01-01: RStudio Projects

1 - Purpose

- Explain the components of RStudio
- Set up your Class Project using RStudio Projects
- Show how commenting works and the importance of commenting
- Choose a color scheme for RStudio

2 - Questions about the material...

If you have any questions about the material in this lesson *feel free to email them to the instructor here*.

3 - Install software on your computer

We are going to install two programs on your computer: **R and RStudio**. R is a 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.

Note: Even if you already have these programs installed, it is a good idea to verify you have the latest version.

Installing the programs:

1. **Install (or update) 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.

2. **Install (or update) 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.

   *Alternatively: to update RStudio, in RStudio go to **Help** -> **Check for Updates***

   Extension: Special instruction for Mac users

4 - RStudio Projects

R is a programming language, RStudio is a programming environment. RStudio provides easy access to the most common R functions kind of like Microsoft Word provides easy access to the most common text editing functions.

We are going to set up an RStudio Project, which will contain all the work you do in this class.

The steps for creating an RStudio Project are:

1. Create a folder on your computer, which we will call the *Root Folder*, and add subfolders to it.
2. Save files in the folders you just created.
3. Link an RStudio Project to the Root Folder.

4.1 - Create a Root Folder with subfolders

The first thing you need to do is create a folder on your computer that will hold all the material for this class. The name and location of the folder is up to you, in *Fig.1*, I called the folder ***R Root***. Inside the Root Folder, add two subfolders: ***data*** and ***scripts***.

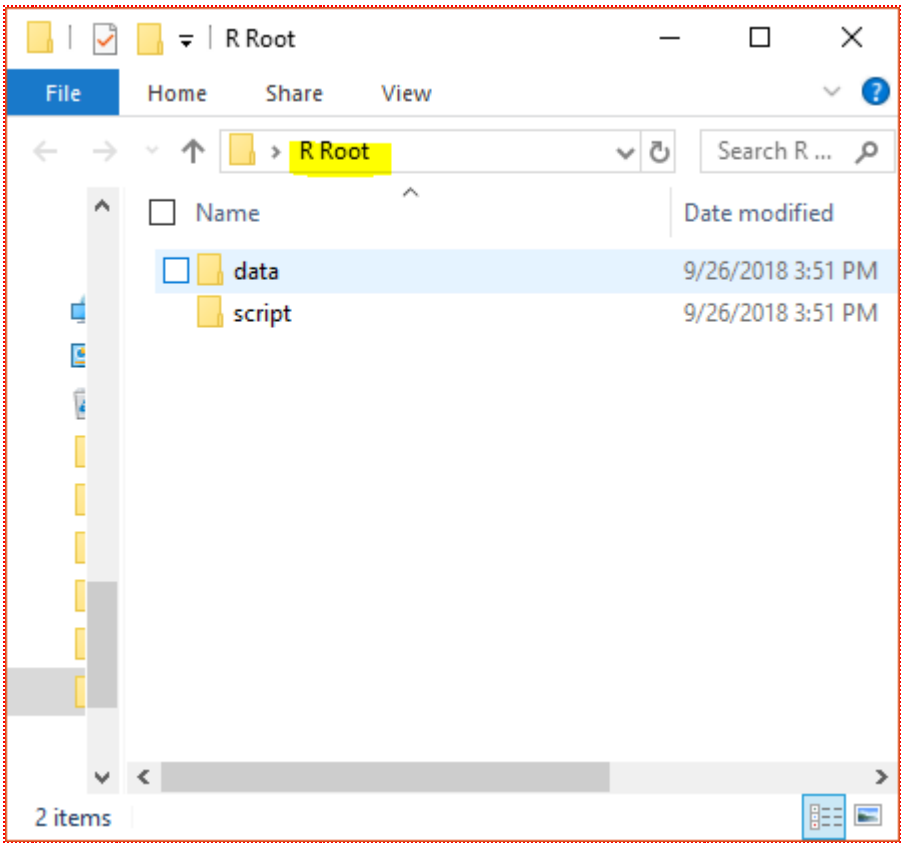

*Fig 1: My Class Folder, called **R Root**, and its subfolders*

4.2 - Add files to the Root Folder

There are three files we are going to download and save to the Class Folder:

1. The Class Project Rubric -- save this to the Root Folder (not a subfolder)
2. A data file in comma-separated-value format (linearModel.csv) -- save this to the **data** subfolder
3. An R script file (linearModel.r) -- save this to the **scripts** subfolder

So you should have:

- A root folder, which I called ***R Root***. Inside the Root folder is:
  - **ClassProjectRubric.docx** file
  - ***data*** folder -- inside the ***data*** folder is:
    - **linearModel.csv** file
  - ***scripts*** folder -- inside the ***scripts*** folder is:
    - **linearModel.r** file

4.3 - Link RStudio to the Root Folder

We are going to create an RStudio Project that is linked to the Root Folder you just created. The main purpose of the RStudio Project is to encapsulate all your class work within the Root Folder and its subfolders.

To create an RStudio Project at the Root Folder:

1. In RStudio choose ***File*** -> ***New Project...***
2. In the ***Create Project*** window choose ***Existing Directory***
3. In the ***Create Project from Existing Directory*** window (*Fig 2*) click ***Browse***
4. In the ***Choose Directory*** window navigate to your Root Folder and click ***Open*** (in this example, ***R Root***)
5. In the ***Create Project from Existing Directory*** window click ***Create Project***

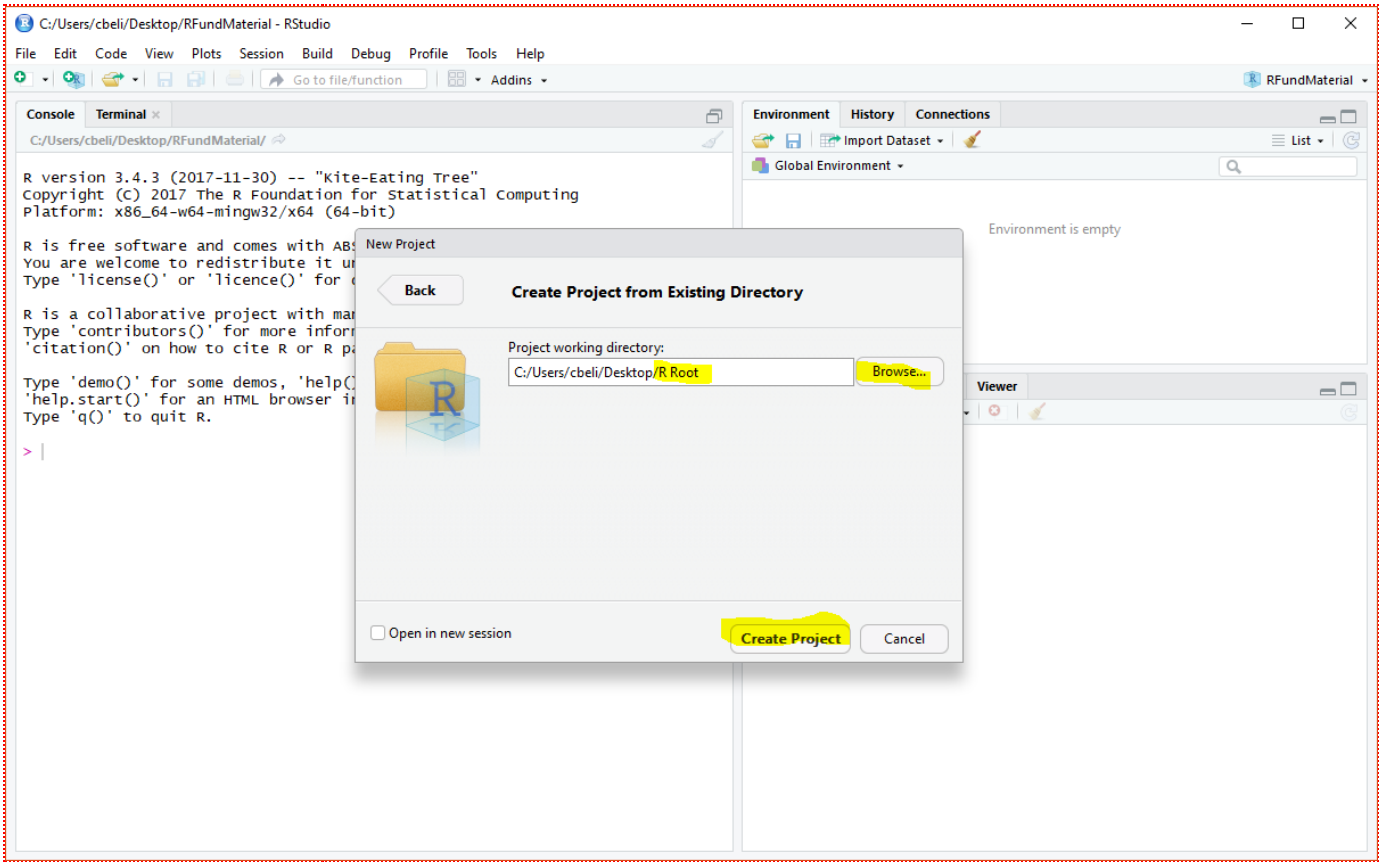

Fig 2: The Create Project from Existing Directory window

5 - RStudio Project File Manager

You should now be running RStudio opened to your RStudio Project that you just created. If you click on the ***Files*** tab in the lower-right corner, you will see all the folders and files within your RStudio Project, which is linked to your Root Folder. *The **Files** tab is a File Manager* (*Fig.3*) -- it can be used to open, add, remove, or rename files and folders.

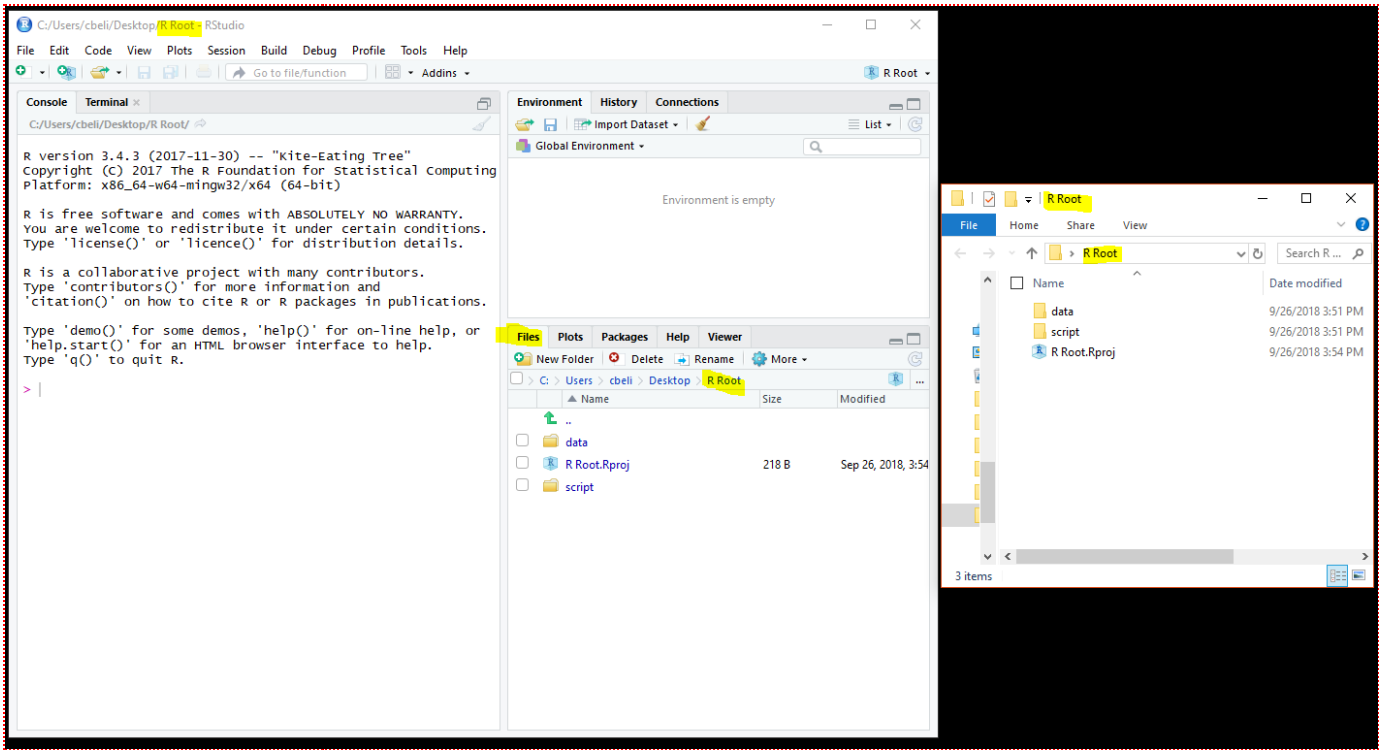

Fig 3: The Files tab and the File Manager -- both show the same thing.

The file ***<Root Folder>.Rproj*** was added by RStudio to your Root Folder. In your file explorer is set to see hidden file, then you will see a second file named ***.Rhistory***. These are not files you will need to use right away but you can learn more about them at *Extension: Files Added by RStudio*

6 - RStudio Basics

You should have RStudio opened to the Class Project you just created. In this section, we are going to execute the script file that we just downloaded, ***linearModel.R***. The script performs a linear regression on fish lengths over a two year period -- the data for the fish lengths is in the other downloaded file, ***linearModel.csv***. 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.

6.1 - Open your RStudio Project

There are multiple ways to open an RStudio Project:

- Open your Root Folder in a file manager and double-click the ***.proj** file
- In RStudio, click ***File*** -> ***Open Project...*** -> navigate to the Root Folder and click the ***.proj** file
- In RStudio, click ***File*** -> ***Recent Projects*** -> choose the RStudio Project you just created

6.2 - Open a script file in your Project

We are going to execute the script file that we downloaded earlier. To open the script file either:

- In ***Files*** tab, go to the ***scripts*** folder and click on ***linearModel.R*** or
- In RStudio click ***File -> Open File ->*** and then find ***linearModel.R*** in the ***scripts*** folder and click ***Open***.

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

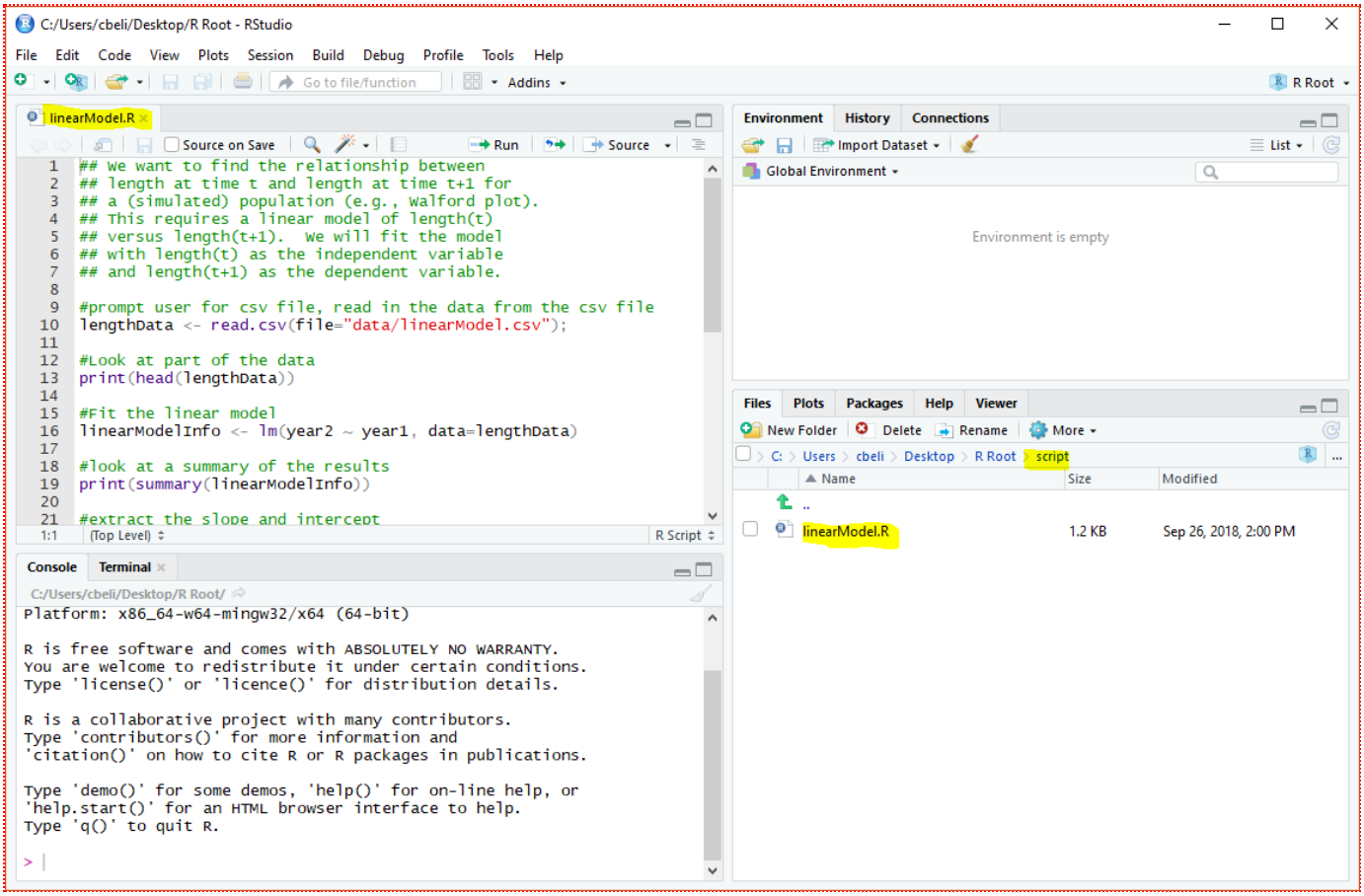

Fig 4: Opening your first script in RStudio

6.3 - Setting up RStudio tabs for scripts

When we are editing and executing script, we generally have the following RStudio tabs open (*Fig.5*):

1. **Editor** -- text editor for the opened script files (upper-left corner)
2. **Console** -- displays information about the execution of your script file (lower-left corner)
3. **Environment** -- displays data points, or variables, from the execution of your script file (upper-right corner)
4. **Plots** -- plots produced by the execution of your script file are displayed here (lower-right corner)

The fourth (lower-right corner) window is on ***Files***, so we will switch tabs to ***Plots***.

This will also be the common tab placement for the images shown in this class.

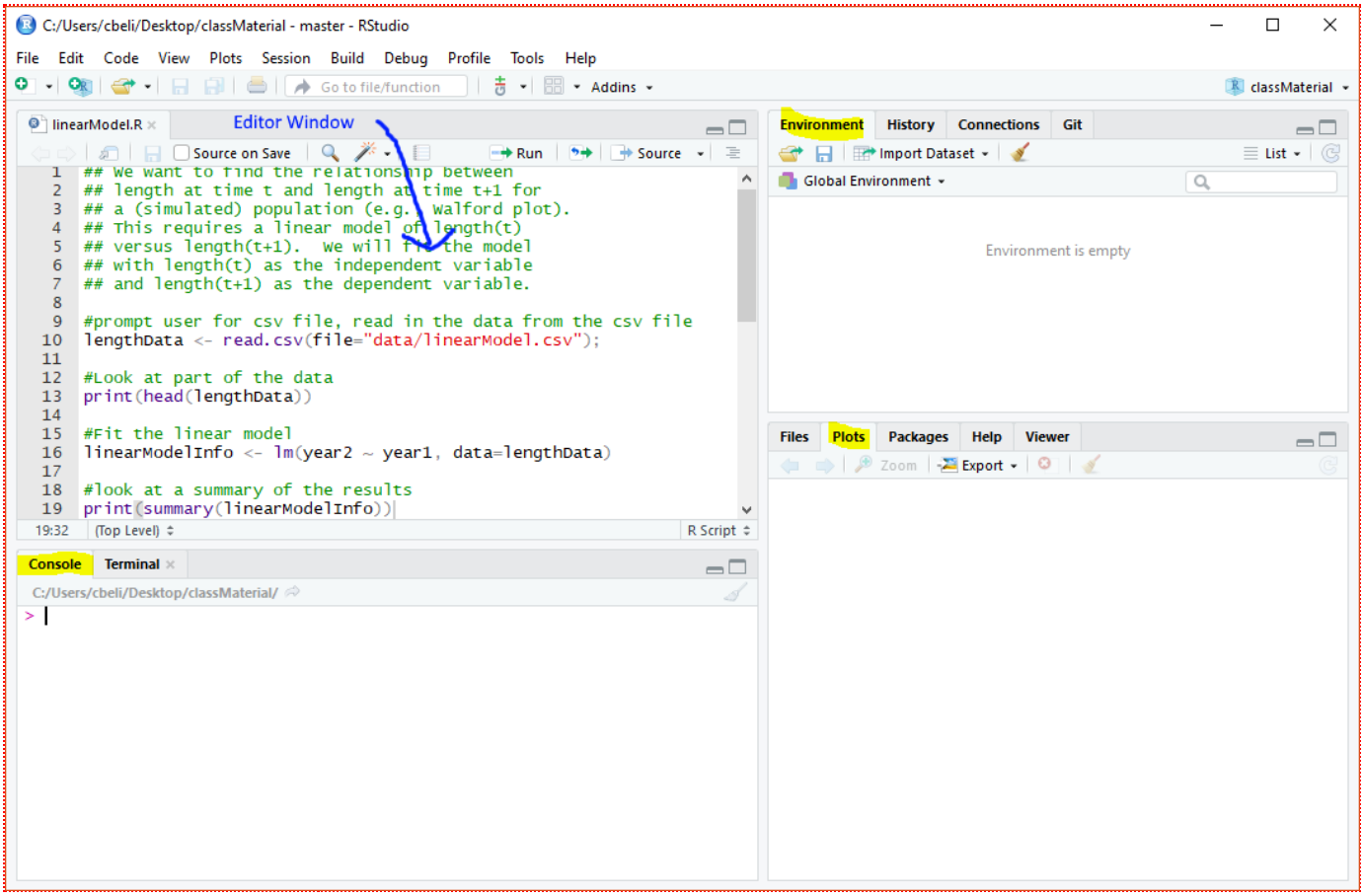

Fig 5: The Tabs most commonly used in RStudio

Note: the ***Help*** tab in the lower-right corner is something you might find useful. *Extension: The Help Tab*

6.4 - Common buttons used in RStudio

The script file, ***linearModel.r***, is a fully functioning program that takes fish data from the Comma Separated Value (CSV) file, ***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 executed your whole script.

*Note: If you have already used R, there is a good chance you highlight lines of code and click **Run** to execute just those lines. We are not using this method in this class. You should always use Source to execute you code. Extension: Run vs. Source*

Press the ***Source*** button (*Fig.6*).

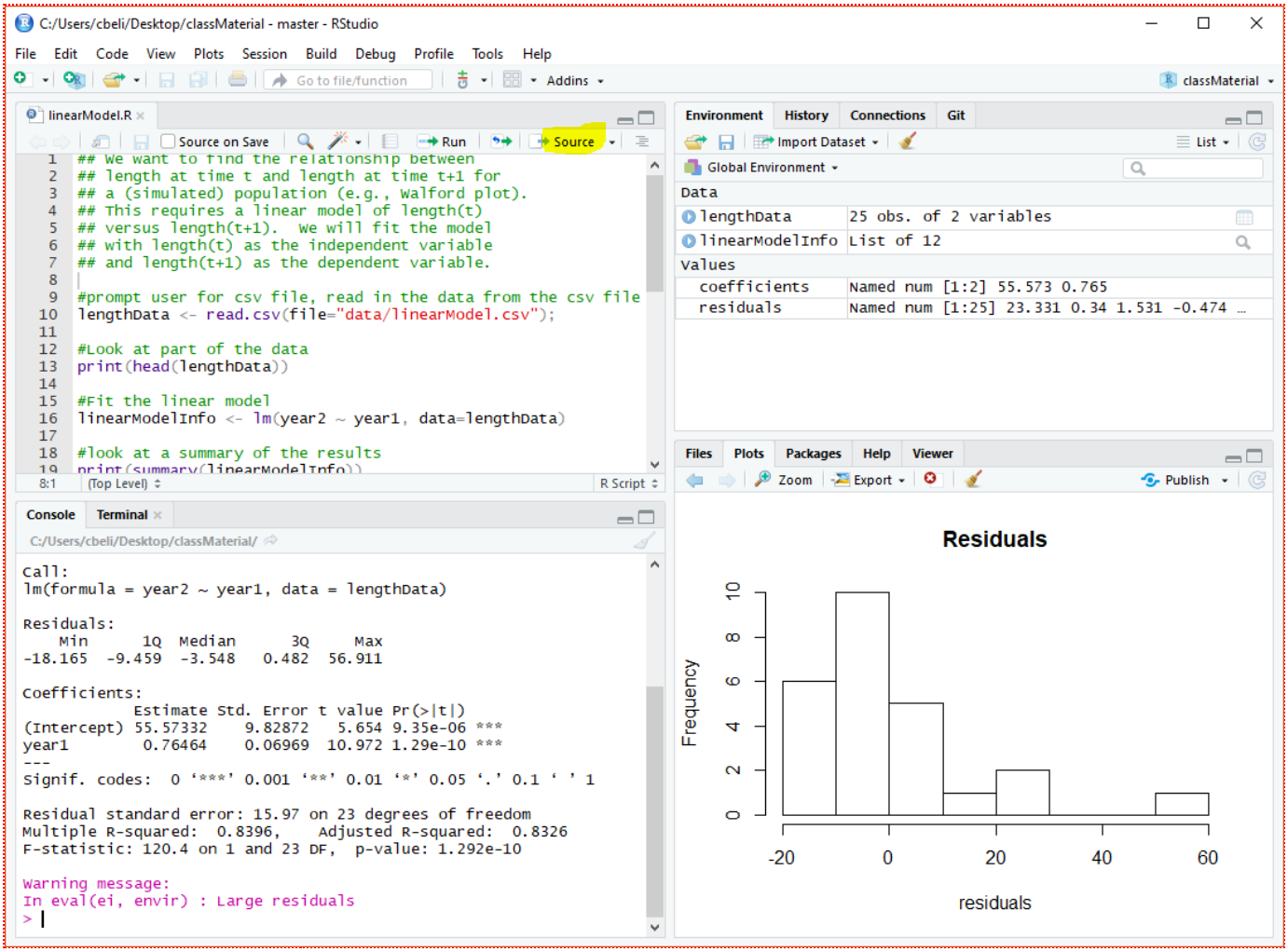

Fig 6: Running the script

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.7*) to switch between the plots.

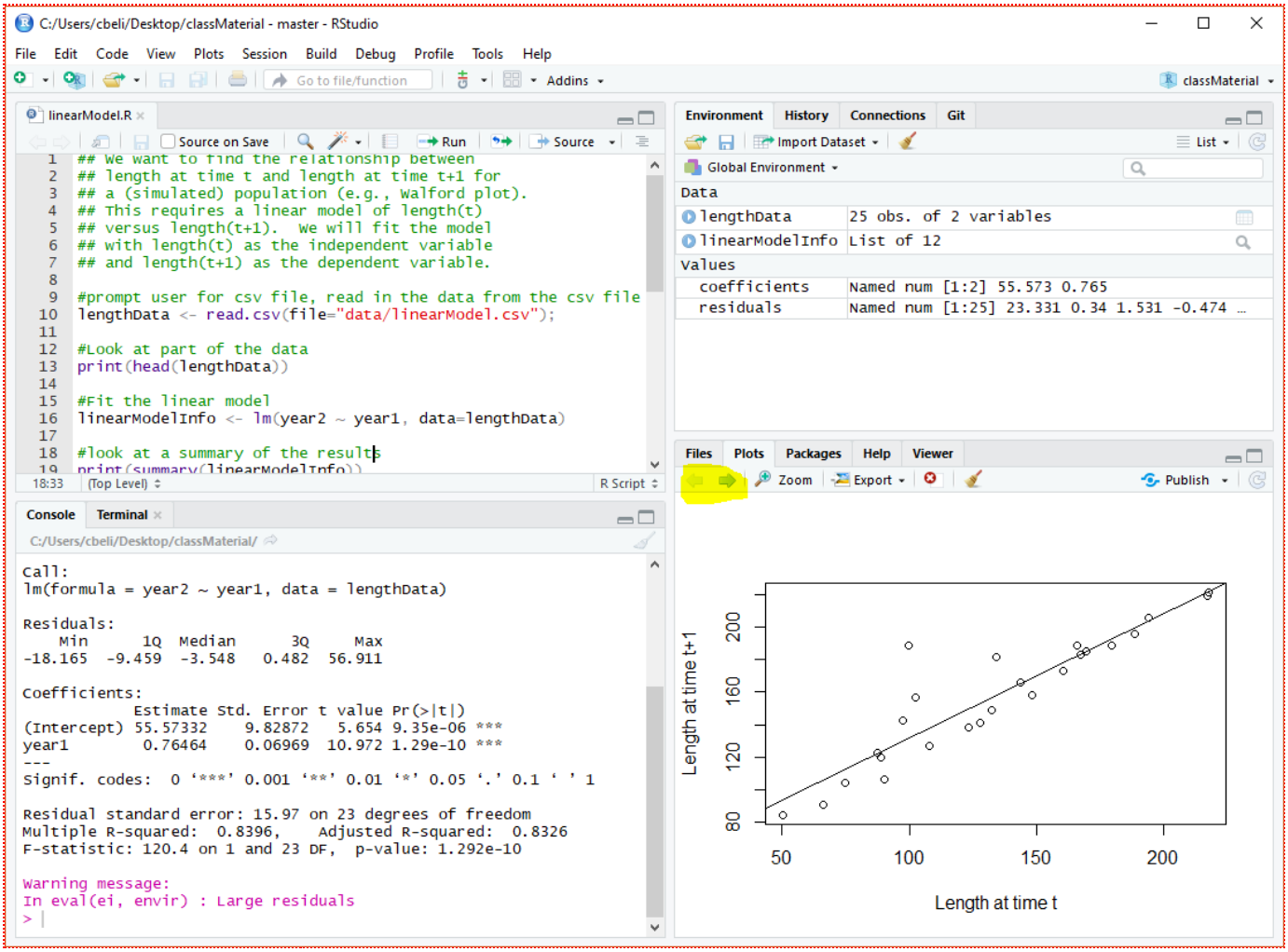

Fig 7: Using the arrow keys in the Plot Window

6.5 - Cleaning up the RStudio windows

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

- To clean the **Environment**, **Plot**, and **Component Windows** use the brush button (*Fig.8*).

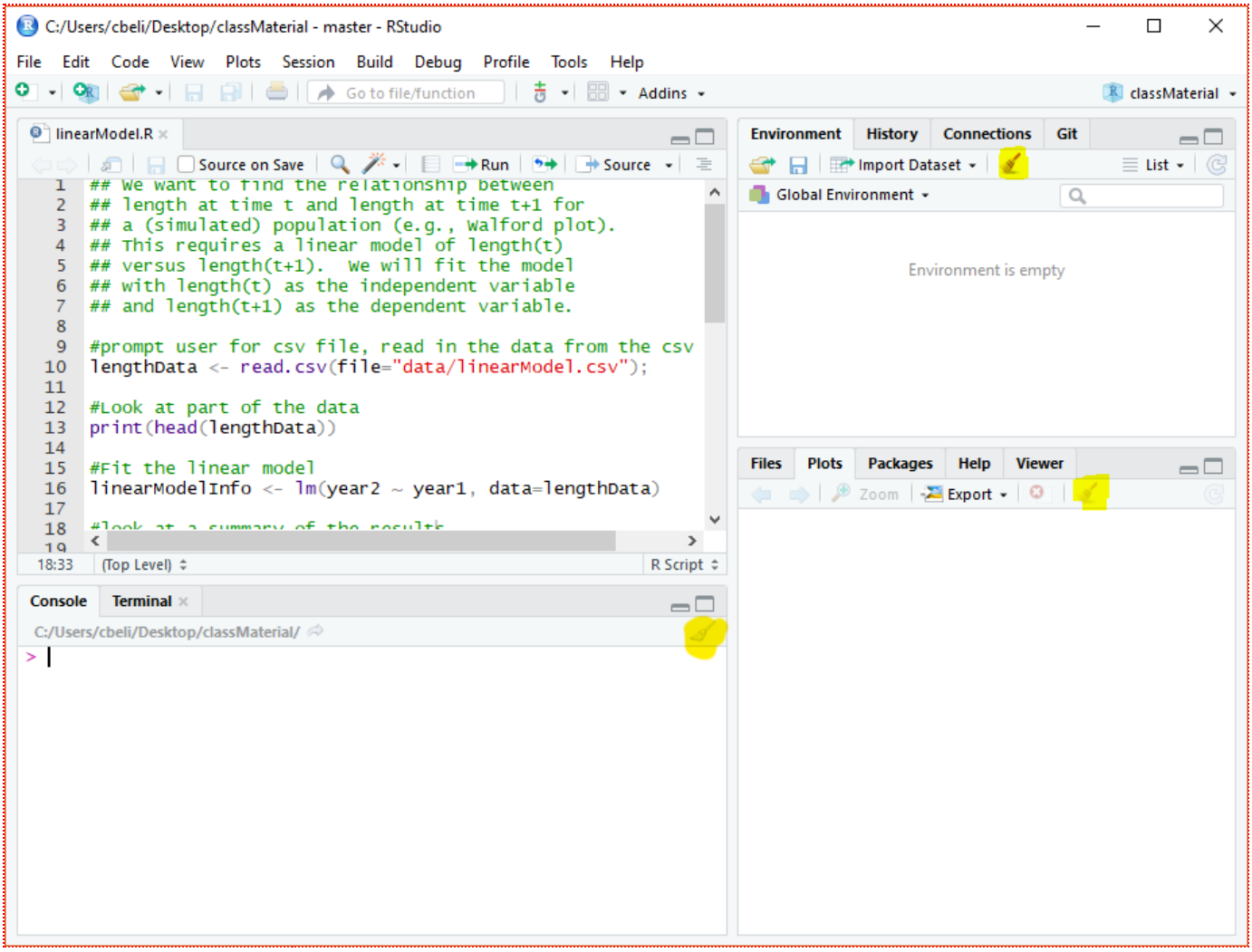

Fig 8: Cleaning out the windows

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

7 - Special note about images in the lessons for this class

RStudio is a rapidly evolving program and it would be too much work to maintain all the images so that they reflect the newest version of RStudio and its visual likeness. Most of the images in this class will reflect the 2017 version of RStudio (v1.0). Also, I have made changes the material in the class so there will be other minor changes. For instance, my RStudio Project name changed in the last few images and some of the RStudio buttons and tabs are different (how observant were you?). I have tried to make sure these differences change nothing functionally. Inevitably, something will fall through the cracks -- so, please, contact the instructor if there is something inconsistent causing you problems!

8 - 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.

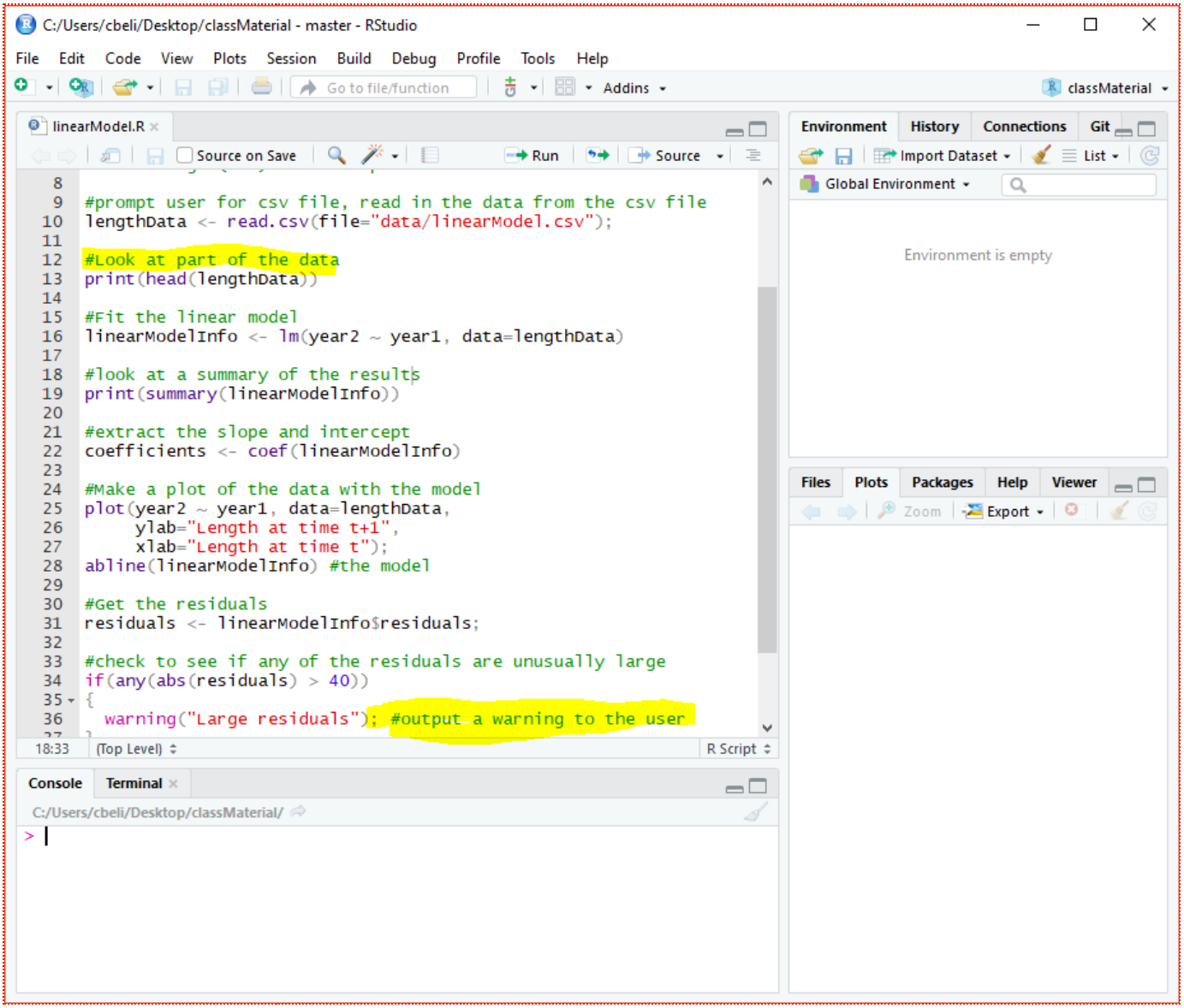

Fig 9: Comments in a script

Notice how the functionality of the script (*Fig.9*) can be generally understood just by reading the comments -- this is the main purpose of commenting.

8.1 - How to make comments in R

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! Commenting also significantly reduces the frustration you will feel when trying to read your own code after an extended absence.

9 - Color Schemes

If you just installed RStudio and are using the default color scheme then it probably looks like *Fig.10*. In this color scheme the comments below are in a lighter green whereas most of the other text is in black.

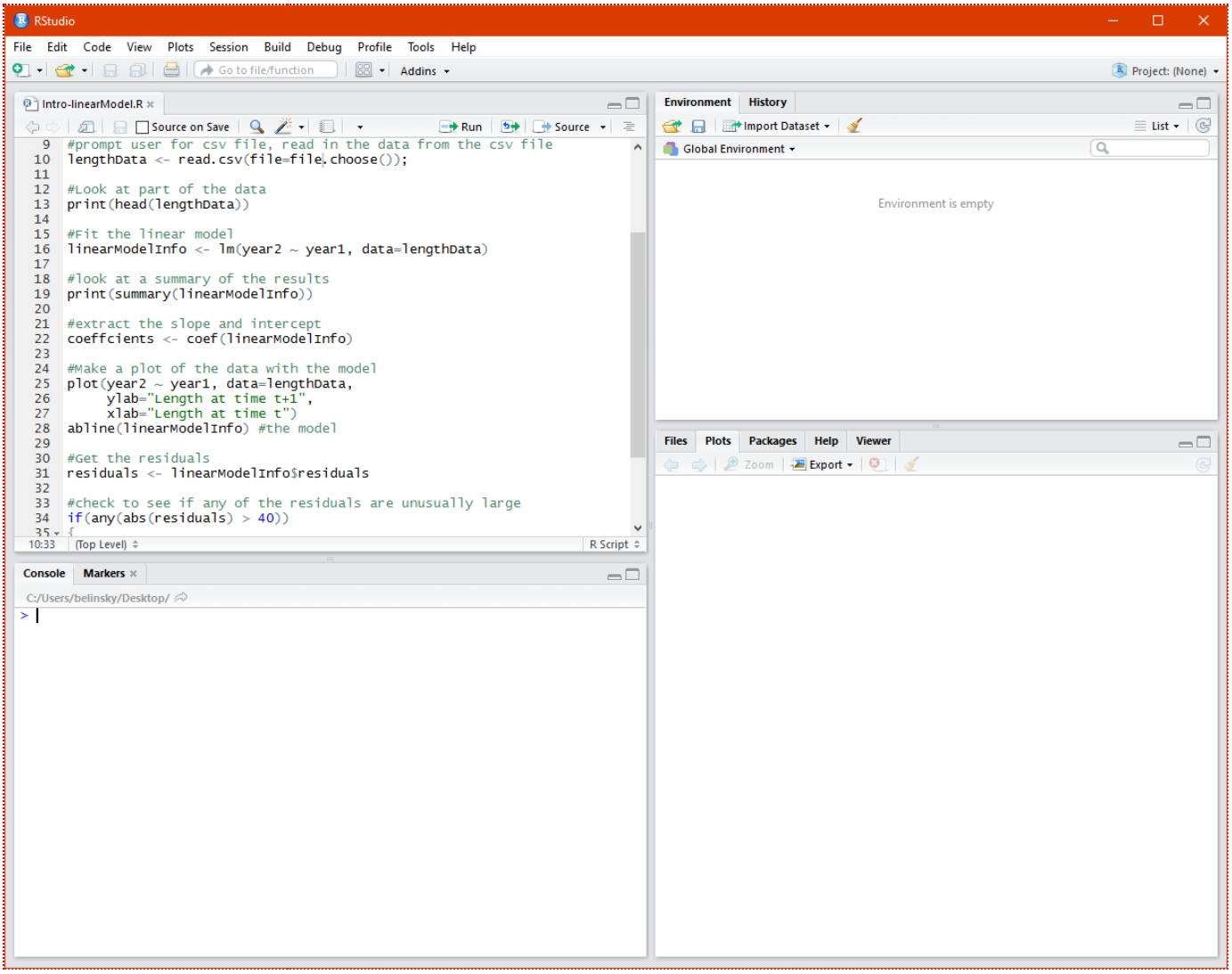

Fig 10: Comments are in the lighter green color

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.11*).

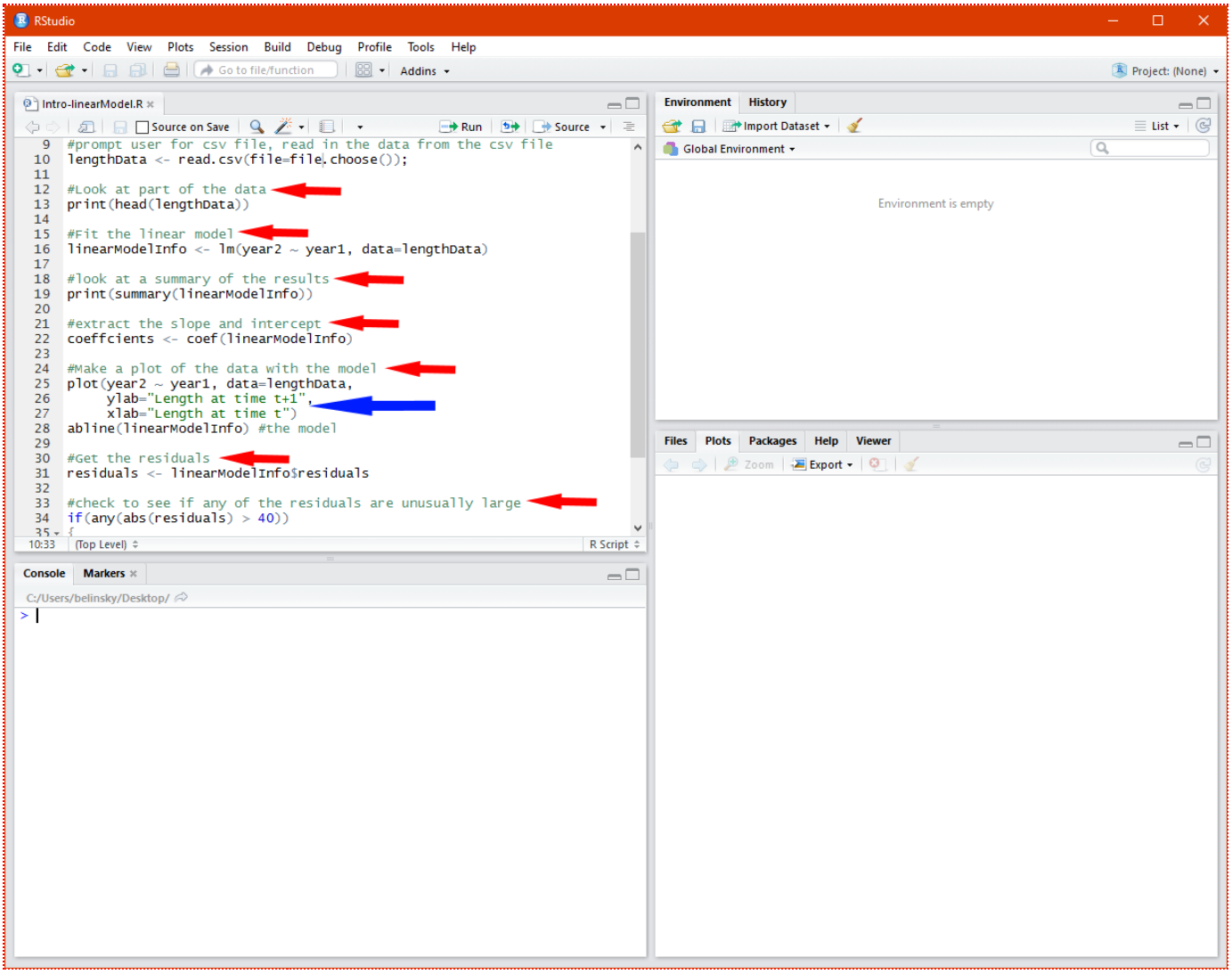

Fig 11: The text color for comments and quotes

9.1 - Changing the color scheme

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 12*)

1. click on ***Tools*** in the main menu (circled in *Fig.12*)
2. choose ***Global Options***
3. When the ***Global Options*** window open (in *Fig.12*), click on ***Appearance***

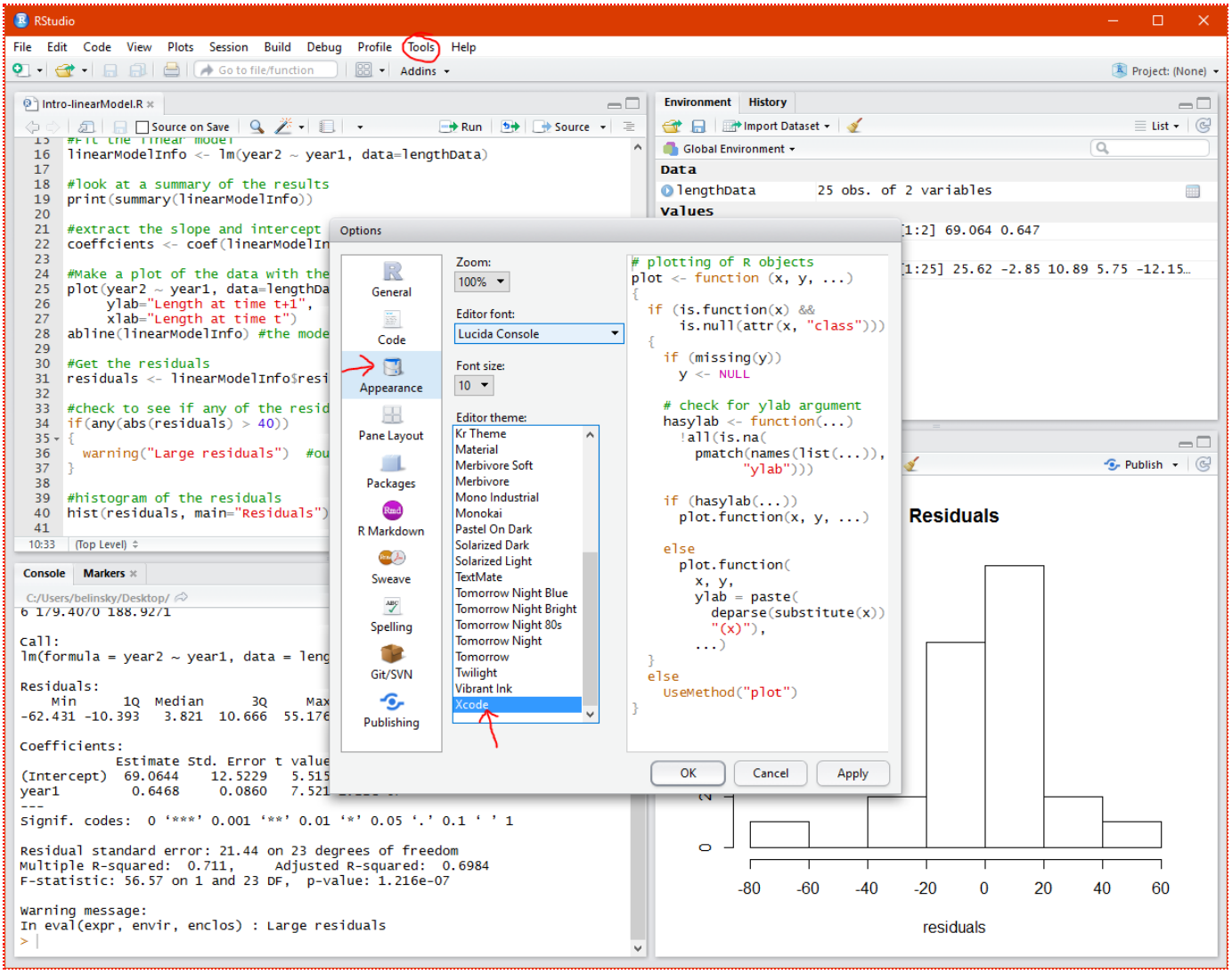

Fig 12: Color schemes for RStudio (note: the Editor font might be different in your window -- that is OK)

The picture above shows the *Xcode* color scheme (*Fig 12*). 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.

9.2 - Adding more color to differentiate output

There are a couple more helpful options you should set in RStudio that use color to distinguish other components of your code.

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

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

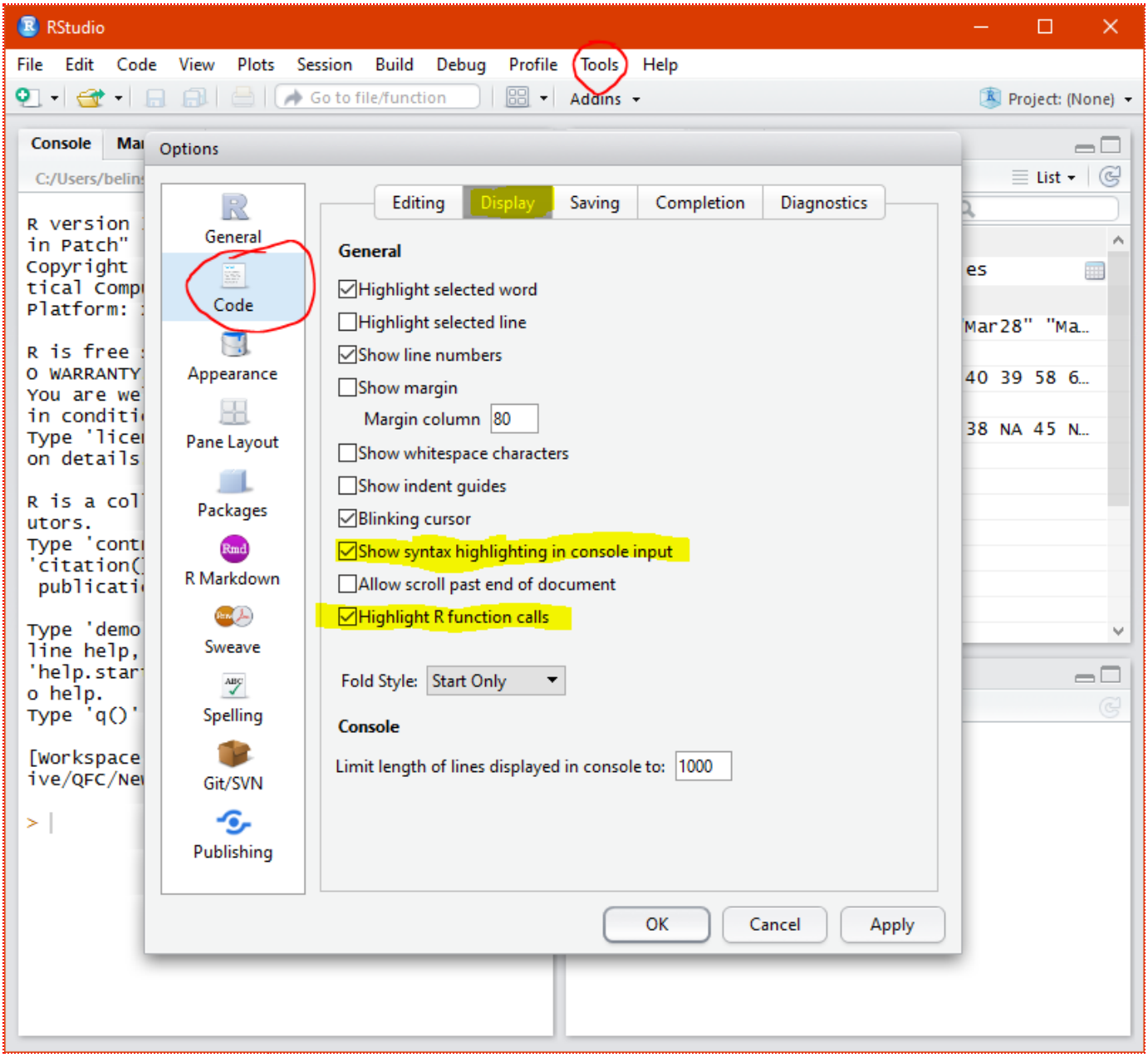

Fig 13: Console Window changes

10 - 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 matching parenthesis or quotes when the user types in a parenthesis or start quote.

You can turn off this feature by:

- going to ***Tools -> Global Options... -> Code -> Editing***
- uncheck *Insert matching parens/quotes*
- set **Surround selection on text insertion** to ***Never***

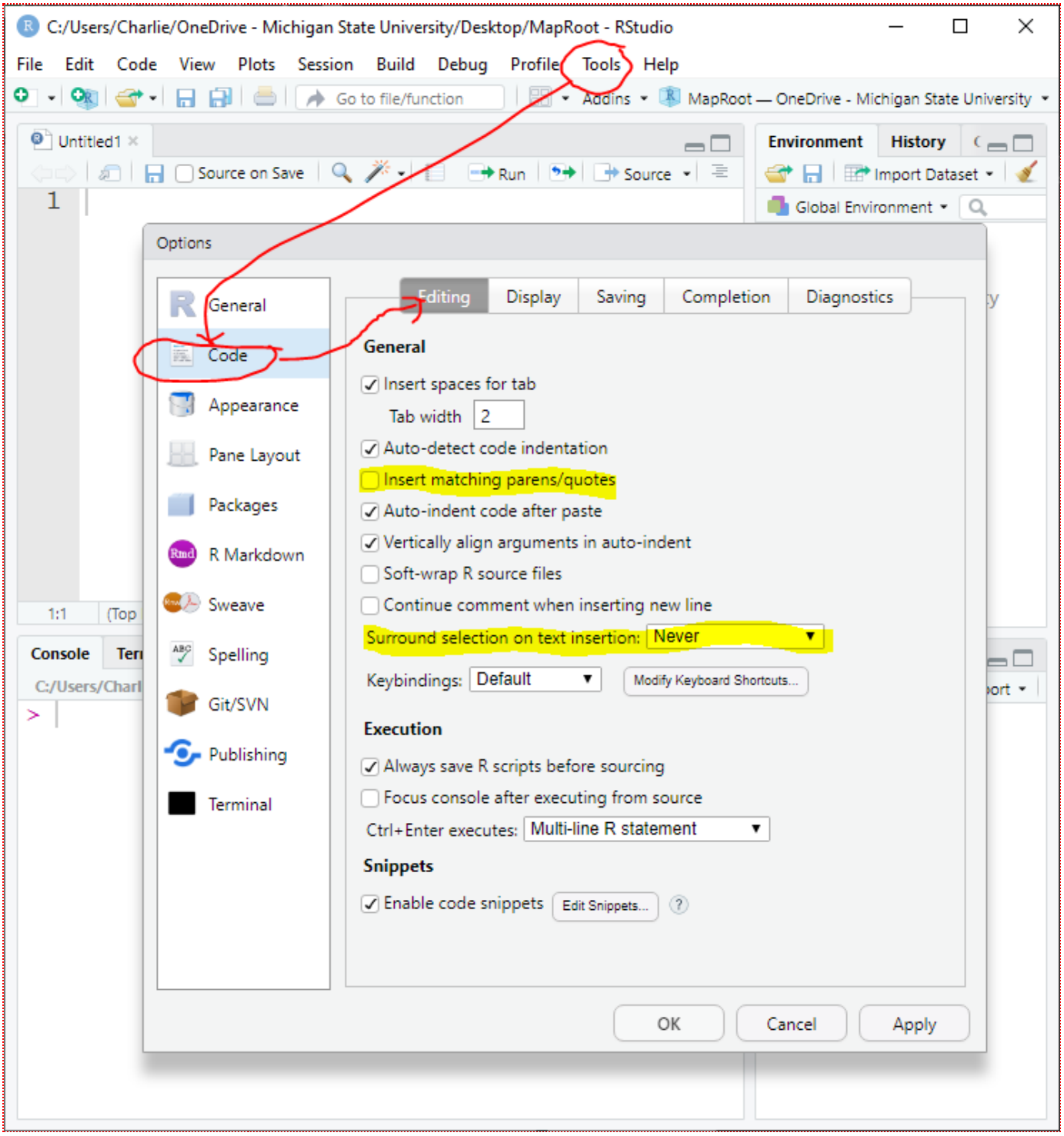

Fig 14: Stop RStudio from automatically ending your parenthesis and quotes.

11 - Application

1. In RStudio, create a new script file (***File*** -> ***New File*** -> ***R Script***)
2. Add some comment lines to the script
3. Save the script as ***app1-1.r*** to your ***script*** folder.

If you have any questions regarding this application, feel free to email them to the instructor here.

You can attach files to the email above. However, as the class progresses, the number of files increases. At that point, it might be easier to send the whole Root Folder as a zipped file. *Instructions for zipping the Root*

Folder are in the extension.

12 - Extension: Zipping your Root Folder

Windows: Zip your Root Folder (*Fig.15*)**:**

1. In your File Manager (not in RStudio), right-click on the Root Folder
2. Choose ***Send to***
3. Choose ***Compressed (zipped) folder***
4. A zipped file named ***<Root Folder>.zip*** with all the Root Folders' contents is created in the same folder

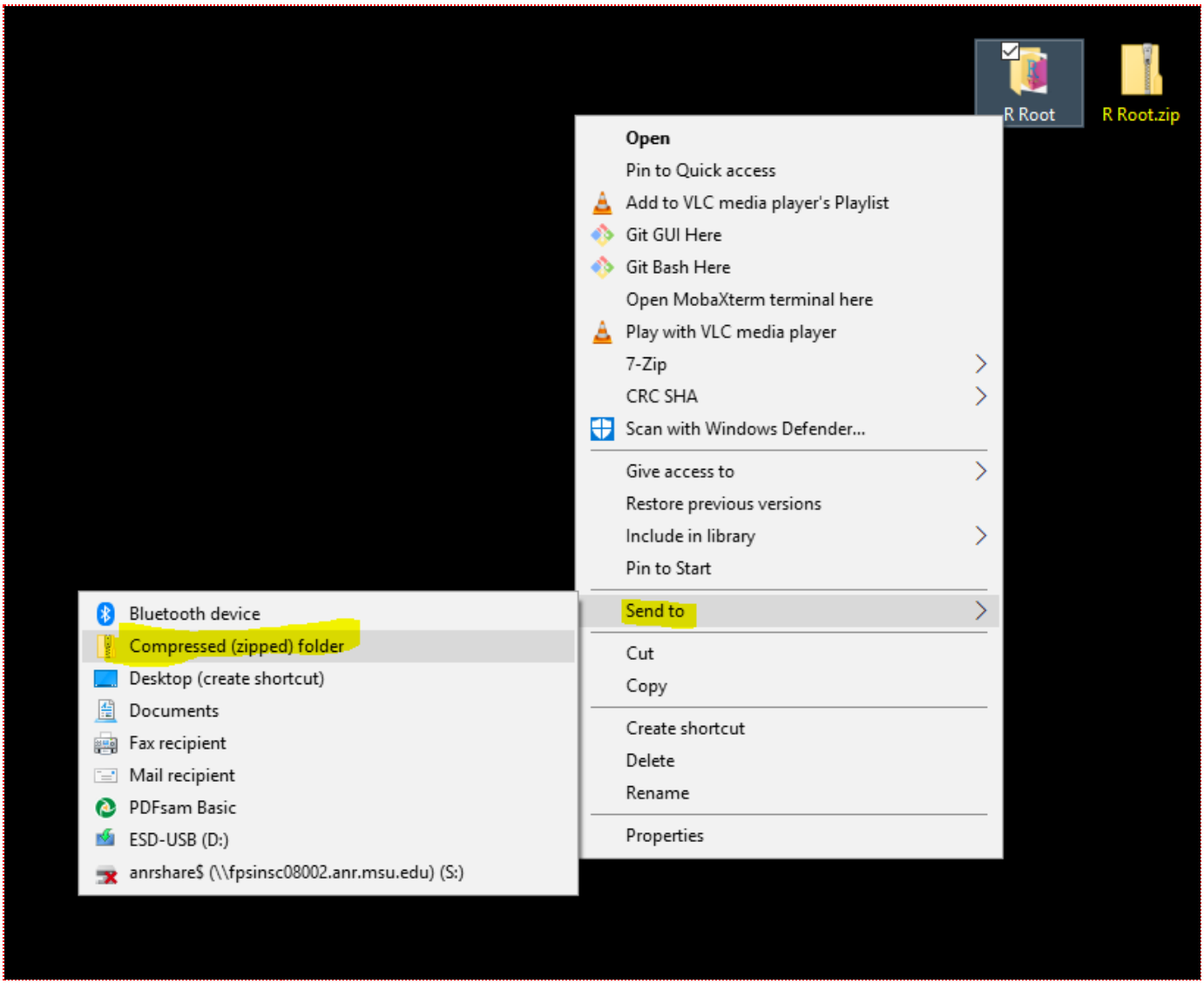

*Fig 15: Zipping your Class Folder on Windows -- in this case the zipped file created is named **R Root.zip***

Mac: Zip your Root Folder:

1. In your File Manager (not in RStudio), right-click on the Root Folder
2. Choose ***Compress "<Root Folder>"***
3. A zipped file named ***<Root Folder>.zip*** with all the Root Folders' contents is created in the same folder

13 - 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.**

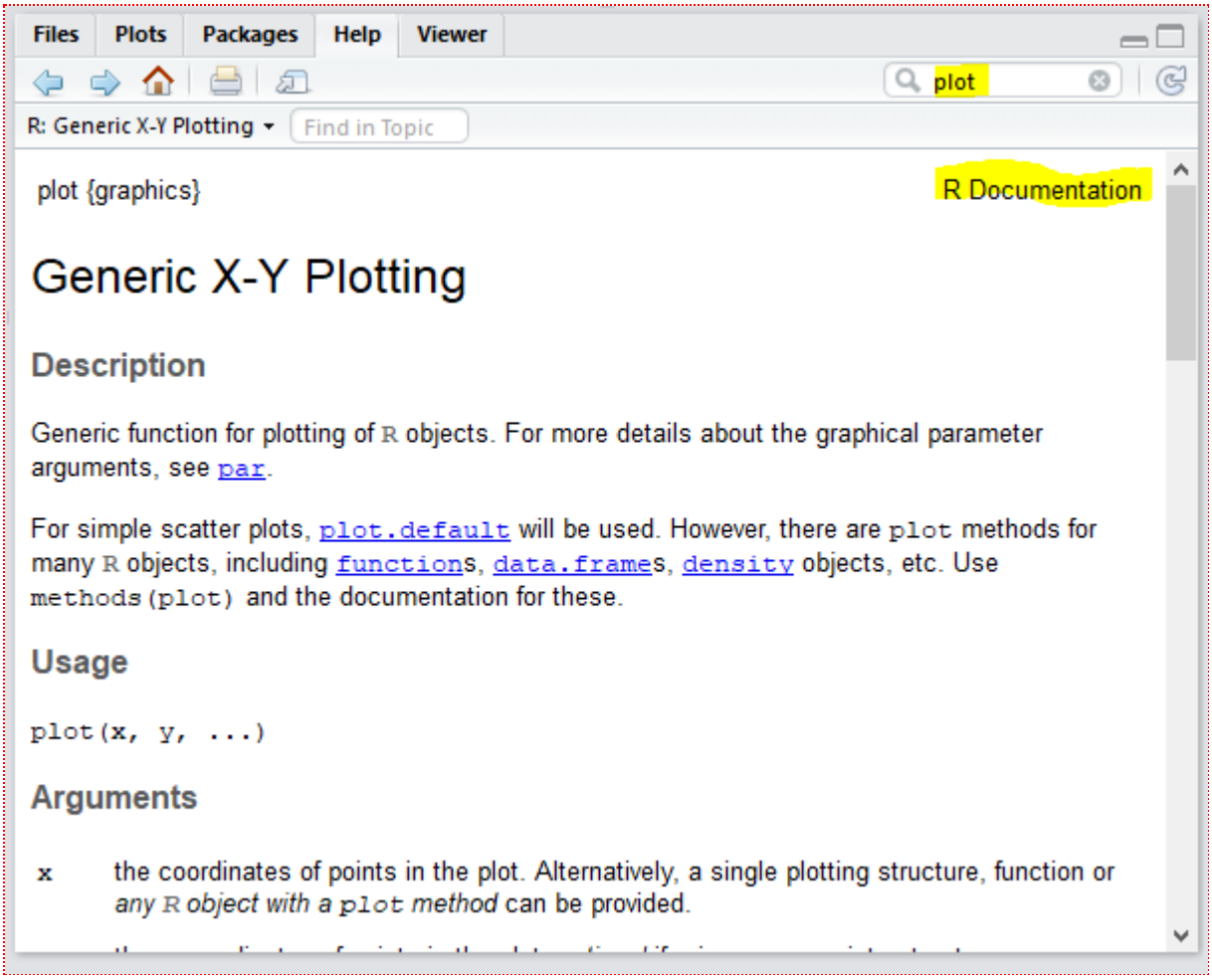

Fig 16: The R plot help page in the online documentation.

The page that appears in the **Help Window** (*Fig.16*) 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.

14 - 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.

15 - Extension: Installing RStudio on a Mac

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 17*).

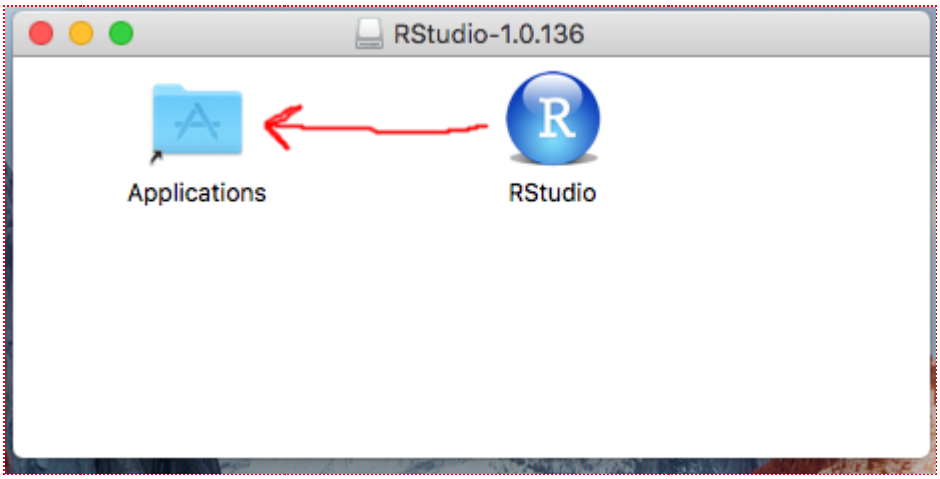

Fig 17: Opening RStudio DMG file

3. In the RStudio window above (*Fig 17*) drag the RStudio file to the Applications folder

Unmount the RStudio device by clicking the Eject button -- shown below (*Fig 18*)

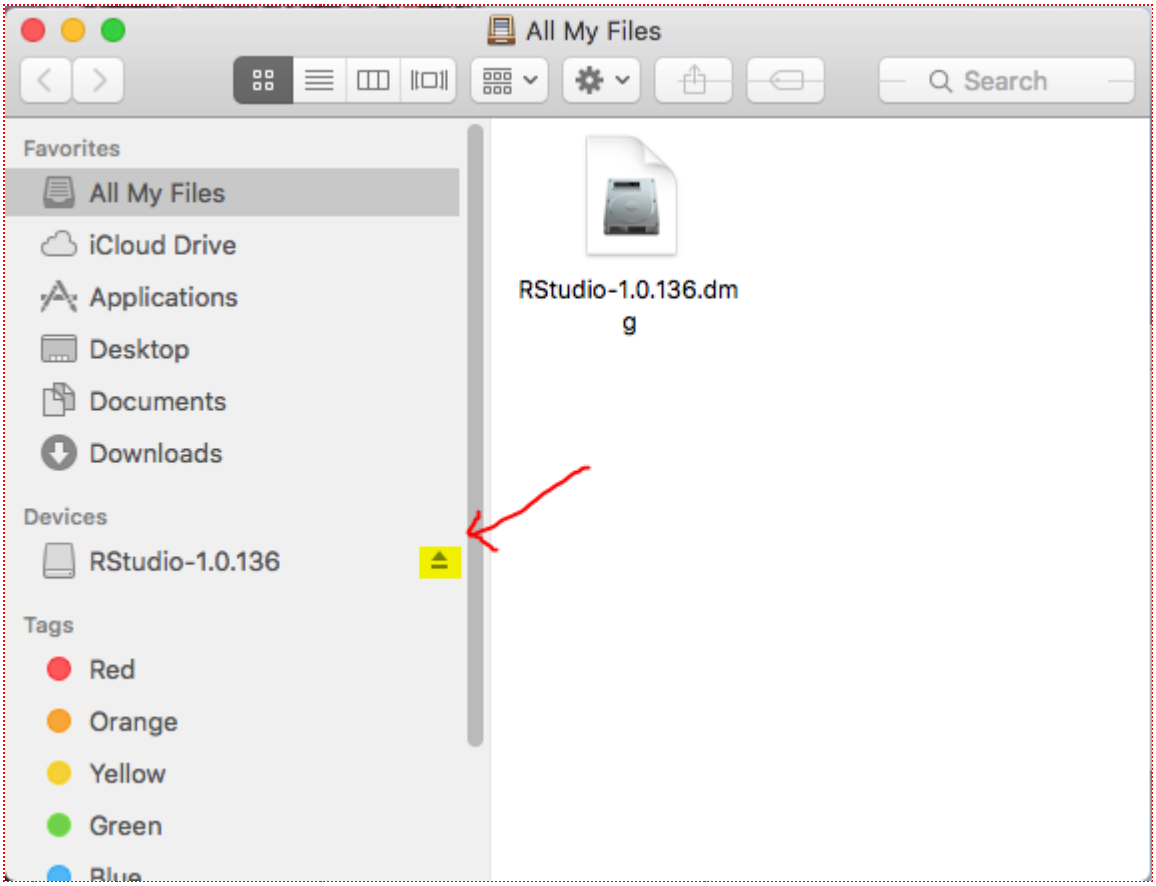

Fig 18: Unmounting the RStudio device (very important!)

16 - Extension: Create a standalone R script

There are two ways to create script file in RStudio:

1. A standalone script
2. As part of an RStudio Project

Many people use the first method. In this class we use the second method. The first method of creating a script file is good if you want to test something out. The second method is better for organizing larger projects or sharing your code with others.

Without getting into gory detail of the issues, problems with the first method occur when you reading from another file, like a data file -- something we will do a lot of starting in Unit 2. This is because the first method does not explicitly declare a root folder, where the method does -- it is the Project folder.

You will often hear R Programmers get around this issue by setting the working directory in code -- something we will talk about later.

17 - Extension: *.RProj and .Rhistory (files added by RStudio)

The ***.RProj*** file (where * is the name of your project) stores basic information about your project -- you do not need to edit this file. In your file manager, double-clicking ***.RProj*** opens the RStudio Project. Inside RStudio, clicking on ***.RProj*** brings up the Project Options window (*Fig.19*), which you can also get to by clicking on ***Tools -> Project Options.***

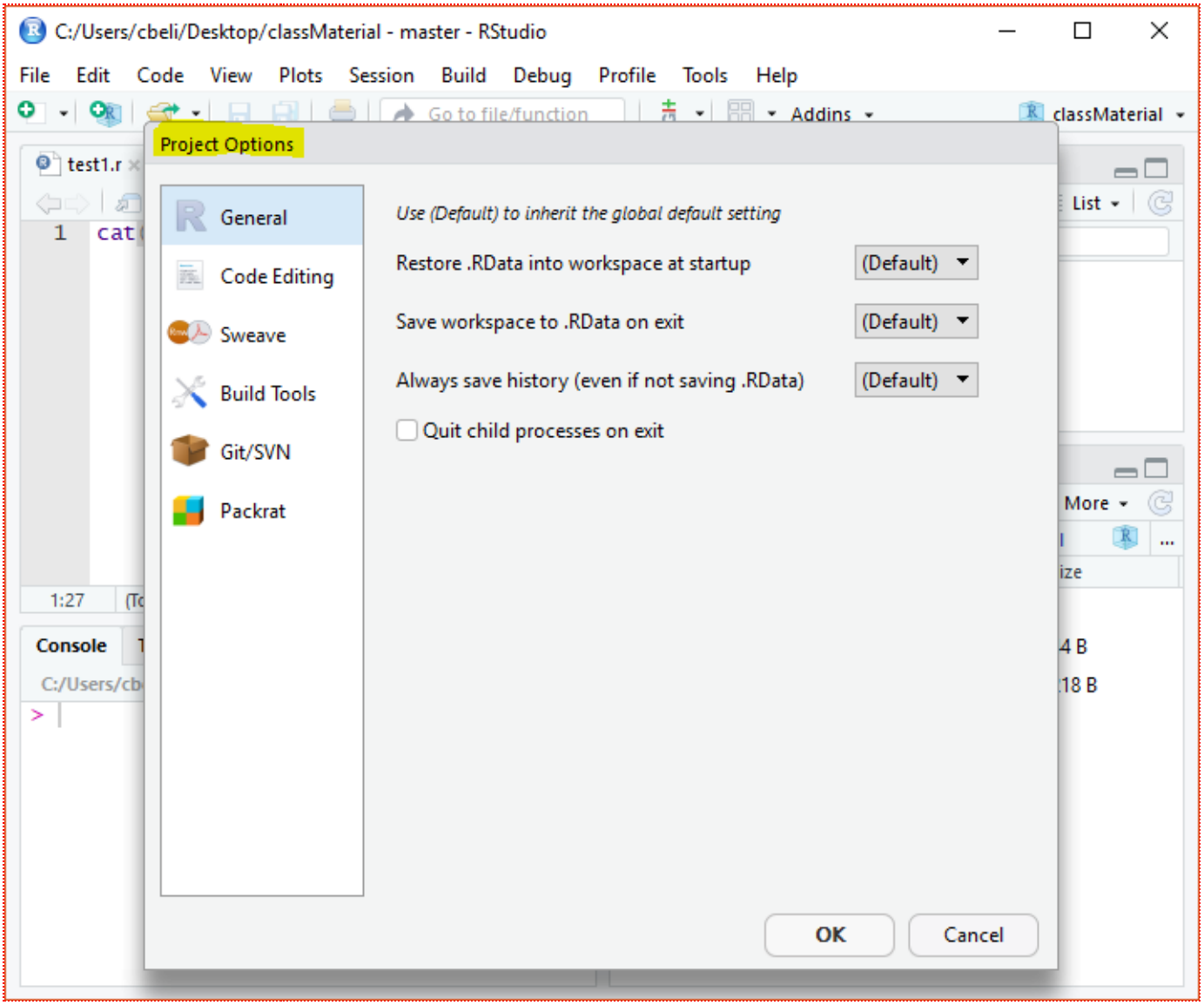

Fig 19: Project Options window

The ***.Rhistory*** file stores the information from the ***Console Window*** (bottom-left corner). For this class, you will not need to use the ***.Rhistory*** file.