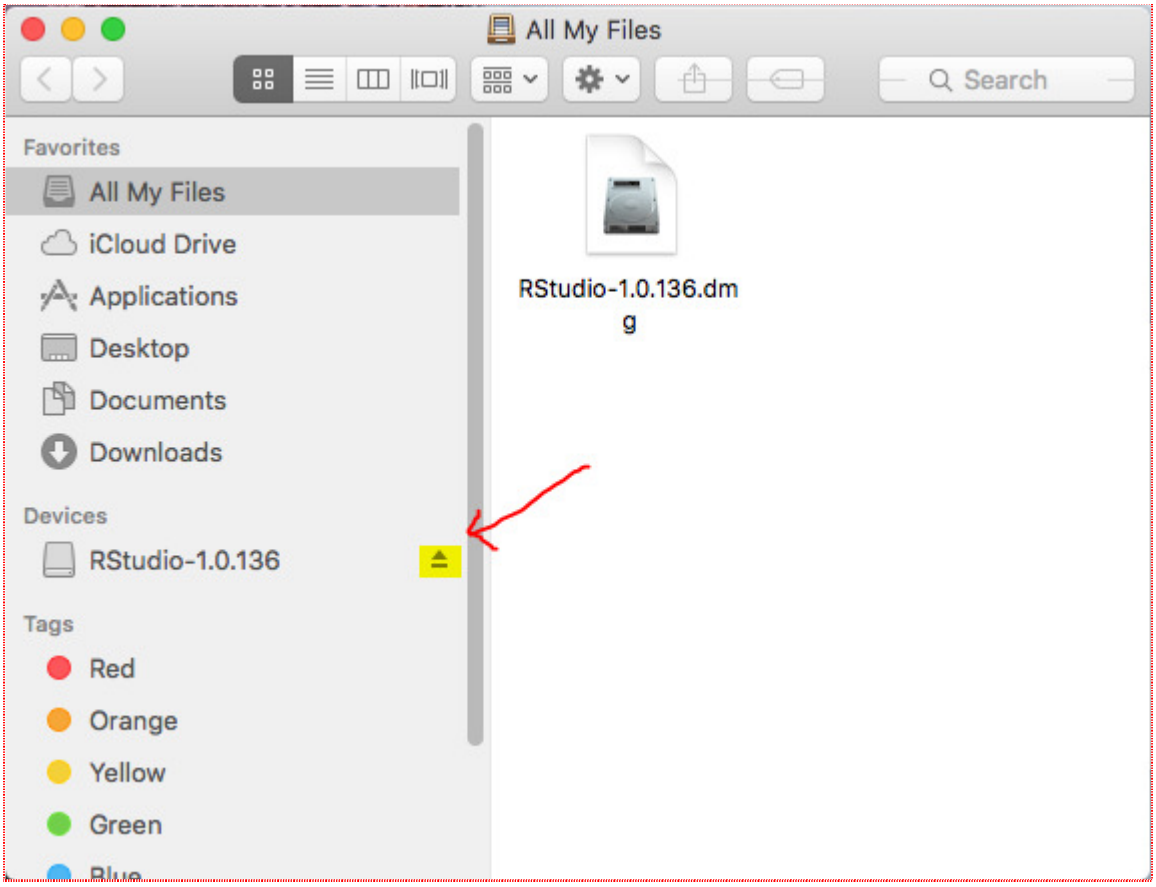

Fig 2: Unmounting the RStudio device (very important!)

4 - RStudio Basics

The picture below (*Fig 3*) shows RStudio when you first open it up and without a script file. This is a view you will rarely see because you will almost always have a script file opened.

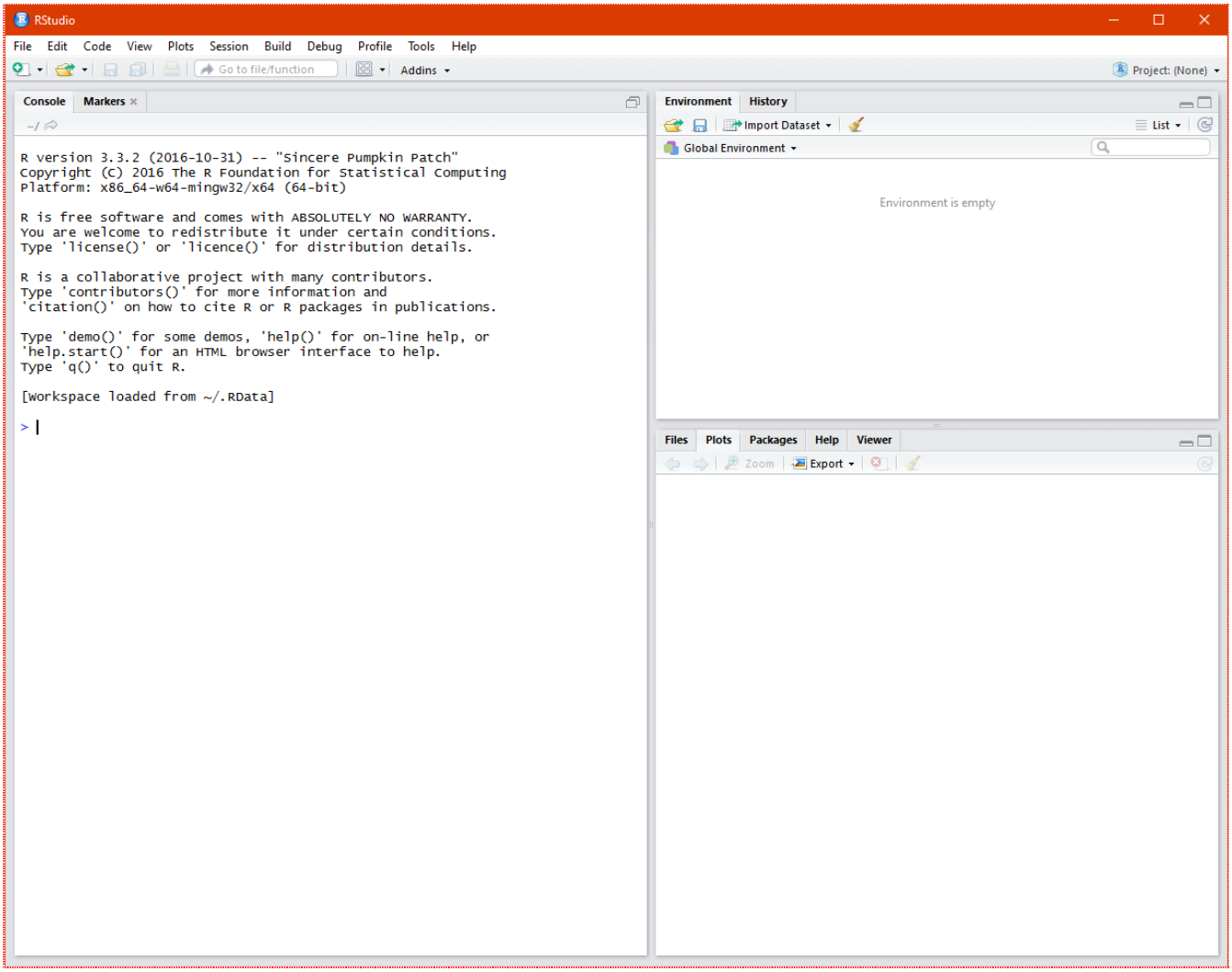

Fig 3: First time opening RStudio

4.1 - Open your first script file

We are going to execute a script that performs a linear regression on fish lengths over a two year period. The code in the script is something you will progressively learn throughout the course -- for now we are executing the script to help you become familiar with the RStudio environment.

You will need to download two files.
1) the data in a csv file (Intro-linearModel.csv), which represents fish lengths from two consecutive years
2) the R script file (Intro-linearModel.r), which executes a linear model using the data in the csv file above
It does not matter where you save the two files as long as you remember where you put them (the desktop is always a convenient location to save files).

Note: You might need to resave the two files because browsers, for security reasons, will save downloaded files as read-only.

First you will open the R script file in RStudio. To do this you can either:

- Double-click on the file, ***Intro-linearModel.R*** or
- In RStudio click ***File -> Open File ->*** and then find ***Intro-linearModel.R*** in the file browser and click ***Open***.

After opening the script file, you should see something that looks like this (*Fig 4*) on your screen...

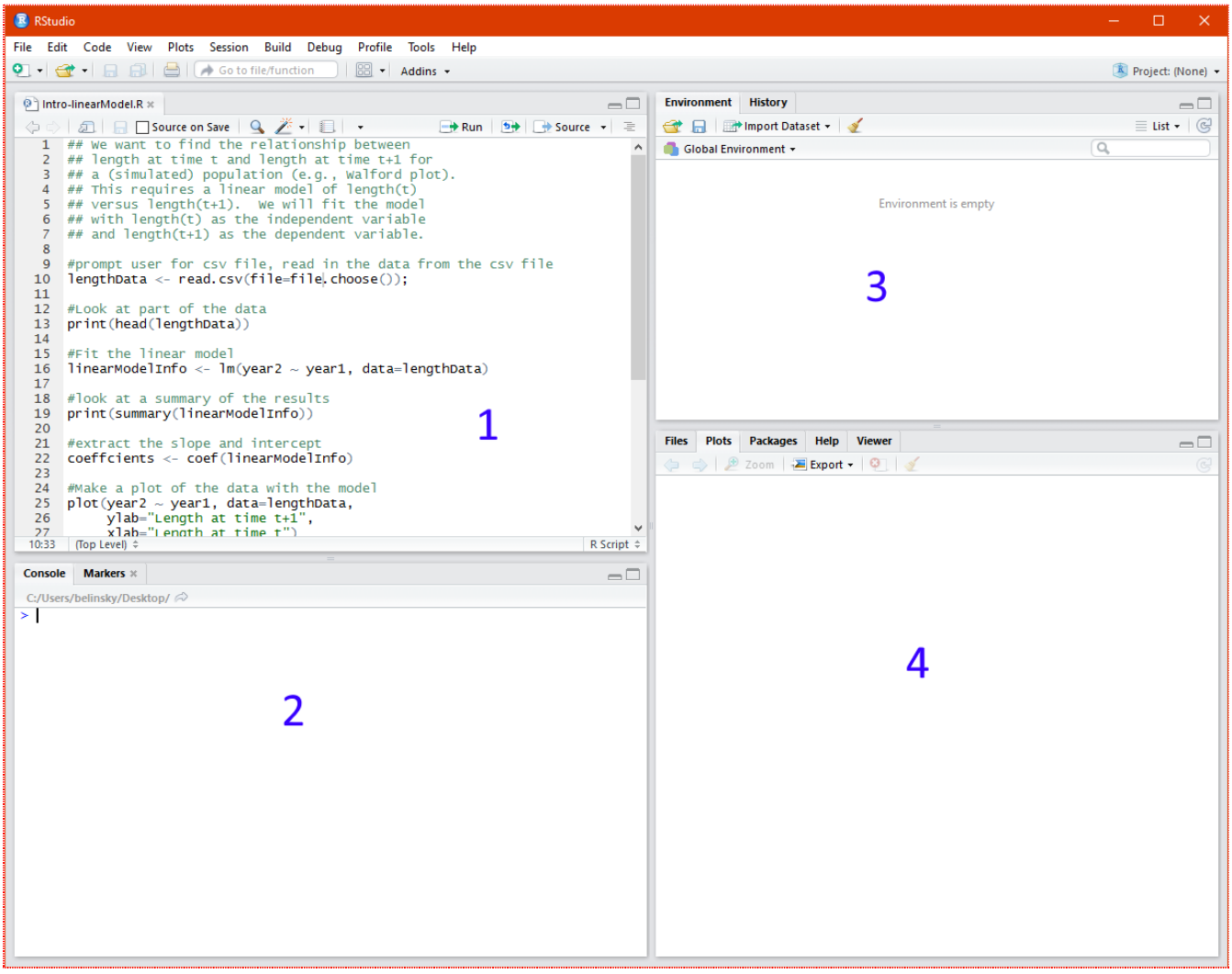

Fig 4: Opening your first script in RStudio

The four windows in the picture above (*Fig 4*) are numbered:
1) **Editor** -- text editor for the opened script files
2) **Console** -- displays information about the execution of your script file
3) **Environment** -- displays data points, or variables, from the execution of your script file
4) **Plots** -- plots produced by the execution of your script file are displayed here

Notes:
- The ***Help*** tab in the **Plots Window** is something you might find useful. *Extension: The Help Tab*
- The **Console, Environment, and Plot Windows** have other tabs you can open but we will not be using these tabs for a while.

4.2 - Common buttons used in RStudio

The script file you opened is a fully functioning program that takes fish data from a Comma Separated Value (CSV) file, ***Intro-linearModel.csv***, and plots out the data -- the code will be explained later. I am going to use this program to demonstrate a few of the very useful buttons in RStudio.

The one button you will use the most is "Source". This is the button that execute your whole script.

Extension: Run vs. Source

Press the ***Source*** button (*Fig 5*). The program asks you to locate the data file (csv) you just downloaded. Navigate the file browser to the csv file and click ***Open***.

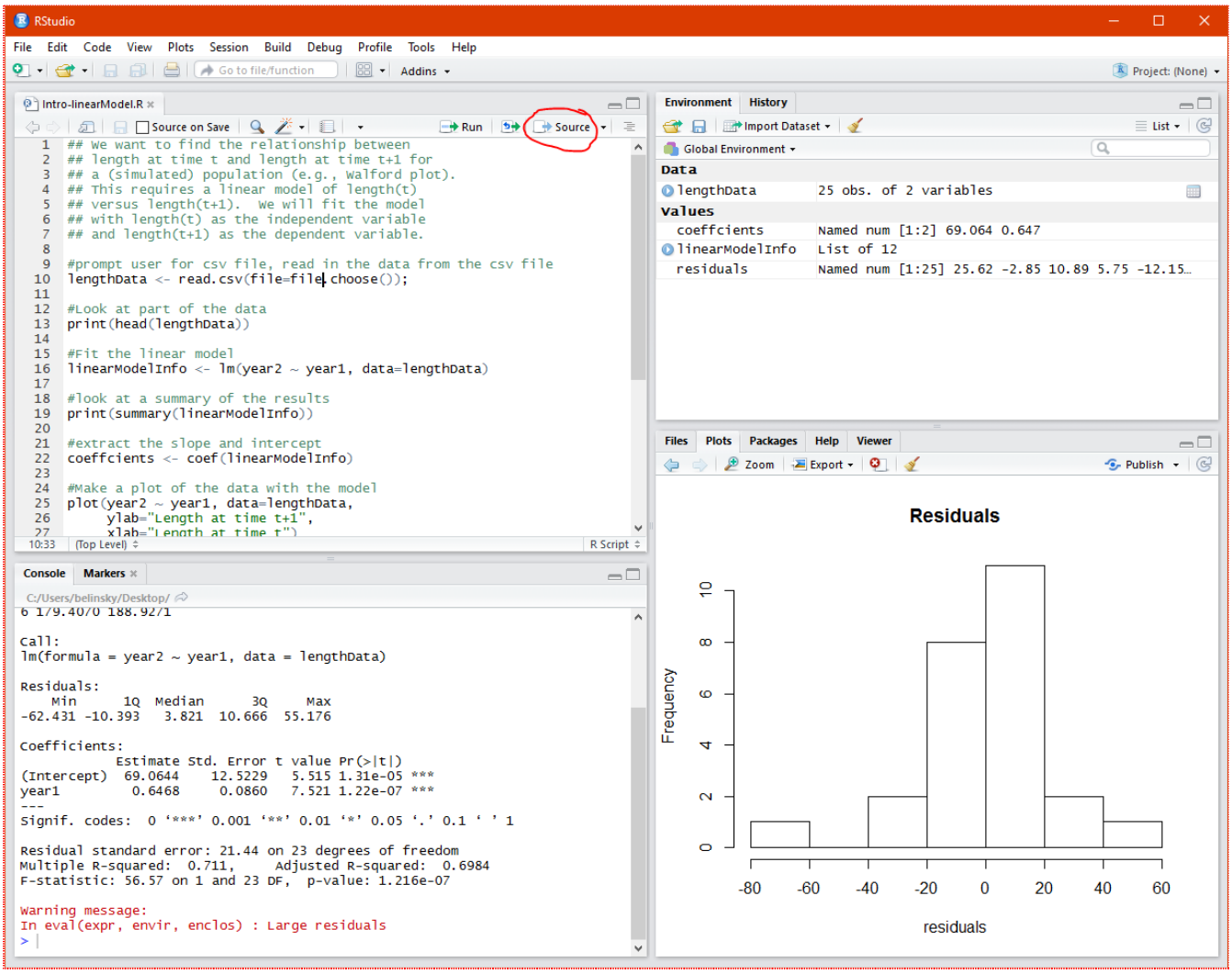

Fig 5: Running the script

After the script is run:

- The **Environment Window** displays values for the data (variables) in the script (e.g., ***lengthData***, ***cofficients***, ***residuals***).
- The **Console Window** displays information about the execution of the script along with a summary of the linear model.
- The **Plot Window** displays two plots. You can use the arrow buttons (*Fig 6*) to switch between the plots.

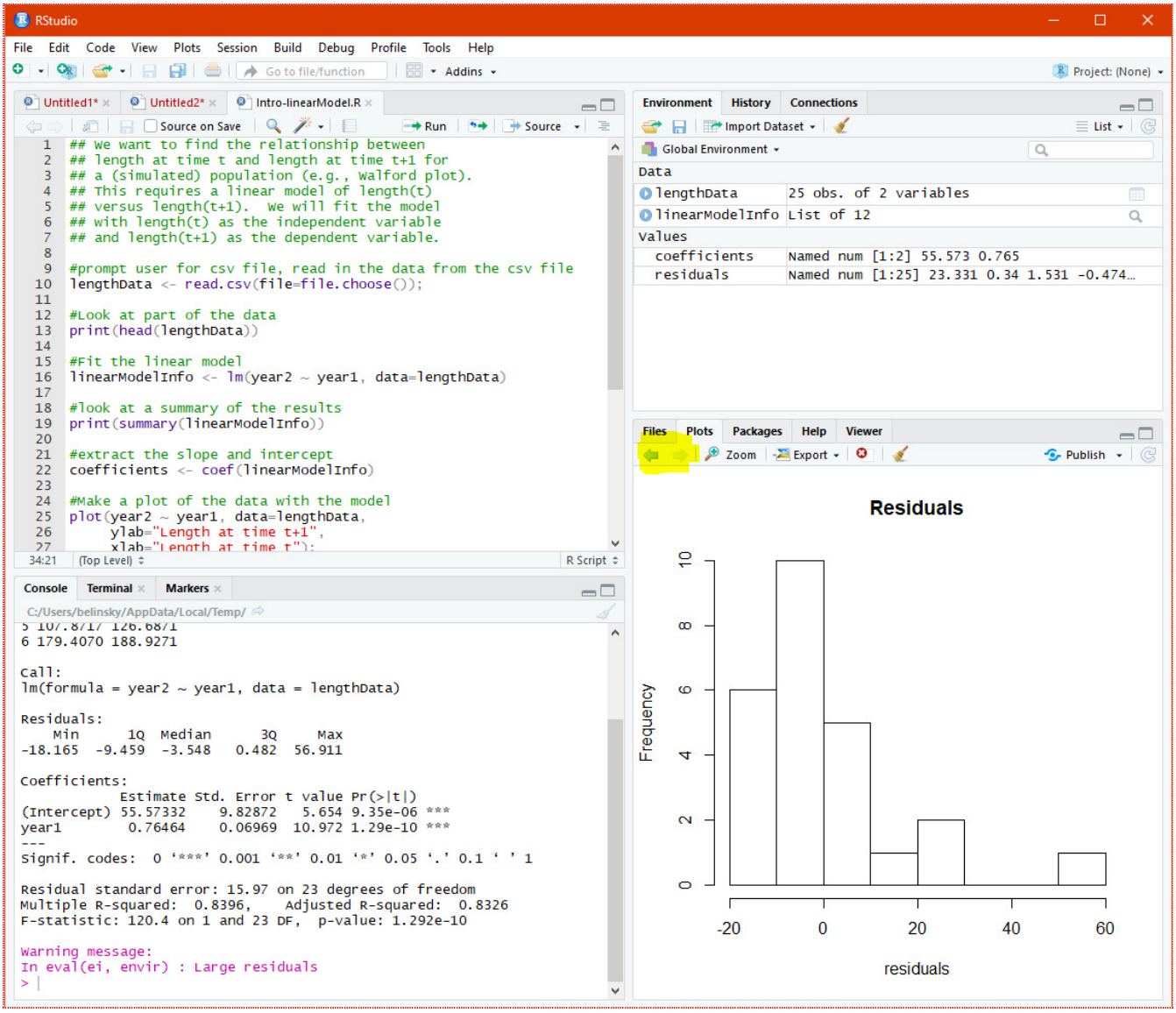

Fig 6: Using the arrow keys in the Plot Window

4.3 - Cleaning up the RStudio windows

There many times where you want to clean up the windows, which can get very crowded with old information.

- To clean the **Environment** and the **Plot Windows**, use the brush button (*Fig 7*).
- To clean the **Console Window**, press Control-L (or Command-L on a Mac) while in the **Console Window.**

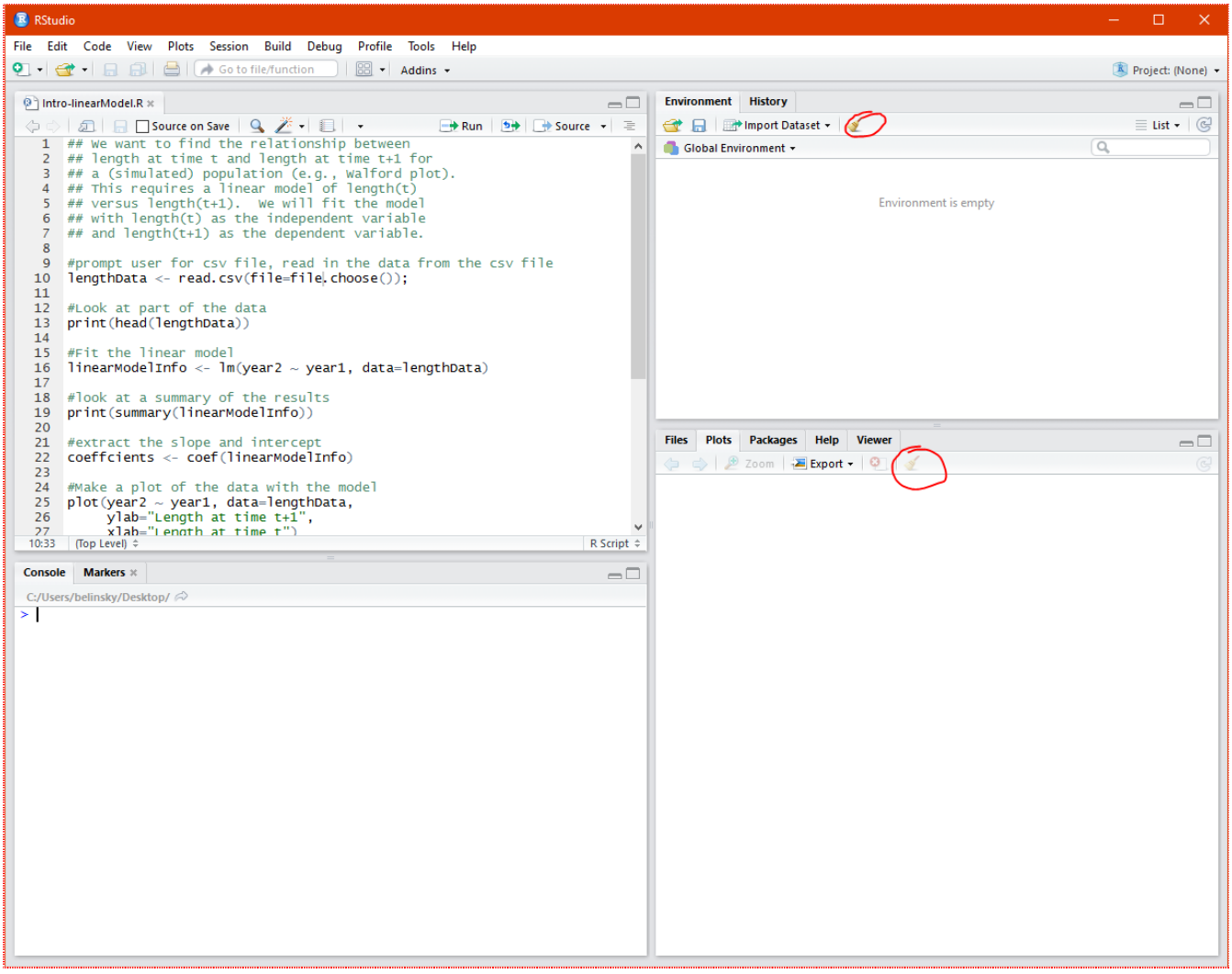

Fig 7: Cleaning out the windows

If you click ***Source*** again, the **Environment, Plot,** and **Console Windows** will once again be populated with data from the script.

5 - Your first programming topic: Commenting

We are not going to spend any time on the R code itself in this lesson. But I want to focus on an important feature of all programming languages called *commenting*. Comments are text in a script that do not get executed -- the text is there to explain to the reader what is happening in the script. In other words, it allows programmers to explain their code within the script file. Notice how the functionality of the script can be generally understood just by reading the comments -- this is the main purpose of commenting.

In the R language, comments are created using the number-sign (**#**) key. Any text on the line after the (**#**) will be treated by R as a comment and it will not be executed. *You are required to make extensive use of comments for your project in this class.* Aside from being a good practice, it significantly reduces the frustration level of people who are trying to evaluate your work!

6 - Color Schemes

Notice how RStudio uses color to delineate comments (*Fig 8*: lines 9,12,15,18,21, etc..). The comments below are in a lighter green whereas most of the other text is in black.

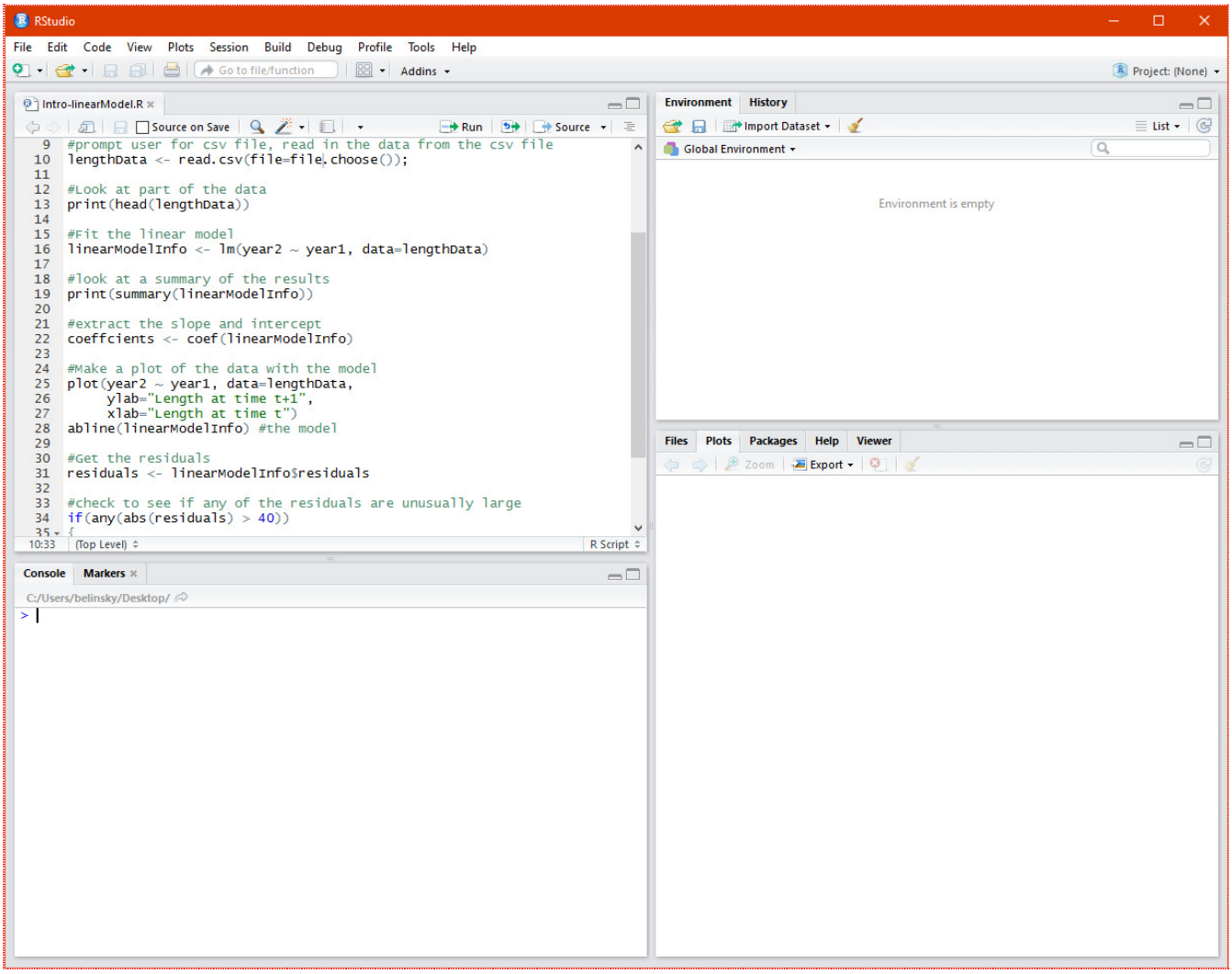

Fig 8: Comments are in the lighter green color

I am not a big fan of the default color scheme in RStudio. It does not create enough differentiation between the different components of a script. For instance, comments (red arrows) are in green and quoted items (blue arrows) are in just a slightly different green (*Fig 9*).

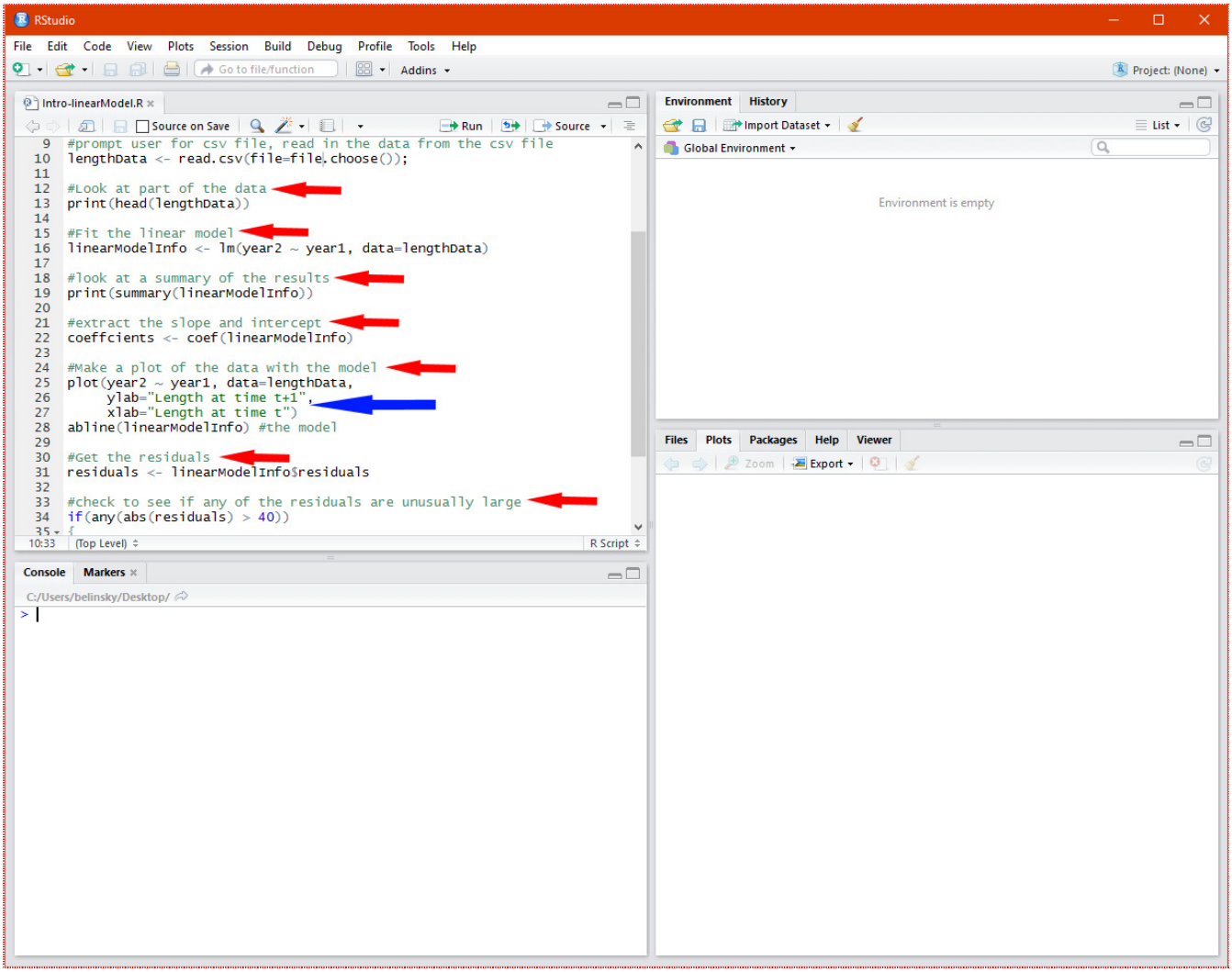

Fig 9: The text color for comments and quotes

A good color scheme can really help a programmer by allowing them to quickly identify parts of a script and common errors, like misplaced quotes.

RStudio offers many color schemes: to change the color scheme (*Fig 10*)
1) click on "Tools" in the main menu (circled in *Fig 10*)
2) choose "Global Options"
3) When the "Global Options" window open (in *Fig 10*), click on "Appearance"

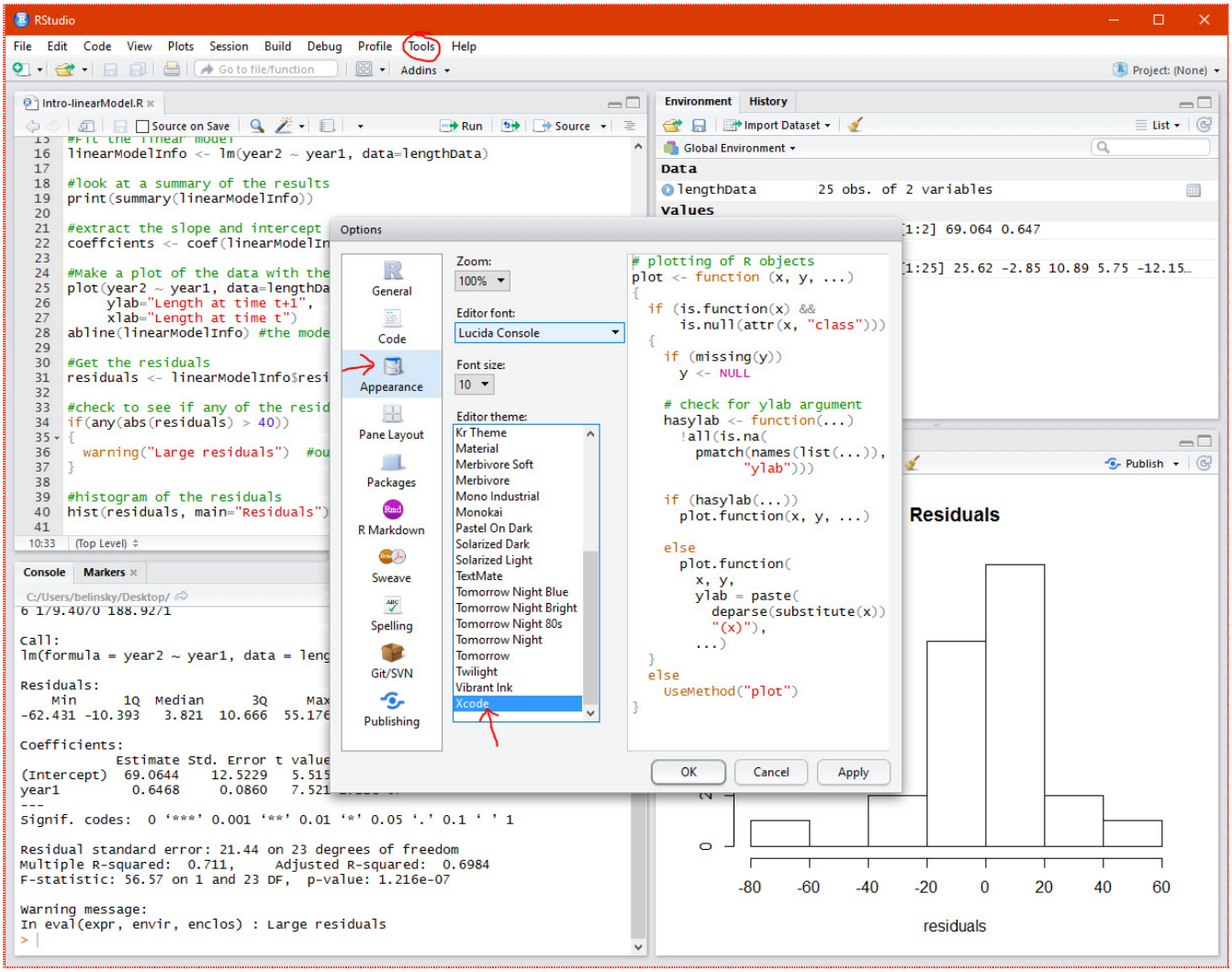

Fig 10: Color schemes for RStudio (note: the Editor font might be different in your window -- that is OK)

The picture above shows the *Xcode* color scheme (*Fig 10*). I prefer **Xcode** because it does a good job differentiating the different aspects of the script. Notice how the comments (in green) are now clearly distinguished from the quotes (in red).

You can choose from around 20 themes in the **Editor theme** window and you can change the theme anytime without affecting anything else.

If you want to want to use the exact same color scheme that I use in the class, which is a slightly modified Xcode theme, follow the instructions in:
Extension: Customize Scheme

6.1 - Adding more color to differentiate output

There are a couple more options you can set in RStudio that add more color variation in the **Console Window**. This is very helpful in the long run because it will help you distinguish the different types of output in the **Console Window**.

To make these changes go to ***Tools -> Code -> Display*** and check the following boxes:

- *Show syntax highlighting in console input*
- *Highlight R function calls*

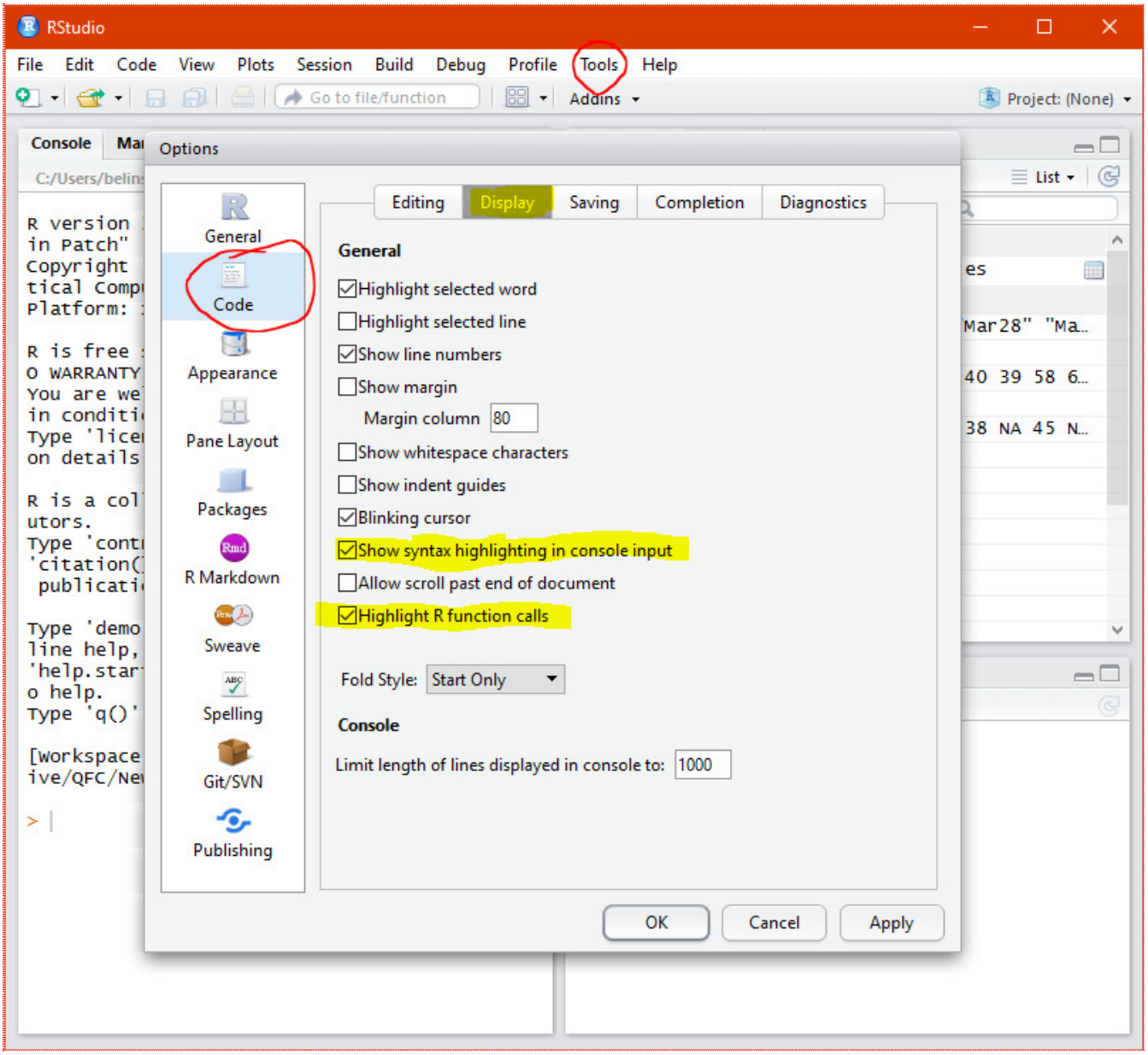

Fig 11: Console Window changes

7 - Stop RStudio from automatically adding matching parenthesis and quotes

Finally, a common complaint I have gotten from my students is they hate the way RStudio tries to be "helpful" by automatically adding end parenthesis and end quotes when the user types in a start parenthesis or start quote.

You can turn off this feature by going to ***Tools -> Code -> Editing*** and *uncheck* "Insert matching parens/quotes"

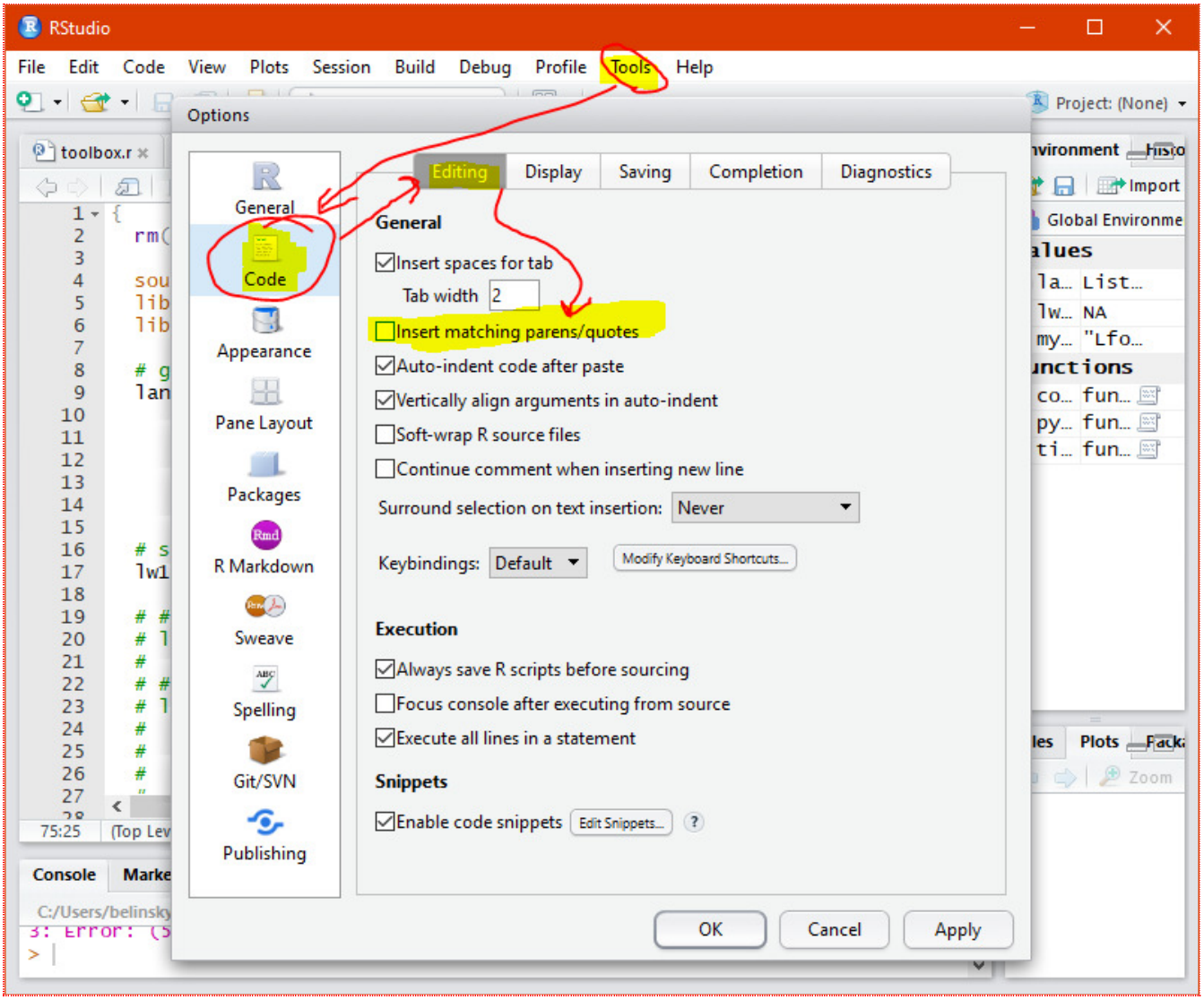

Fig 12: Stop RStudio from automatically ending your parenthesis and quotes.

8 - Extension: Install customized color scheme

You might have noticed that the Xcode theme in (*Fig 10*) is slightly different than the Xcode theme on your computer. Specifically, the ***if***, ***else***, and ***warning*** statements are colored orange as opposed to purple. I did this to differentiate these statements from the output in the console. Unfortunately, this is not an easy thing to change in RStudio.

If you would like to use my color scheme, which is the color scheme I will use for the whole class, then you need to:

1) choose "XCode" as the color scheme
2) download my color scheme file
3) save my color scheme file to the proper directory (explained below)

8.1 - Saving my color scheme file to the proper directory:

Note: the downloaded filename is **DA6683C1A9EFEF24EAE992398513B1C4.cache.css** and the name cannot change.

On a Windows computer (you need to be logged on as an admin of the computer):

1) Go to the following folder: ***C -> Program File -> RStudio -> www -> rstudio***
   Or: copy and paste *C:\Program Files\RStudio\www\rstudio* into the top search bar of a File Explorer window
2) Copy the file (***DA6683***...) into the current folder (overwriting the old file)
3) If asked: give the administrative privileges.

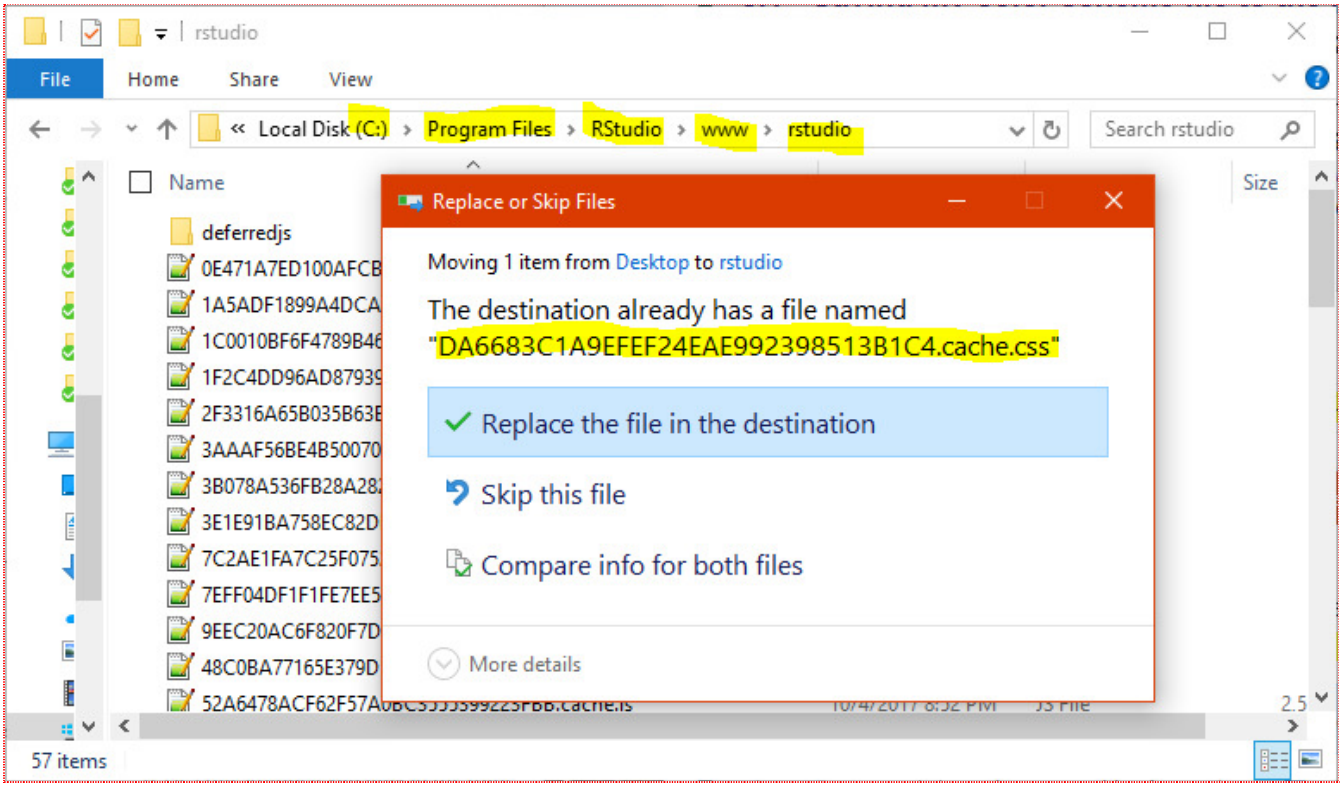

Fig 13: Adding the color scheme file to the appropriate folder in Windows

On a Mac (you need to be logged on as an admin of the computer):

1) Go to the ***Applications*** folder
2) right-click (or control-click) on the ***RStudio*** application
3) choose "Show Package Content"
4) Go through the following folders: ***Content -> Resources -> www -> rstudio***
5) Copy the file (***DA6683***...) into the current ***rstudio*** folder (overwriting the old file)
6) If asked: give the administrative privileges.

Restart RStudio and the color change will take effect -- you will see the added orange color in the code (like *Fig 8*)

9 - Extension: The Help Tab

The Help Tab effectively acts as an online search through the R documentation. So, if you type in "plot" in the search bar and hit enter, the R plot help page from the online documentation will appear. Note: you could have done the same thing by typing "?plot" in the **Console Window.**

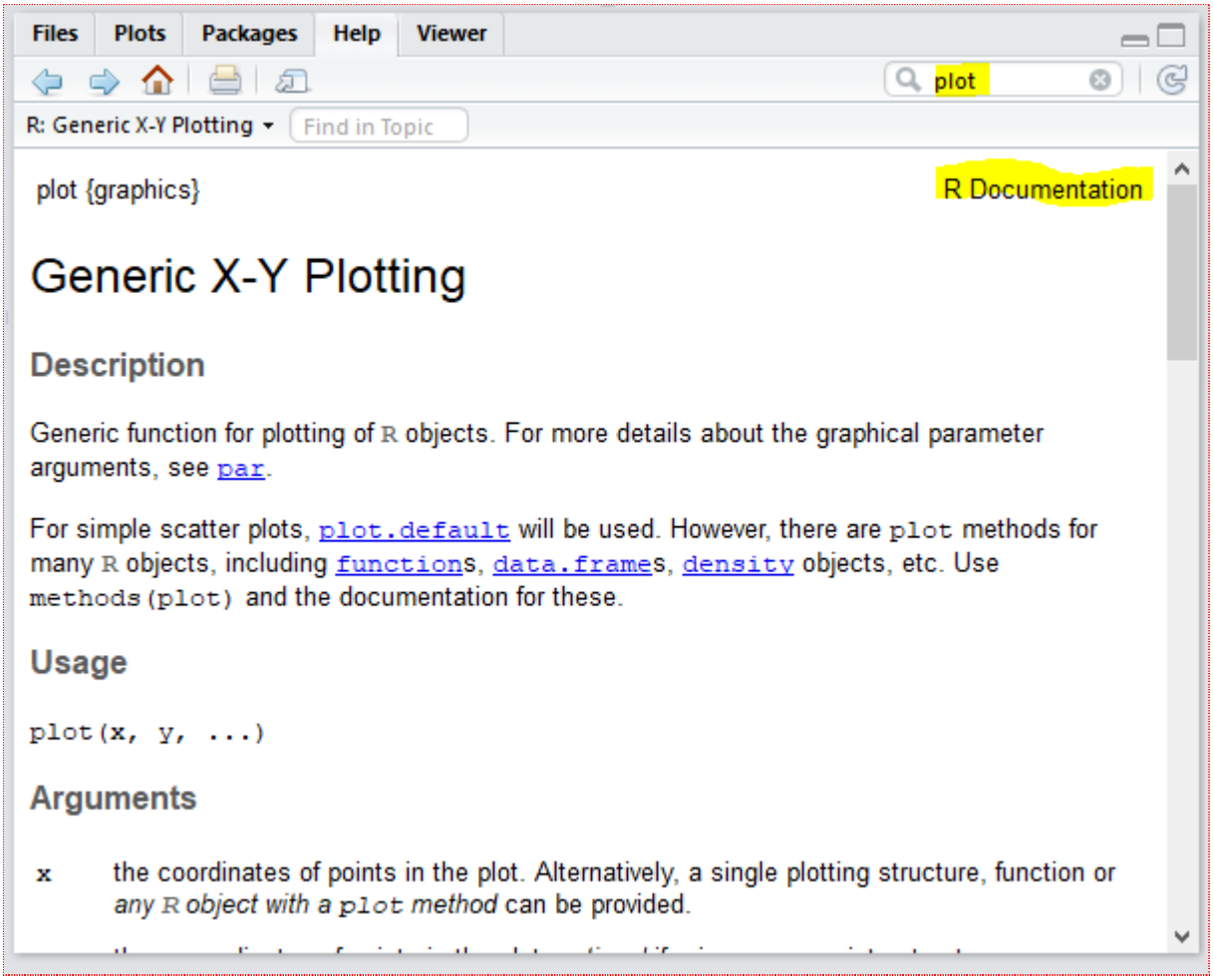

Fig 14: The R plot help page in the online documentation.

The page that appears in the **Help Window** (*Fig 14*) is this page: https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/plot.html

https://stat.ethz.ch is where the official documentation for R is located. So, you will see this website appear quite often when you do an internet search for something R related.

10 - Extension: Run vs. Source

Technically speaking, the difference between ***Run*** and ***Source*** is:

- ***Source*** will execute all the code in a script file.
- ***Run*** will execute only the line that the cursor is on or all lines that are highlighted.

The real difference lies in a historical discussion of scripting vs. programming, which is a discussion beyond this class. Suffice to say, R was originally intended to be a quick-and-dirty scripting language where users could immediately pull in data and produce a plot or perform statistical analysis. Using this method, an R-user would produce and execute one line of code at a time using the equivalent of the ***Run*** button.

As R has grown, the focus has shifted into developing well-structured code that can be easily shared, tweaked, and reused -- similar to a modern programming language like C. Using this method, an R-user would produce a full script and execute it all using the equivalent of the ***Source*** button. This is the method we will be using in this class.

RStudio is an open source project that is designed to provide a structured environment for R-users.