01-01: GGPlot Installation and Setup

1 - Purpose

- Installation of software and packages needed for class
- Explain the components of RStudio
- Set up an RStudio Project -- this is where all your work for class will go

2 - Install software on your computer

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

   *You can also update RStudio by clicking **Help** -> **Check for Updates***

   Extension: Special instruction for Mac users

2.1 - Installing the GGPlot2 package

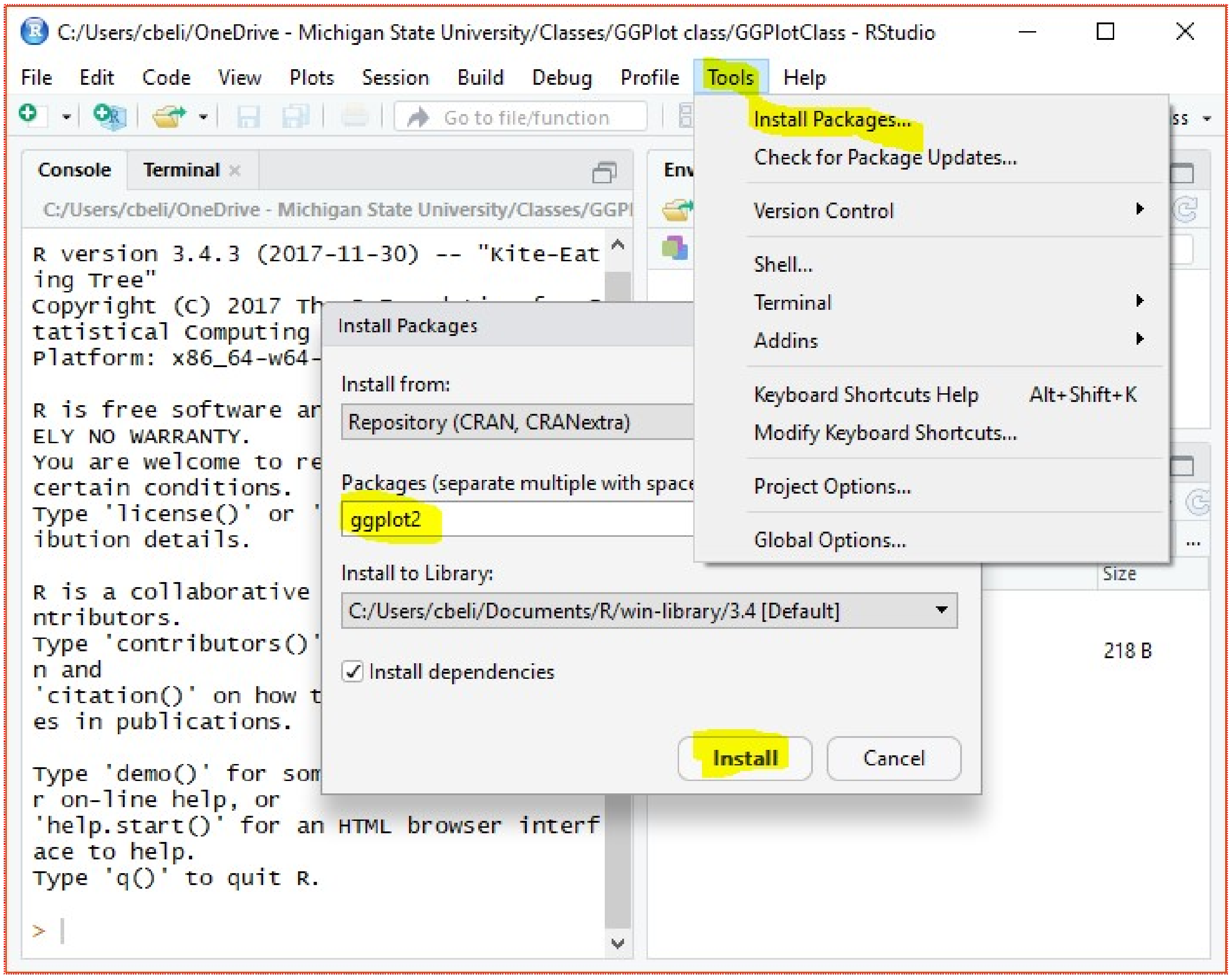
The package, ***GGPlot2***, does not come with R or RStudio -- so we are going to install the ***GGPlot2*** package in RStudio.

To install the package ***GGPlot2*** using RStudio (*fig 1*):

1. Click ***Tools*** -> ***Install Packages...***
2. In the ***Install Packages*** window, type **GGPlot2** in the ***Packages*** textbox
3. Click ***Install***

Extension: another way to install packages in R

GGPlot2 requires many other packages -- and RStudio will install those packages along with GGPlot2. Be patient, as this could take a little time.


C:/Users/cbeli/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotClass - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Console Terminal
Install Packages...
Check for Package Updates...
Version Control
Shell...
Terminal
Addins
Keyboard Shortcuts Help Alt+Shift+K
Modify Keyboard Shortcuts...
Project Options...
Global Options...
Install Packages
Install from:
Packages (separate multiple with space
Install to Library:
Install dependencies
Install
Cancel
Size
218 B

```
R version 3.4.3 (2017-11-30) -- "Kite-Eat
ing Tree"
Copyright (C) 2017 Th
tatistical Computing
Platform: x86_64-w64-

R is free software ar
ELY NO WARRANTY.
You are welcome to re
certain conditions.
Type 'license()' or '
ibution details.

R is a collaborative
ntributors.
Type 'contributors()'
n and
'citation()' on how t
es in publications.

Type 'demo()' for som
r on-line help, or
'help.start()' for an HTML browser interf
ace to help.
Type 'q()' to quit R.

>
```

Install from: Repository (CRAN, CRANextra)

Packages (separate multiple with space: ggplot2

Install to Library: C:/Users/cbeli/Documents/R/win-library/3.4 [Default]

[✓] Install dependencies

Fig 1: Installing the GGPlot2 Package in RStudio

3 - RStudio Projects

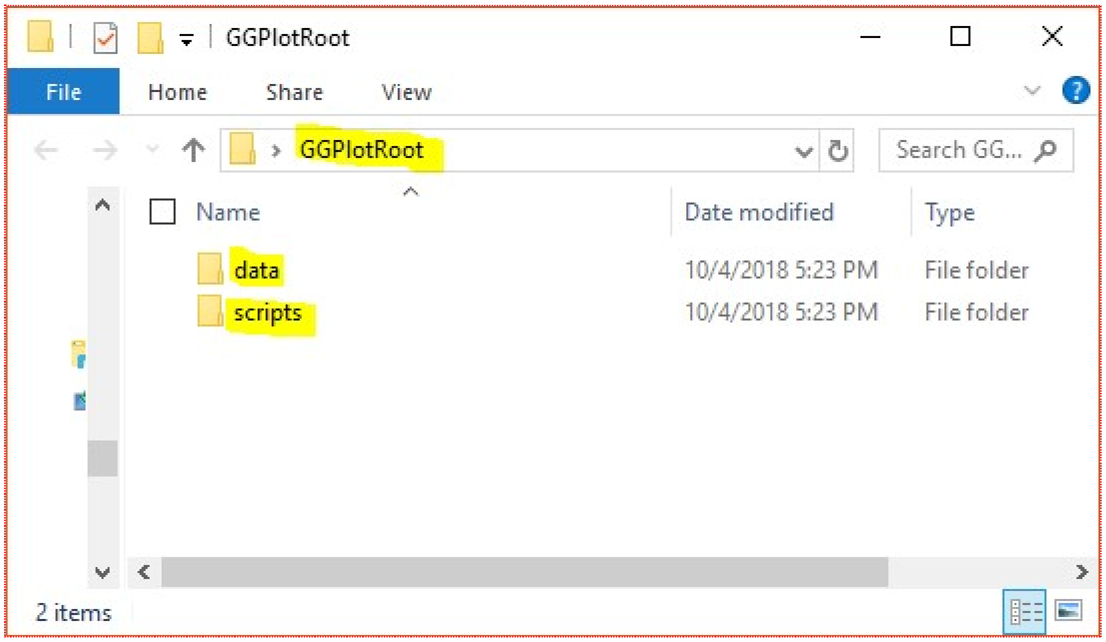
Most of the work you will do in R involves multiple scripts and data files. This means there needs to be a folder structure so that your R scripts know where to find the other files. ***setwd()*** can used to set the working, or root, directory for your project. But ***setwd()*** is awkward and needs to be changed as the files move between computers.

In RStudio, you can use RStudio Projects to create a folder structure that holds all your files -- a structure can be easily transferred to other computers.

The broad steps for creating an RStudio Project are:
1. Create a folder on your computer, which will be the *Root Folder* of your project, and add subfolders to it.
   *Note: **Root Folder** is a description of the folder -- you can name the folder whatever you want.*
2. Save files in the folders you just created.
3. In RStudio, link a new RStudio Project to the Root Folder.

3.1 - 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. This folder will be the **Root Folder** for all class material -- the name and location of the folder is up to you. In *fig 2*, I called the folder ***GGPlotRoot***. Inside the **Root Folder**, add two subfolders: ***data*** and ***scripts***.



Name	Date modified	Type
data	10/4/2018 5:23 PM	File folder
scripts	10/4/2018 5:23 PM	File folder

*Fig 2: My Class Folder, called **GGPlotRoot**, and its subfolders*

3.2 - Add files to the Root Folder

We are going to download and save three files to the **Root Folder** and the subfolders:
1. The Class Project Rubric.docx -- save this directly to the **Root Folder** -- not a subfolder
2. A data file in comma-separated-value format (lesson01.csv) -- save this to the **data** subfolder
3. An R script file (lesson01.r) -- save this to the **scripts** subfolder

So you should have:

- A **Root Folder**, which I named ***GGPlotRoot*** (you can choose another name)
- The file **ClassProjectRubric.docx** inside the **Root Folder**
- Two folders, ***data*** and ***scripts***, inside the **Root Folder**
- The file **lesson01.csv** inside the ***data*** folder
- The file **lesson01.r** inside the ***scripts*** folder

3.3 - Link RStudio to the Root Folder

We are going to create an **RStudio Project** that is linked to the **Root Folde**r 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 3*) click ***Browse***
4. In the ***Choose Directory*** window navigate to your **Root Folder** and click ***Open*** (in this example, ***GGPlotRoot***)
5. In the ***Create Project from Existing Directory*** window click ***Create Project***

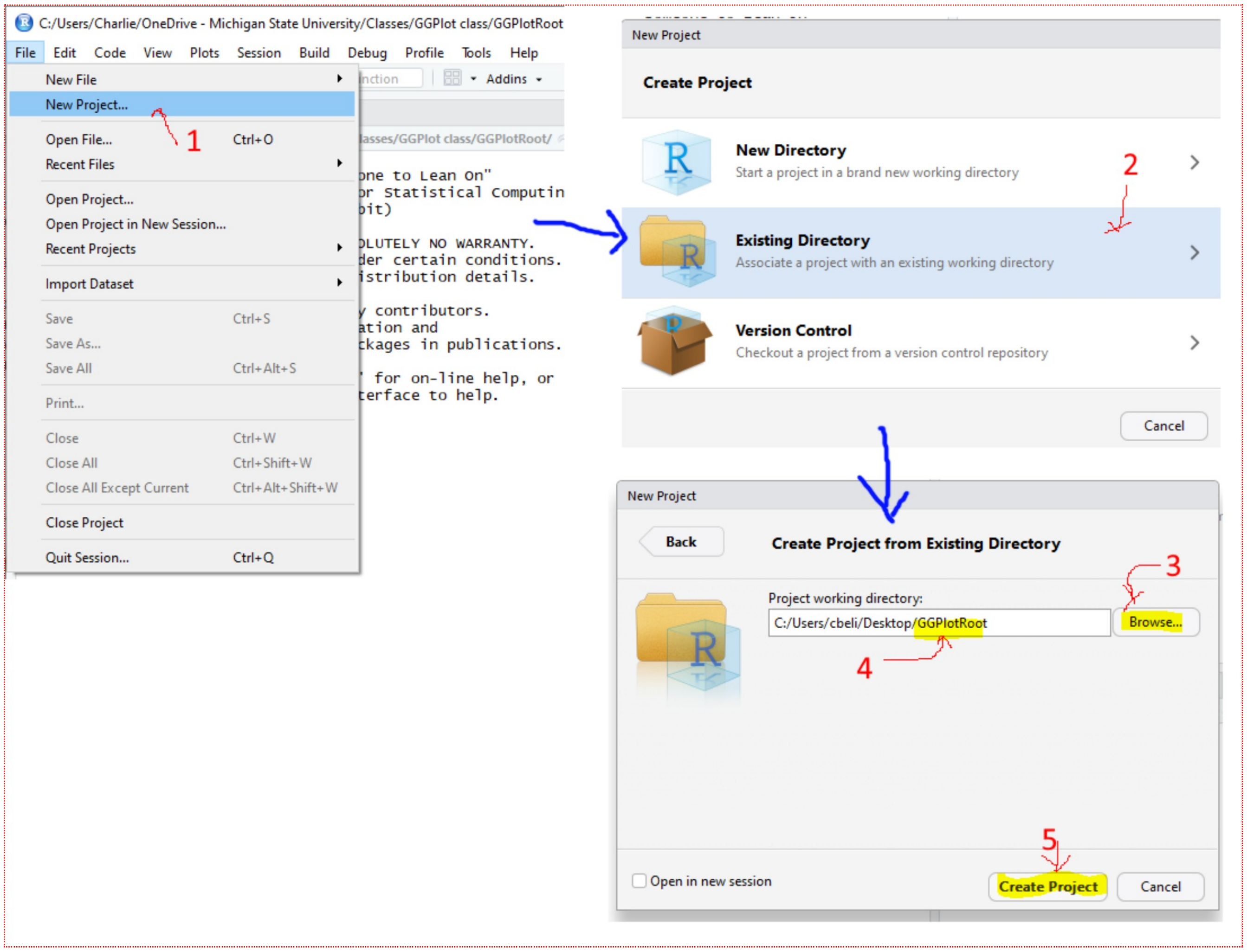
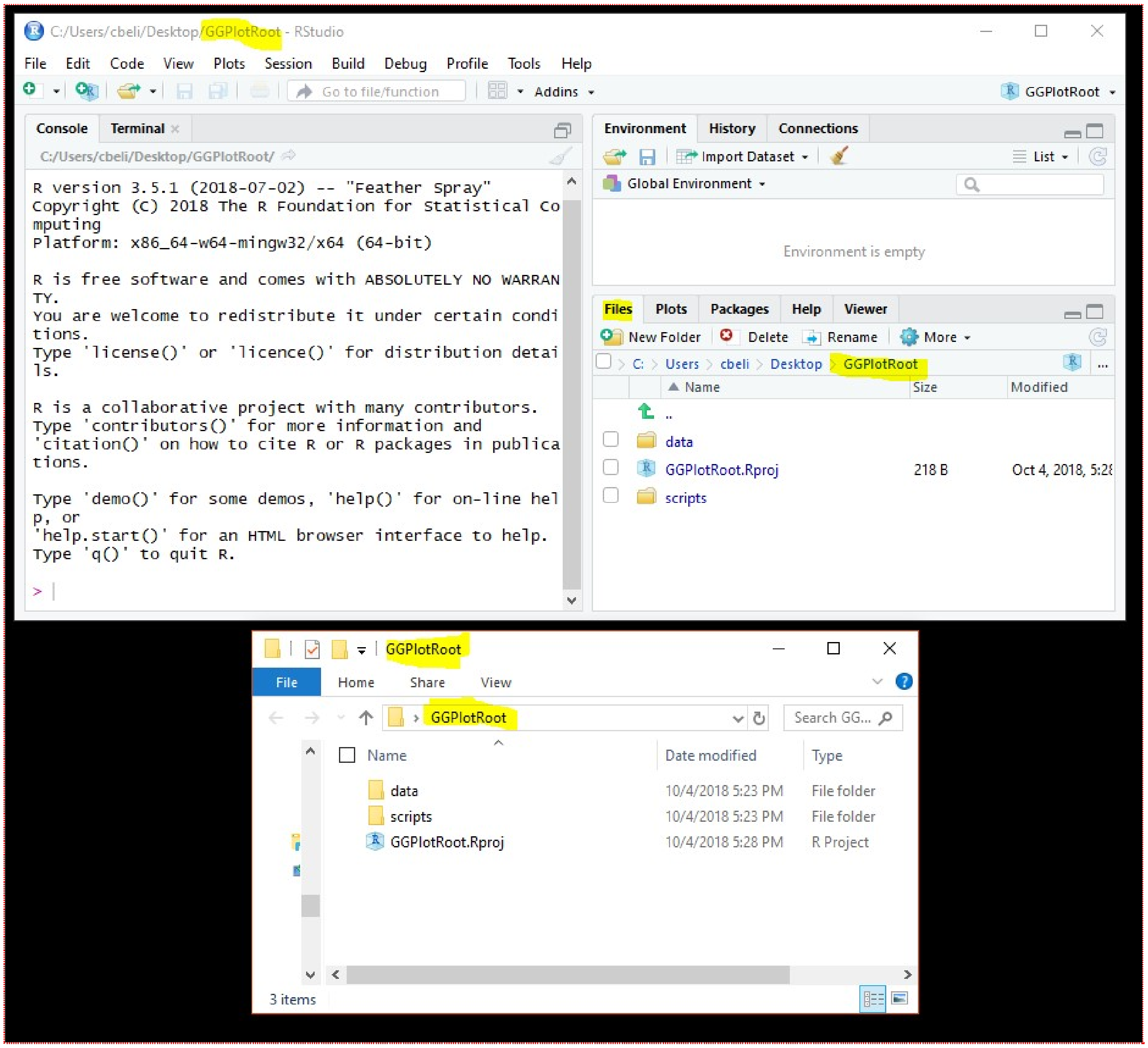

C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot
File Edit Code View Plots Session Build Debug Profile Tools Help
New File
New Project...
Open File... Ctrl+O
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save Ctrl+S
Save As...
Save All Ctrl+Alt+S
Print...
Close Ctrl+W
Close All Ctrl+Shift+W
Close All Except Current Ctrl+Alt+Shift+W
Close Project
Quit Session... Ctrl+Q
1
New Project
Create Project
New Directory
Start a project in a brand new working directory
2
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel
New Project
Back
Create Project from Existing Directory
3
Project working directory:
C:/Users/cbeli/Desktop/GGPlotRoot
Browse...
4
5
Open in new session
Create Project
Cancel

Fig 3: The Create Project from Existing Directory window

4 - RStudio Project File Manager

You should now be running RStudio, opened to the 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 (linked to your Root Folder). *The RStudio **Files** tab is essentially a File Explorer in Windows or Finder in*

Mac (*fig 4*) -- the ***Files*** tab can be used to open, add, remove, or rename files and folders.


C:/Users/cbeli/Desktop/GGPlotRoot - RStudio

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Co
mputing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRAN
TY.
You are welcome to redistribute it under certain condi
tions.
Type 'license()' or 'licence()' for distribution detai
ls.

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

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

> |
```

Environment is empty

C: > Users > cbeli > Desktop > GGPlotRoot

Name	Size	Modified
..		
data		
GGPlotRoot.Rproj	218 B	Oct 4, 2018, 5:28
scripts		

GGPlotRoot

Name	Date modified	Type
data	10/4/2018 5:23 PM	File folder
scripts	10/4/2018 5:23 PM	File folder
GGPlotRoot.Rproj	10/4/2018 5:28 PM	R Project

3 items

Fig 4: The Files tab and the File Explorer in Windows-- both show the same thing.

A file was added by RStudio to your Root Folder called ***<Root Folder>.Rproj***. If your File Explorer Window is setup to see hidden files, then you will also see a 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*

5 - RStudio Basics

In this section, we are going to execute the script file, ***lesson01.R***, that we just downloaded to your RStudio Project's ***script*** folder.

lesson01.R takes weather data from the other downloaded file, ***lesson01.csv*** and creates boxplots showing how Wind Direction and Wind Speed relate to the Change In Temperature. The code in ***lesson01.R*** 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.

5.1 - Open your RStudio Project

There are multiple ways to open an RStudio Project -- three of them are:

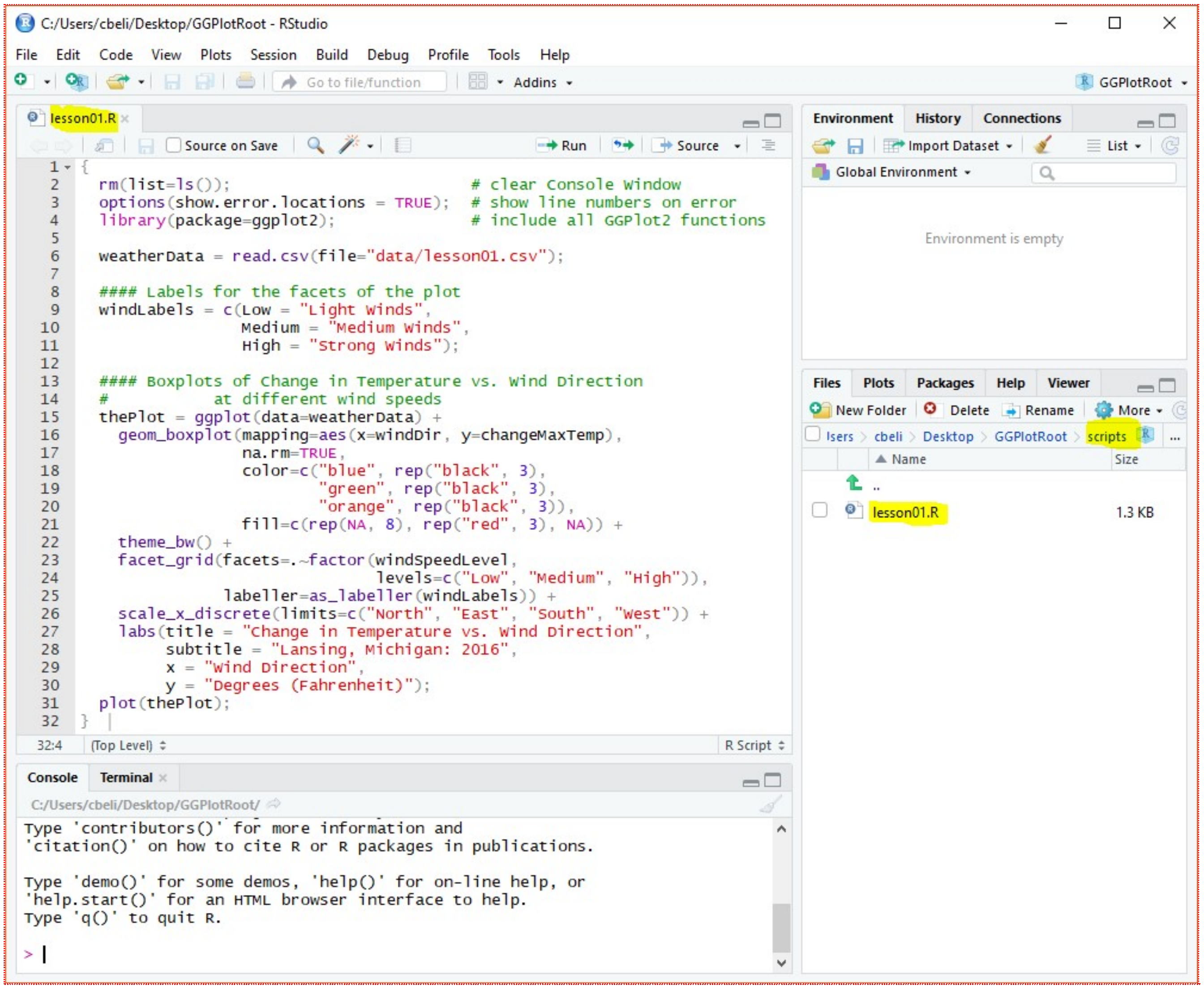
- Open your Root Folder in File Explorer (Windows) or Finder (MAC) window and double-click the ***.proj** file (where * is the name of your Project)
- In RStudio, click ***File*** -> ***Open Project...*** -> navigate to the Root Folder and click the ***.proj** file
- In RStudio, click ***File*** -> ***Recent Projects*** -> choose the RStudio Project you just created

5.2 - 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 ***lesson01.R*** or
- In RStudio click ***File -> Open File ->*** and then find ***lesson01.R*** in the ***scripts*** folder and click ***Open***.

After opening the script file, you should see something that looks like this (*fig 5*) on your screen:


C:/Users/cbeli/Desktop/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
GGPlotRoot
lesson01.R
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Isers cbeli Desktop GGPlotRoot scripts
Name Size
lesson01.R 1.3 KB
32:4 (Top Level) R Script
Console Terminal
C:/Users/cbeli/Desktop/GGPlotRoot/

```
{
  rm(list=ls());                            # clear Console Window
  options(show.error.locations = TRUE);     # show line numbers on error
  library(package=ggplot2);                 # include all GGPlot2 functions

  weatherData = read.csv(file="data/lesson01.csv");

  #### Labels for the facets of the plot
  windLabels = c(Low = "Light Winds",
                 Medium = "Medium Winds",
                 High = "Strong Winds");

  #### Boxplots of Change in Temperature vs. Wind Direction
  #           at different wind speeds
  thePlot = ggplot(data=weatherData) +
    geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                 na.rm=TRUE,
                 color=c("blue", rep("black", 3),
                         "green", rep("black", 3),
                         "orange", rep("black", 3)),
                 fill=c(rep(NA, 8), rep("red", 3), NA)) +
    theme_bw() +
    facet_grid(facets=.~factor(windSpeedLevel,
                               levels=c("Low", "Medium", "High")),
               labeller=as_labeller(windLabels)) +
    scale_x_discrete(limits=c("North", "East", "South", "West")) +
    labs(title = "Change in Temperature vs. Wind Direction",
         subtitle = "Lansing, Michigan: 2016",
         x = "Wind Direction",
         y = "Degrees (Fahrenheit)");
  plot(thePlot);
}
```

```
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.

> 
```

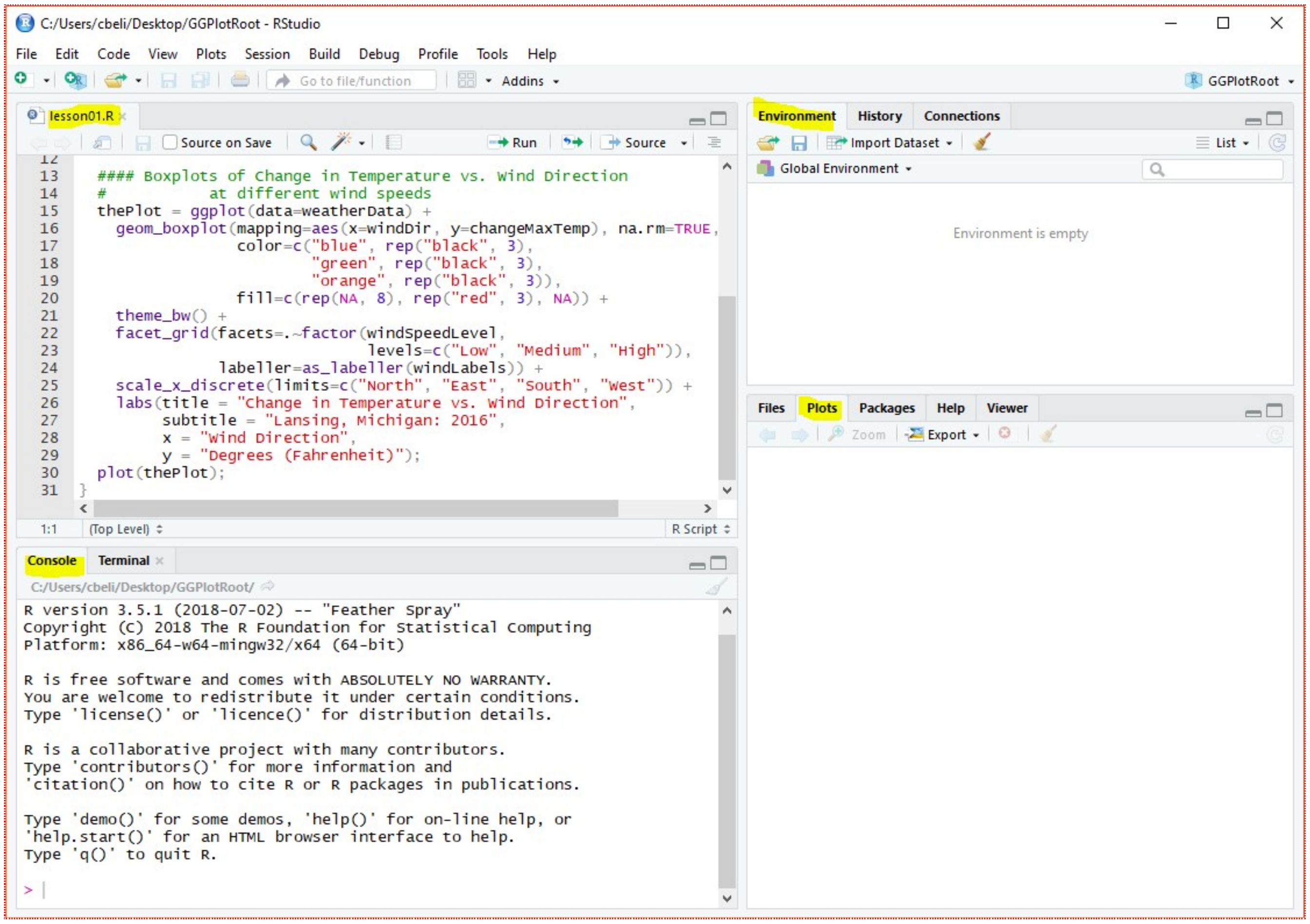
Fig 5: Opening your first script in RStudio

5.3 - Setting up RStudio tabs for scripts

When we are editing and executing an R script, we generally have the following RStudio tabs open (*fig 6*):

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 switched the tab to ***Plots***:



```
  #### Boxplots of Change in Temperature vs. Wind Direction
  #                at different wind speeds
  thePlot = ggplot(data=weatherData) +
    geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp), na.rm=TRUE,
                 color=c("blue", rep("black", 3),
                         "green", rep("black", 3),
                         "orange", rep("black", 3)),
                 fill=c(rep(NA, 8), rep("red", 3), NA)) +
    theme_bw() +
    facet_grid(facets=.~factor(windSpeedLevel,
                               levels=c("Low", "Medium", "High")),
               labeller=as_labeller(windLabels)) +
    scale_x_discrete(limits=c("North", "East", "South", "West")) +
    labs(title = "Change in Temperature vs. Wind Direction",
         subtitle = "Lansing, Michigan: 2016",
         x = "Wind Direction",
         y = "Degrees (Fahrenheit)");
  plot(thePlot);
}
```

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

> 
```

Fig 6: The Tabs most commonly used in RStudio

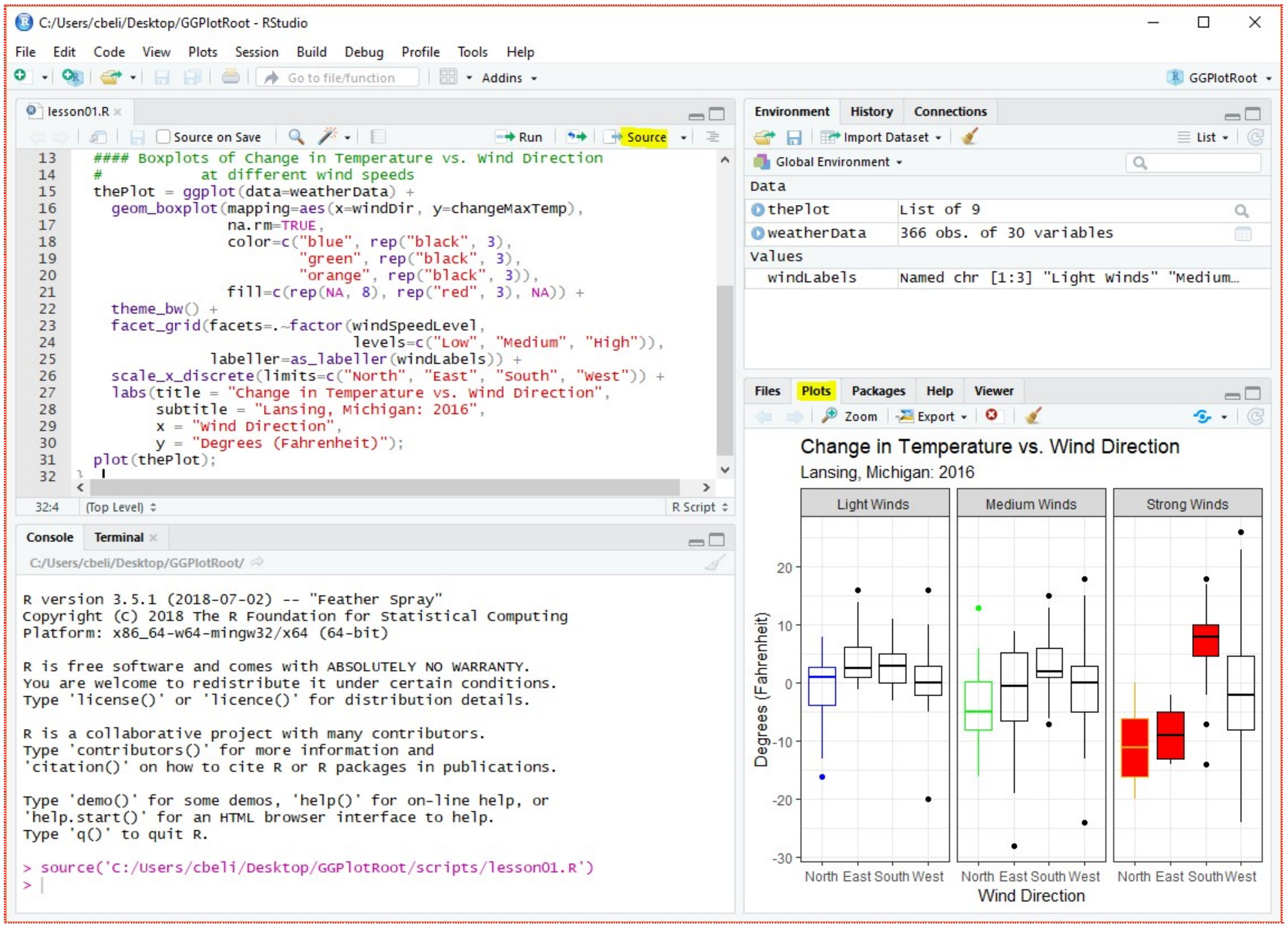
*Note: the **Help** tab in the lower-right corner is something you might find useful. Extension: The Help Tab*

5.4 - Common buttons used in RStudio

The script file, ***lesson01.r***, is a fully functioning script that takes temperature and wind data from the Comma Separated Value (CSV) file, ***lesson01.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 useful buttons in RStudio.

The one button you will use most in R is ***Source***, which executes your whole script. Press the ***Source*** button to execute the script (*fig 7*).

*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. For this class, you should always use Source to execute you code.*   For more information go to *Extension: Run vs. Source.*


C:/Users/cbeli/Desktop/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
GGPlotRoot
lesson01.R
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
Zoom
Export
Console Terminal
C:/Users/cbeli/Desktop/GGPlotRoot/
32:4 (Top Level) R Script
Change in Temperature vs. Wind Direction
Lansing, Michigan: 2016
Light Winds
Medium Winds
Strong Winds
Degrees (Fahrenheit)
20
10
0
-10
-20
-30
North East South West
Wind Direction

```
#### Boxplots of Change in Temperature vs. Wind Direction
#          at different wind speeds
thePlot = ggplot(data=weatherData) +
  geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
               na.rm=TRUE,
               color=c("blue", rep("black", 3),
                       "green", rep("black", 3),
                       "orange", rep("black", 3)),
               fill=c(rep(NA, 8), rep("red", 3), NA)) +
  theme_bw() +
  facet_grid(facets=.~factor(windSpeedLevel,
                             levels=c("Low", "Medium", "High")),
           labeller=as_labeller(windLabels)) +
  scale_x_discrete(limits=c("North", "East", "South", "West")) +
  labs(title = "Change in Temperature vs. Wind Direction",
       subtitle = "Lansing, Michigan: 2016",
       x = "Wind Direction",
       y = "Degrees (Fahrenheit)");
plot(thePlot);
}
```

```
Data
thePlot       List of 9
weatherData   366 obs. of 30 variables
Values
windLabels    Named chr [1:3] "Light Winds" "Medium…
```

```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

> source('C:/Users/cbeli/Desktop/GGPlotRoot/scripts/lesson01.R')
> 
```

Fig 7: Running (Sourcing) the script

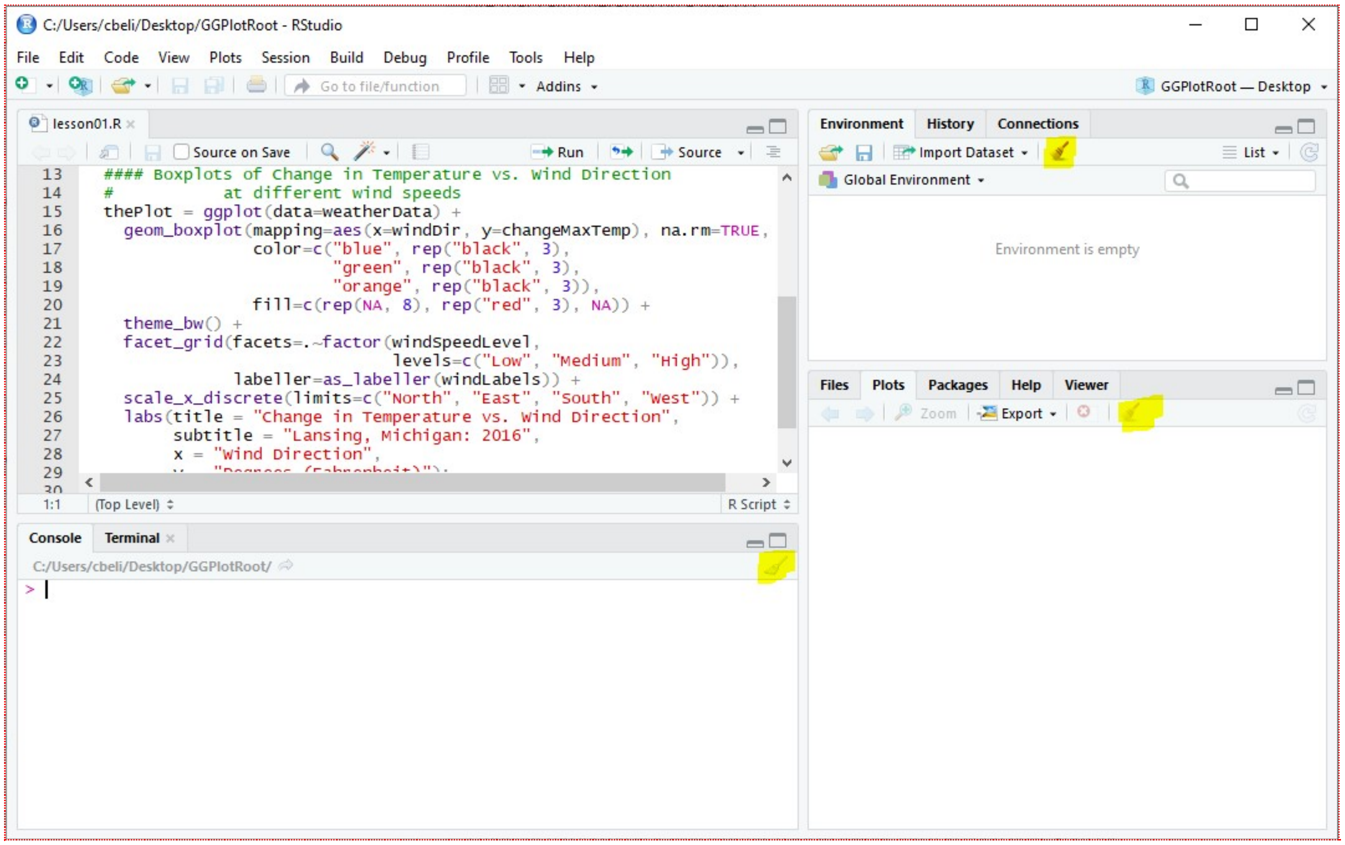
After the script is run:

- The **Environment Window** displays values for the data (variables) in the script (e.g., ***weatherData***, ***windLabels***).
- The **Console Window** displays information about the execution of the script.
- The **Plot Window** displays the box plots. If there are multiple plots, you can use the arrow buttons to switch between the plots.

5.5 - Cleaning up the RStudio windows

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

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


C:/Users/cbeli/Desktop/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
GGPlotRoot — Desktop
lesson01.R
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Zoom
Export
1:1
(Top Level)
R Script
Console Terminal
C:/Users/cbeli/Desktop/GGPlotRoot/

```
#### Boxplots of Change in Temperature vs. Wind Direction
#           at different wind speeds
thePlot = ggplot(data=weatherData) +
  geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp), na.rm=TRUE,
               color=c("blue", rep("black", 3),
                       "green", rep("black", 3),
                       "orange", rep("black", 3)),
               fill=c(rep(NA, 8), rep("red", 3), NA)) +
  theme_bw() +
  facet_grid(facets=.~factor(windSpeedLevel,
                             levels=c("Low", "Medium", "High")),
             labeller=as_labeller(windLabels)) +
  scale_x_discrete(limits=c("North", "East", "South", "West")) +
  labs(title = "Change in Temperature vs. Wind Direction",
       subtitle = "Lansing, Michigan: 2016",
       x = "Wind Direction",
```

```
>
```

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.

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

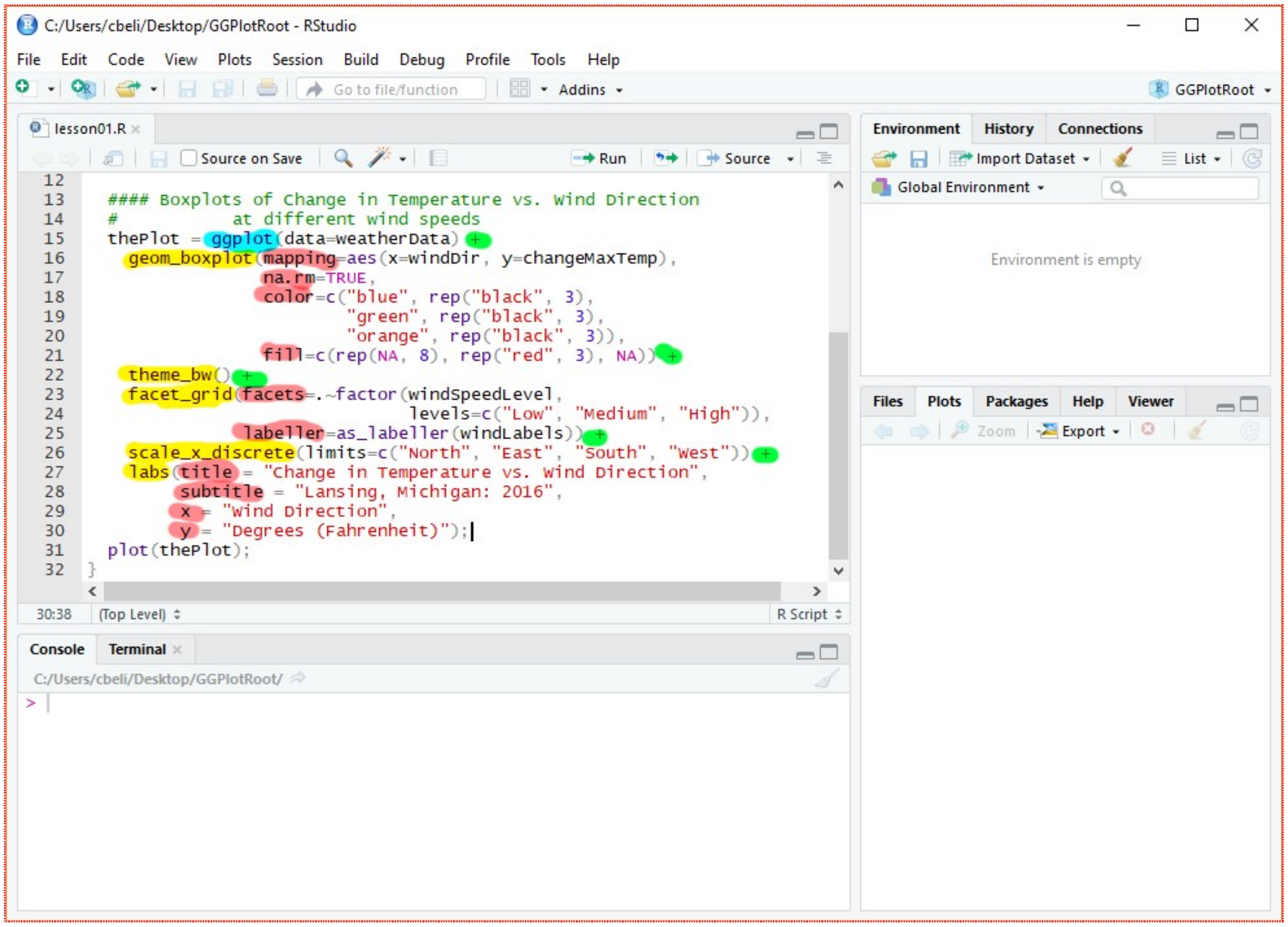
RStudio is a rapidly evolving program and it would be too much work to maintain all the images so that they reflect the visual likeness of the newest version of RStudio. Most of the images in this class will reflect the 2018 version of RStudio (v1.1) but changes to the images are often made and they will probably be made with a different version of RStudio. 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 issues!

7 - Your first programming topic: Spacing Code

In GGPlot, you create a plot by initializing a GGPlot canvas, with ***ggplot()***, and then add components (e.g., a boxplot) to the canvas. Most components will, in turn, have subcomponents, (e.g., color or width).

Fig 9 shows a plot that has:

- A function to initialize GGPlot -- highlighted in blue
- Five components -- highlighted in yellow
- **+** to "add" components to the canvas -- highlighted in green
- 10 subcomponents -- highlighted in red



C:/Users/cbeli/Desktop/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
GGPlotRoot
lesson01.R
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Zoom
Export
30:38
(Top Level)
R Script
Console Terminal
C:/Users/cbeli/Desktop/GGPlotRoot/

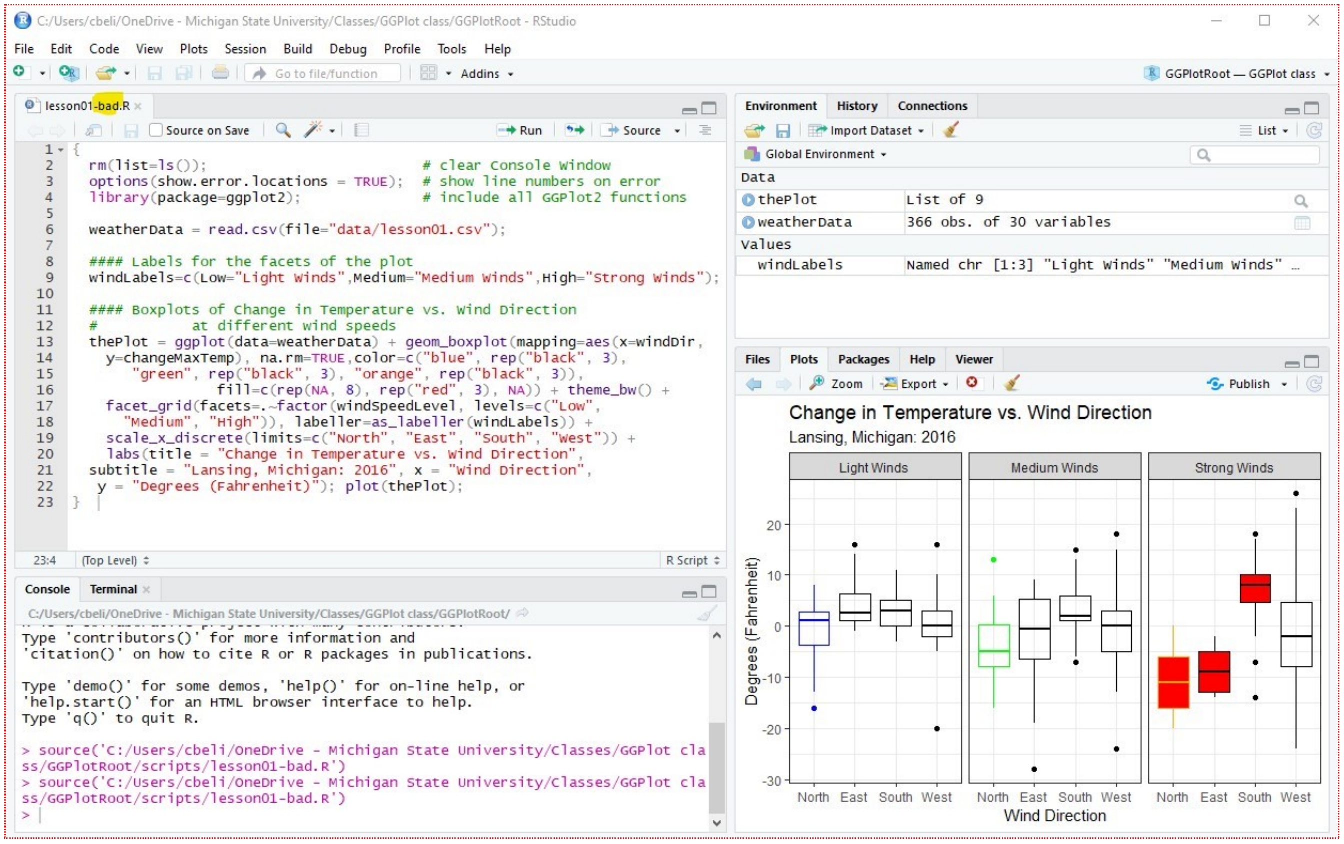
```
  #### Boxplots of Change in Temperature vs. Wind Direction
  #            at different wind speeds
  thePlot = ggplot(data=weatherData) +
    geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                 na.rm=TRUE,
                 color=c("blue", rep("black", 3),
                         "green", rep("black", 3),
                         "orange", rep("black", 3)),
                 fill=c(rep(NA, 8), rep("red", 3), NA)) +
    theme_bw() +
    facet_grid(facets=.~factor(windSpeedLevel,
                               levels=c("Low", "Medium", "High")),
               labeller=as_labeller(windLabels)) +
    scale_x_discrete(limits=c("North", "East", "South", "West")) +
    labs(title = "Change in Temperature vs. Wind Direction",
         subtitle = "Lansing, Michigan: 2016",
         x = "Wind Direction",
         y = "Degrees (Fahrenheit)");
  plot(thePlot);
}
```

```
>
```

Fig 9: Spacing code to emphasize the different components of a plot

7.1 - Component and subcomponent spacing

We will be going over the code in *fig 9* in much more detail in later lessons. For now, we are going to focus on the spacing. As you can see in *fig 10*, *the code for a plot can quickly get unwieldy* -- that is why it is important to consider code spacing from the very beginning. For this class, all components and subcomponents on a GGPlot will get its own line of code. This is also a requirement of your Class Project as it makes the code much easier to read.



```
{
  rm(list=ls());                         # clear Console Window
  options(show.error.locations = TRUE);  # show line numbers on error
  library(package=ggplot2);              # include all GGPlot2 functions

  weatherData = read.csv(file="data/lesson01.csv");

  #### Labels for the facets of the plot
  windLabels=c(Low="Light Winds",Medium="Medium Winds",High="Strong Winds");

  #### Boxplots of Change in Temperature vs. Wind Direction
  #           at different wind speeds
  thePlot = ggplot(data=weatherData) + geom_boxplot(mapping=aes(x=windDir,
    y=changeMaxTemp), na.rm=TRUE,color=c("blue", rep("black", 3),
       "green", rep("black", 3), "orange", rep("black", 3)),
                fill=c(rep(NA, 8), rep("red", 3), NA)) + theme_bw() +
    facet_grid(facets=.~factor(windSpeedLevel, levels=c("Low",
      "Medium", "High")), labeller=as_labeller(windLabels)) +
    scale_x_discrete(limits=c("North", "East", "South", "West")) +
    labs(title = "Change in Temperature vs. Wind Direction",
  subtitle = "Lansing, Michigan: 2016", x = "Wind Direction",
   y = "Degrees (Fahrenheit)"); plot(thePlot);
}
```

```
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.

> source('C:/Users/cbeli/OneDrive - Michigan State University/Classes/GGPlot cla
ss/GGPlotRoot/scripts/lesson01-bad.R')
> source('C:/Users/cbeli/OneDrive - Michigan State University/Classes/GGPlot cla
ss/GGPlotRoot/scripts/lesson01-bad.R')
>
```

```
Global Environment
Data
thePlot        List of 9
weatherData    366 obs. of 30 variables
Values
windLabels     Named chr [1:3] "Light Winds" "Medium Winds" …
```

Fig 10: Poorly spaced code -- it works but it is too hard to read!

7.2 - GGPlot Functions are components

The components of a GGPlot are really the functions in the GGPlot package. The subcomponents are the parameters of those functions.

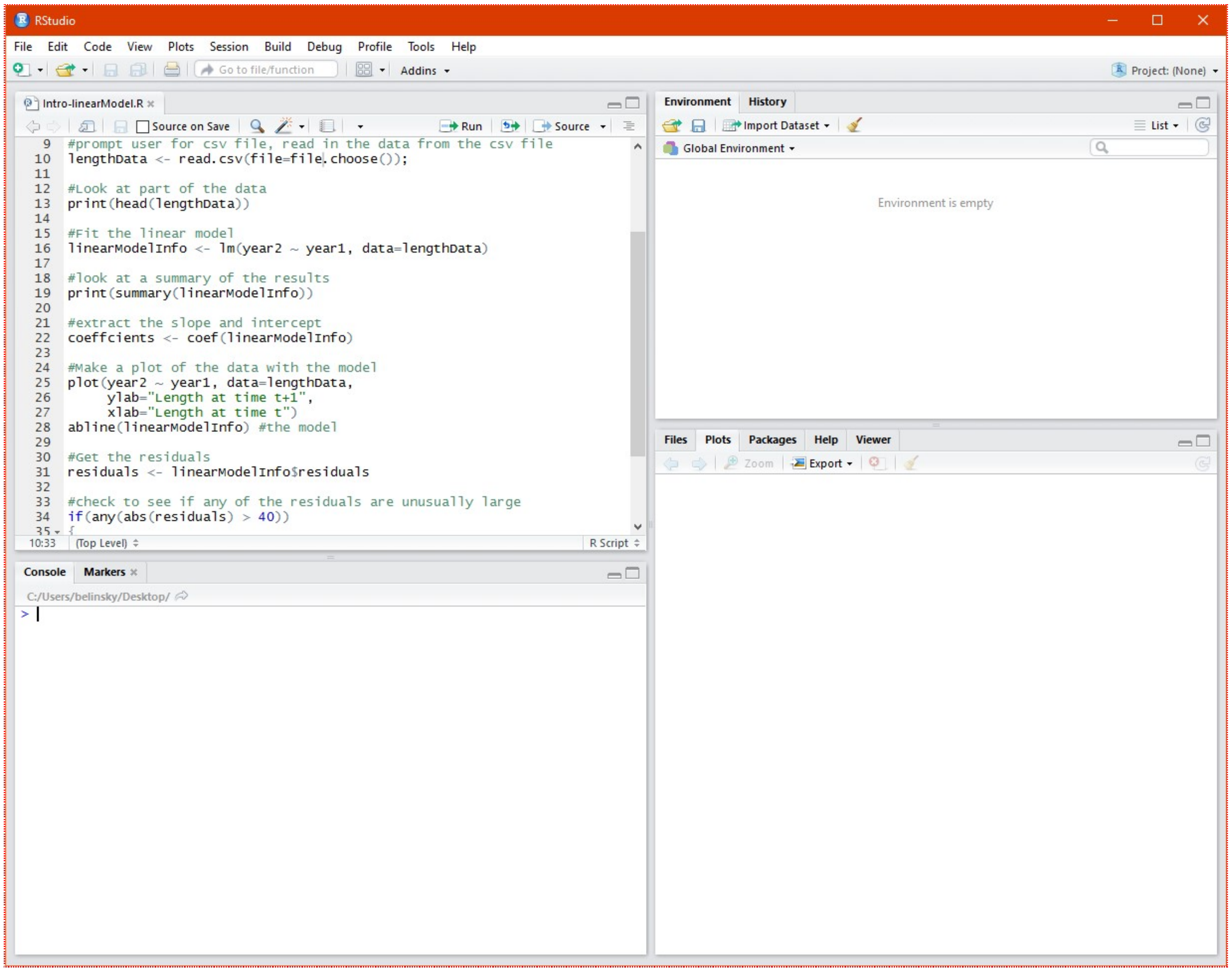
One component (i.e., function) in the GGPlot package is ***labs()***, which is the labeling component in GGPlot. ***labs()*** has multiple subcomponents (i.e., parameters) representing the different parts of the plot that can be labelled. ***labs()*** was used in *fig 9* to add a ***title*** and ***subtitle*** and ***x***-axis and ***y***-axis labels. It looks like this:

```
# component (function) labs() with four subcomponents (parameters) being modified
labs(title = "Change in Temperature vs. Wind Direction",
     subtitle = "Lansing, Michigan: 2016",
     x = "Wind Direction",
     y = "Degrees (Fahrenheit)")
```

In this class, we will refer to ***labs()*** as a component and ***title***, ***subtitle***, ***x***, and ***y*** as subcomponents.

8 - 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. In *fig 11*, there are comments on lines 9, 12, 15, 18, etc. These comments are colored light green but the color of the comments depends on your color scheme (next section).



```
#prompt user for csv file, read in the data from the csv file
lengthData <- read.csv(file=file.choose());

#Look at part of the data
print(head(lengthData))

#Fit the linear model
linearModelInfo <- lm(year2 ~ year1, data=lengthData)

#look at a summary of the results
print(summary(linearModelInfo))

#extract the slope and intercept
coeffcients <- coef(linearModelInfo)

#Make a plot of the data with the model
plot(year2 ~ year1, data=lengthData,
     ylab="Length at time t+1",
     xlab="Length at time t")
abline(linearModelInfo) #the model

#Get the residuals
residuals <- linearModelInfo$residuals

#check to see if any of the residuals are unusually large
if(any(abs(residuals) > 40))
{
```

Environment is empty

```
C:/Users/belinsky/Desktop/
> 
```

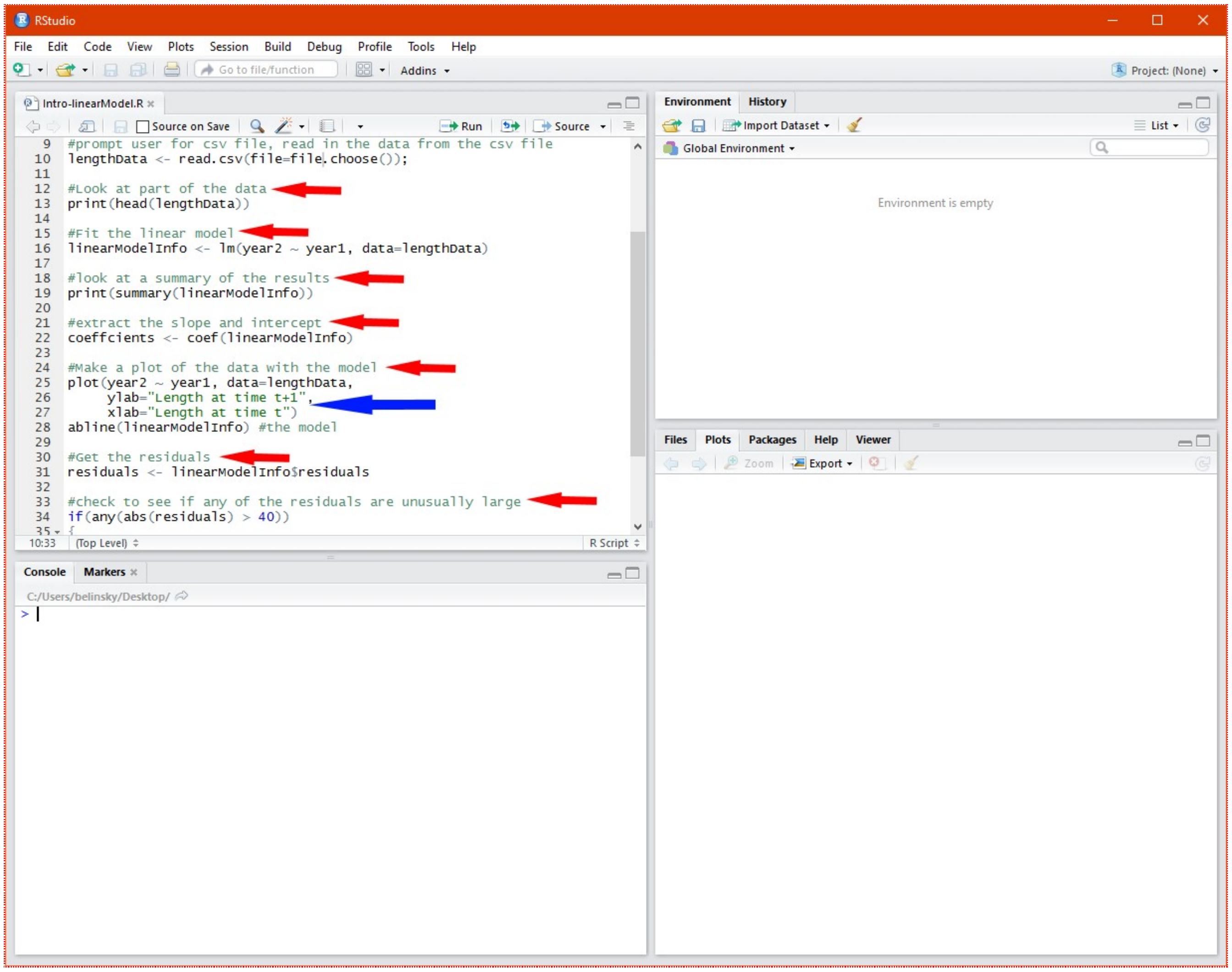
Fig 11: Comments are in the lighter green color

You are required to make 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 11*. In this color scheme the comments are in a lighter green whereas most of the other text is in black.

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 12*).



```
#prompt user for csv file, read in the data from the csv file
lengthData <- read.csv(file=file.choose());

#Look at part of the data
print(head(lengthData))

#Fit the linear model
linearModelInfo <- lm(year2 ~ year1, data=lengthData)

#look at a summary of the results
print(summary(linearModelInfo))

#extract the slope and intercept
coeffcients <- coef(linearModelInfo)

#Make a plot of the data with the model
plot(year2 ~ year1, data=lengthData,
     ylab="Length at time t+1",
     xlab="Length at time t")
abline(linearModelInfo) #the model

#Get the residuals
residuals <- linearModelInfo$residuals

#check to see if any of the residuals are unusually large
if(any(abs(residuals) > 40))
{
```

```
C:/Users/belinsky/Desktop/
>
```

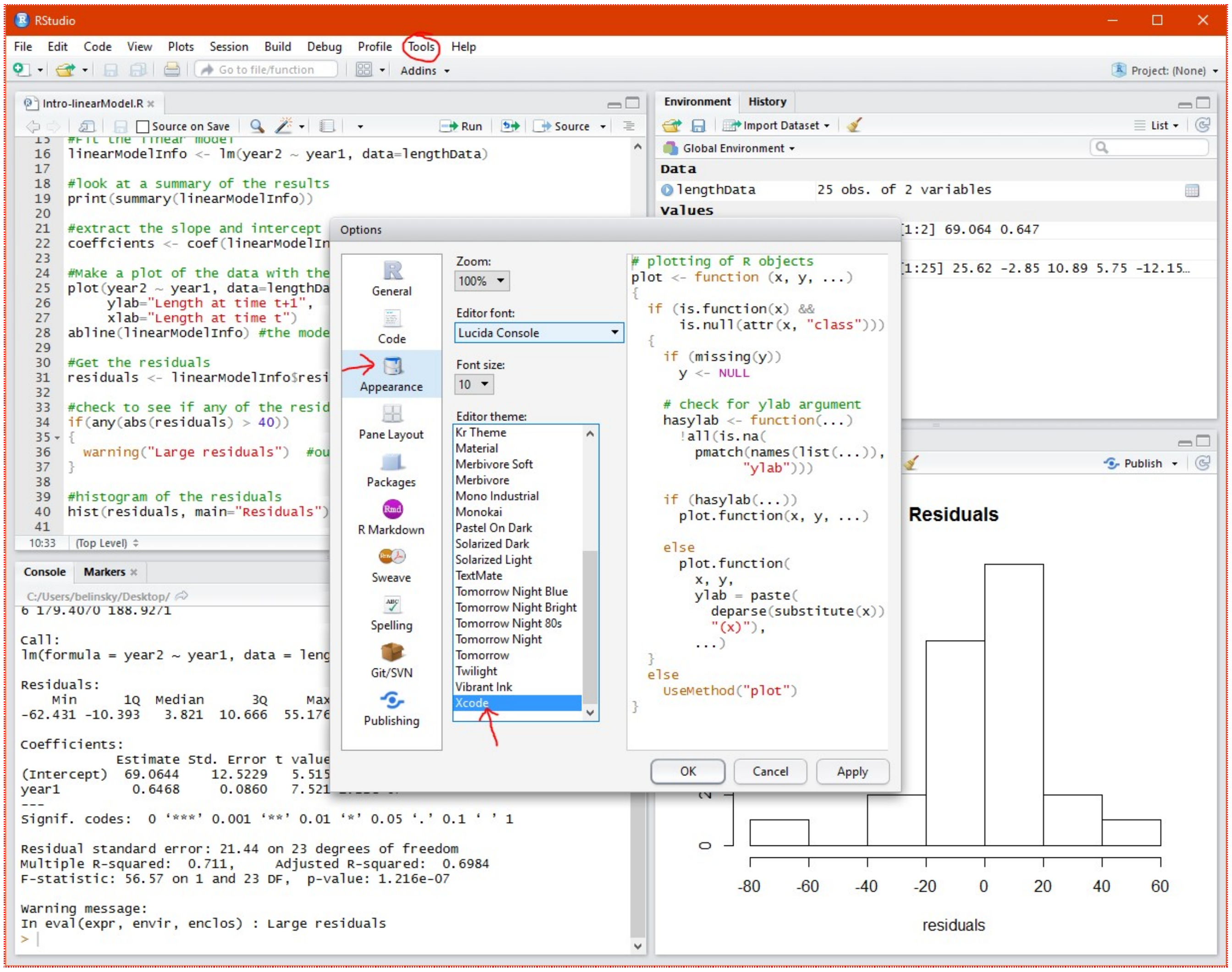
Fig 12: The text color for comments and quotes

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 -- you can change the color scheme by:

1. clicking on ***Tools*** in the main menu (circled in *fig 13*)
2. choose ***Global Options***
3. When the ***Global Options*** window open (in *fig 13*), click on ***Appearance***



```
#Fit the linear model
linearModelInfo <- lm(year2 ~ year1, data=lengthData)

#look at a summary of the results
print(summary(linearModelInfo))

#extract the slope and intercept
coeffcients <- coef(linearModelIn

#Make a plot of the data with the
plot(year2 ~ year1, data=lengthDa
     ylab="Length at time t+1",
     xlab="Length at time t")
abline(linearModelInfo) #the mode

#Get the residuals
residuals <- linearModelInfo$resi

#check to see if any of the resid
if(any(abs(residuals) > 40))
{
  warning("Large residuals")  #ou
}

#histogram of the residuals
hist(residuals, main="Residuals")
```

Data
lengthData 25 obs. of 2 variables
Values
[1:2] 69.064 0.647
[1:25] 25.62 -2.85 10.89 5.75 -12.15…

Zoom: 100%

Editor font: Lucida Console

Font size: 10

Editor theme:
- Kr Theme
- Material
- Merbivore Soft
- Merbivore
- Mono Industrial
- Monokai
- Pastel On Dark
- Solarized Dark
- Solarized Light
- TextMate
- Tomorrow Night Blue
- Tomorrow Night Bright
- Tomorrow Night 80s
- Tomorrow Night
- Tomorrow
- Twilight
- Vibrant Ink
- Xcode

```
# plotting of R objects
plot <- function (x, y, ...)
{
  if (is.function(x) &&
      is.null(attr(x, "class")))
  {
    if (missing(y))
      y <- NULL

    # check for ylab argument
    hasylab <- function(...)
      !all(is.na(
        pmatch(names(list(...)),
               "ylab")))

    if (hasylab(...))
      plot.function(x, y, ...)

    else
      plot.function(
        x, y,
        ylab = paste(
          deparse(substitute(x)),
          "(x)"),
        ...)
  }
  else
    UseMethod("plot")
}
```

```
6 179.4070 188.9271

Call:
lm(formula = year2 ~ year1, data = leng

Residuals:
    Min      1Q  Median      3Q     Max
-62.431 -10.393   3.821  10.666  55.176

Coefficients:
            Estimate Std. Error t value
(Intercept)  69.0644    12.5229   5.515
year1         0.6468     0.0860   7.521
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 21.44 on 23 degrees of freedom
Multiple R-squared:  0.711,     Adjusted R-squared:  0.6984
F-statistic: 56.57 on 1 and 23 DF,  p-value: 1.216e-07

Warning message:
In eval(expr, envir, enclos) : Large residuals
>
```

Fig 13: Color schemes for RStudio (note: the Editor font might be different in your window -- that is OK)

The image above shows the *Xcode* color scheme (*fig 13*). 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).

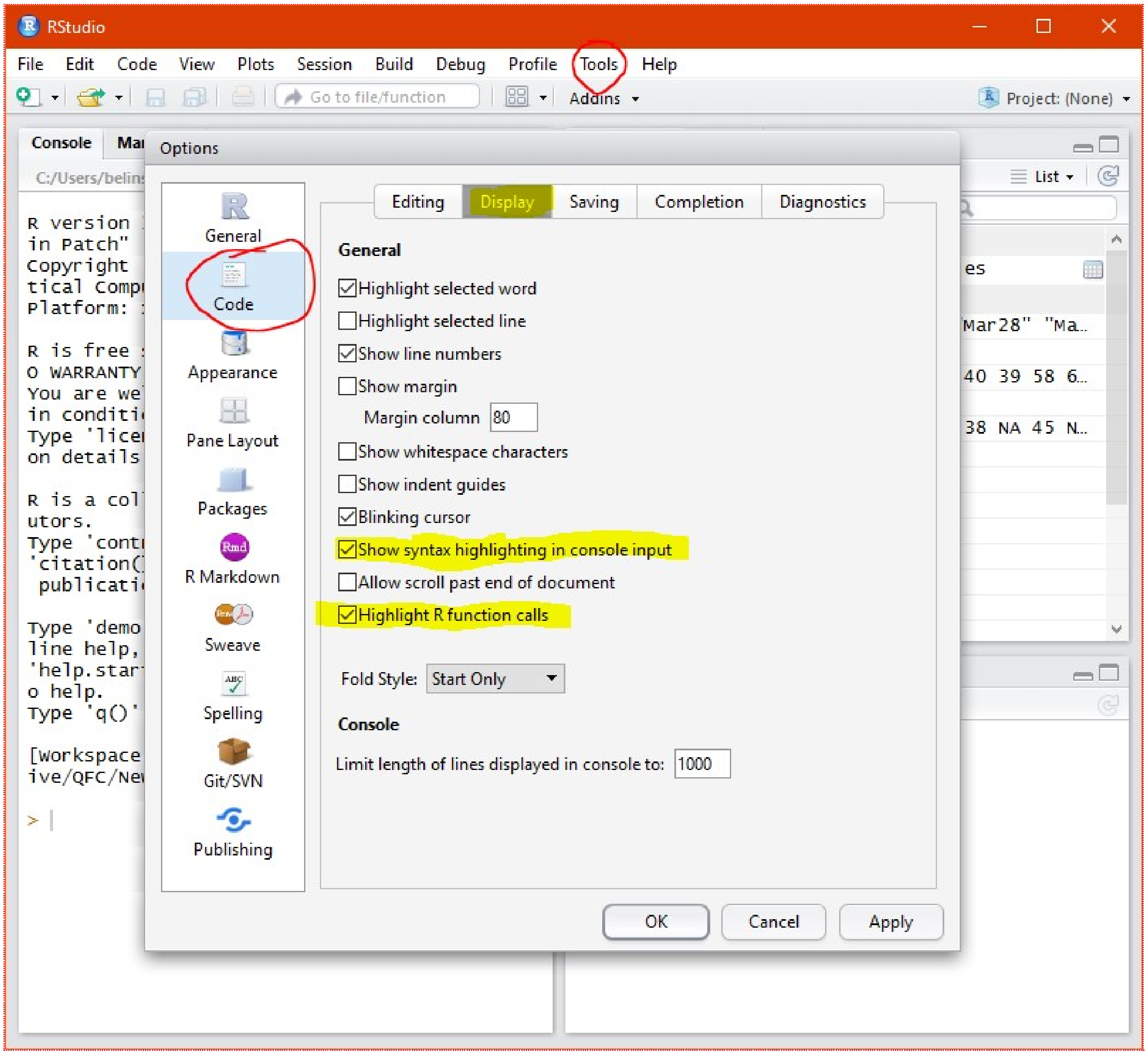
You can choose from around 20 themes in the **Editor theme** window and you can change the theme anytime without affecting anything else.

9.2 - Adding more color to differentiate output

There are a couple more helpful options in RStudio that use color to distinguish components of your code and the output in the Console Window.

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*



General

- [✓] Highlight selected word
- [ ] Highlight selected line
- [✓] Show line numbers
- [ ] Show margin
- Margin column 80
- [ ] Show whitespace characters
- [ ] Show indent guides
- [✓] Blinking cursor
- [✓] Show syntax highlighting in console input
- [ ] Allow scroll past end of document
- [✓] Highlight R function calls

Fold Style: Start Only

Console

Limit length of lines displayed in console to: 1000

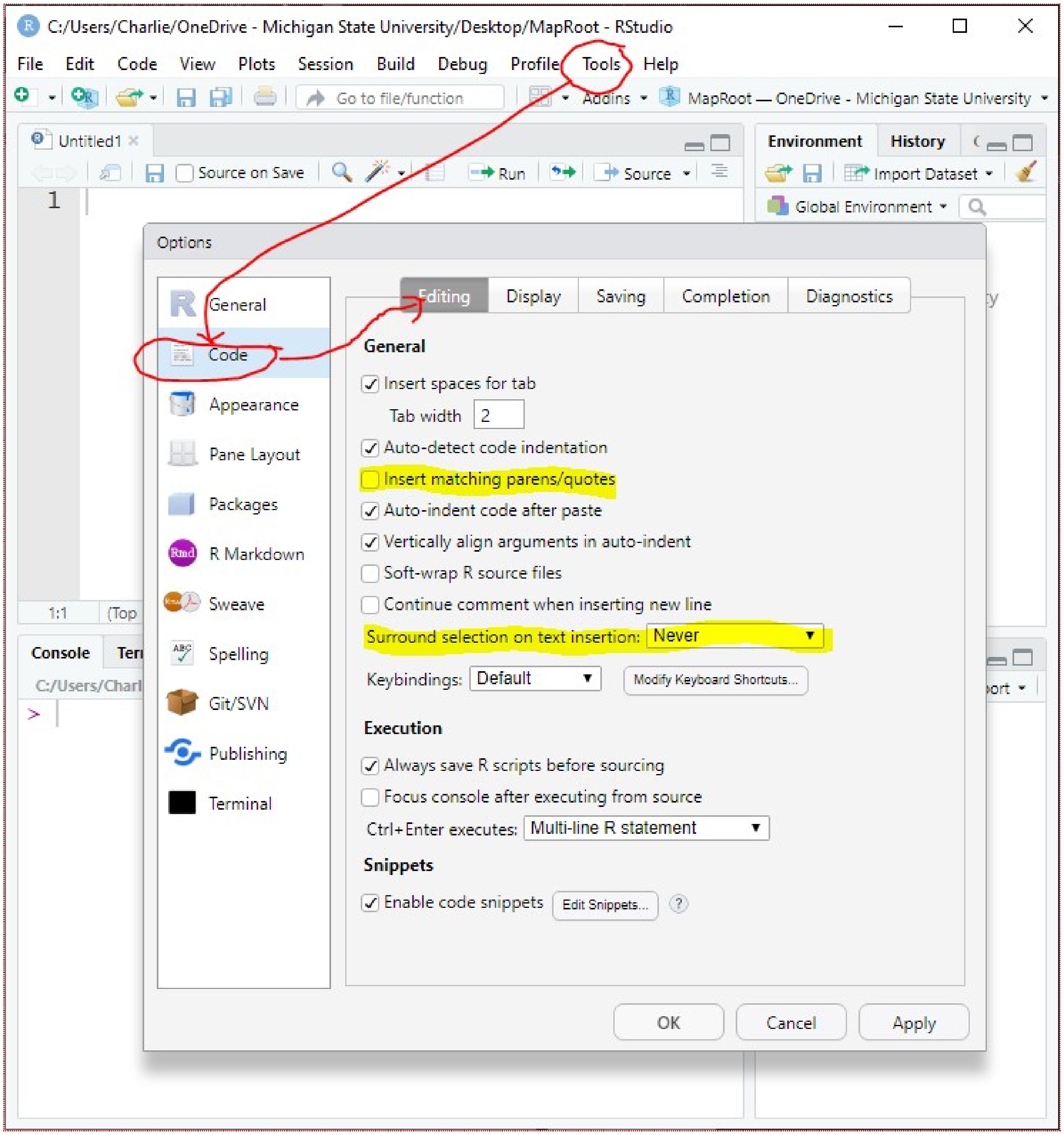
Fig 14: Console Window changes

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***



General

- [✓] Insert spaces for tab
- Tab width: 2
- [✓] Auto-detect code indentation
- [ ] Insert matching parens/quotes
- [✓] Auto-indent code after paste
- [✓] Vertically align arguments in auto-indent
- [ ] Soft-wrap R source files
- [ ] Continue comment when inserting new line
- Surround selection on text insertion: Never
- Keybindings: Default

Execution

- [✓] Always save R scripts before sourcing
- [ ] Focus console after executing from source
- Ctrl+Enter executes: Multi-line R statement

Snippets

- [✓] Enable code snippets

Fig 15: Stop RStudio from automatically ending your parenthesis and quotes.

11 - Application

1. In your RStudio Project, create a new script file (***File*** -> ***New File*** -> ***R Script***)
2. Copy and paste this lesson's script, ***lesson01.r***, to the new script.
3. Change the colors in the boxplot (edit lines 17-21)
4. Save the script as ***app01.r***, and to your ***script*** folder
5. Zip you Root Folder (instructions for Windows and Mac below)
6. *Email the zipped Root Folder to your instructor*

Windows: Zip your Root Folder (*fig 16*)**:**

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*** is created with all the contents of the Root Folder

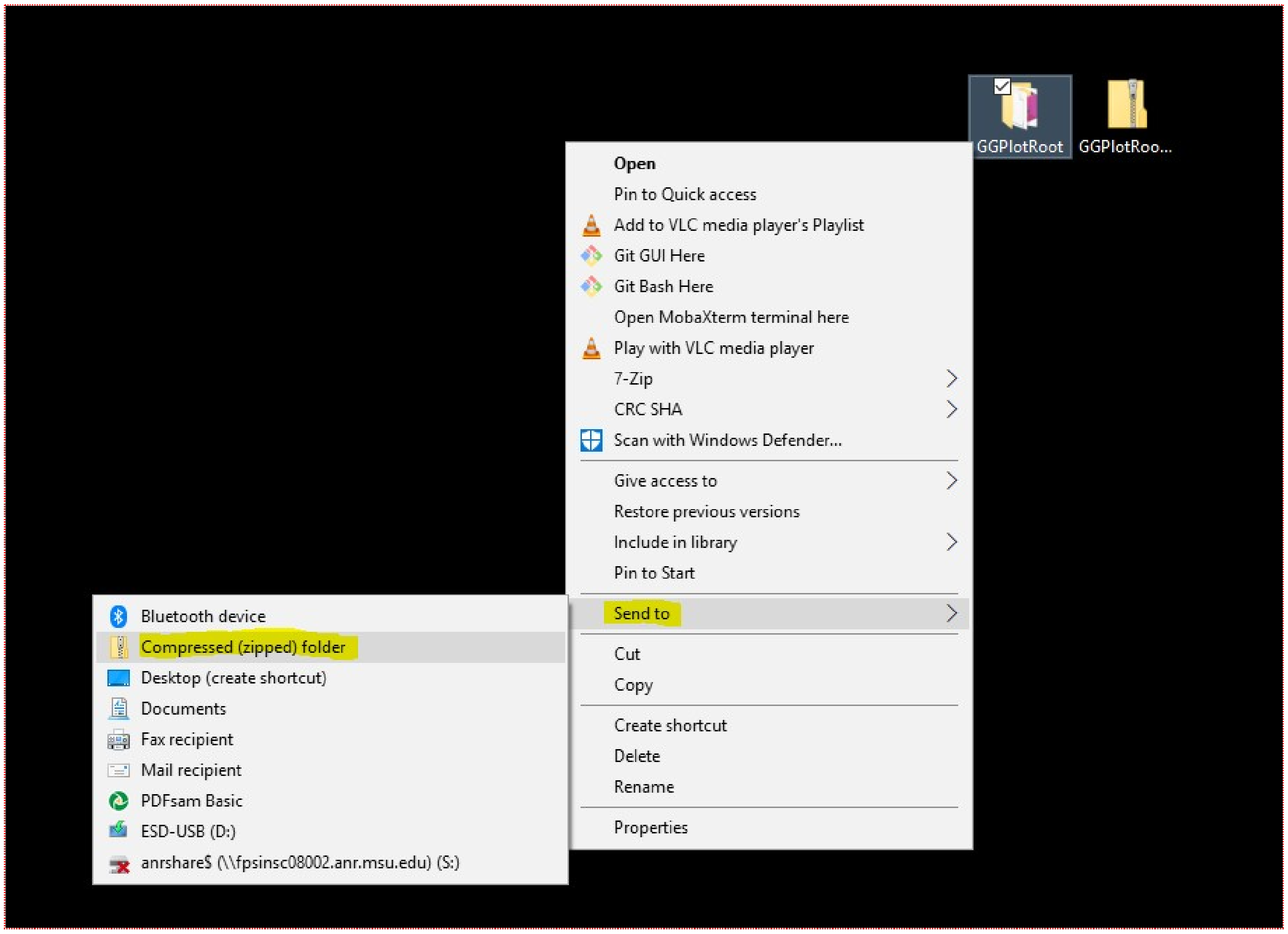


*Fig 16: Zipping your Class Folder on Windows -- in this case the zipped file created is named **GGPlotRoot.zip***

Mac: Zip your Root Folder:

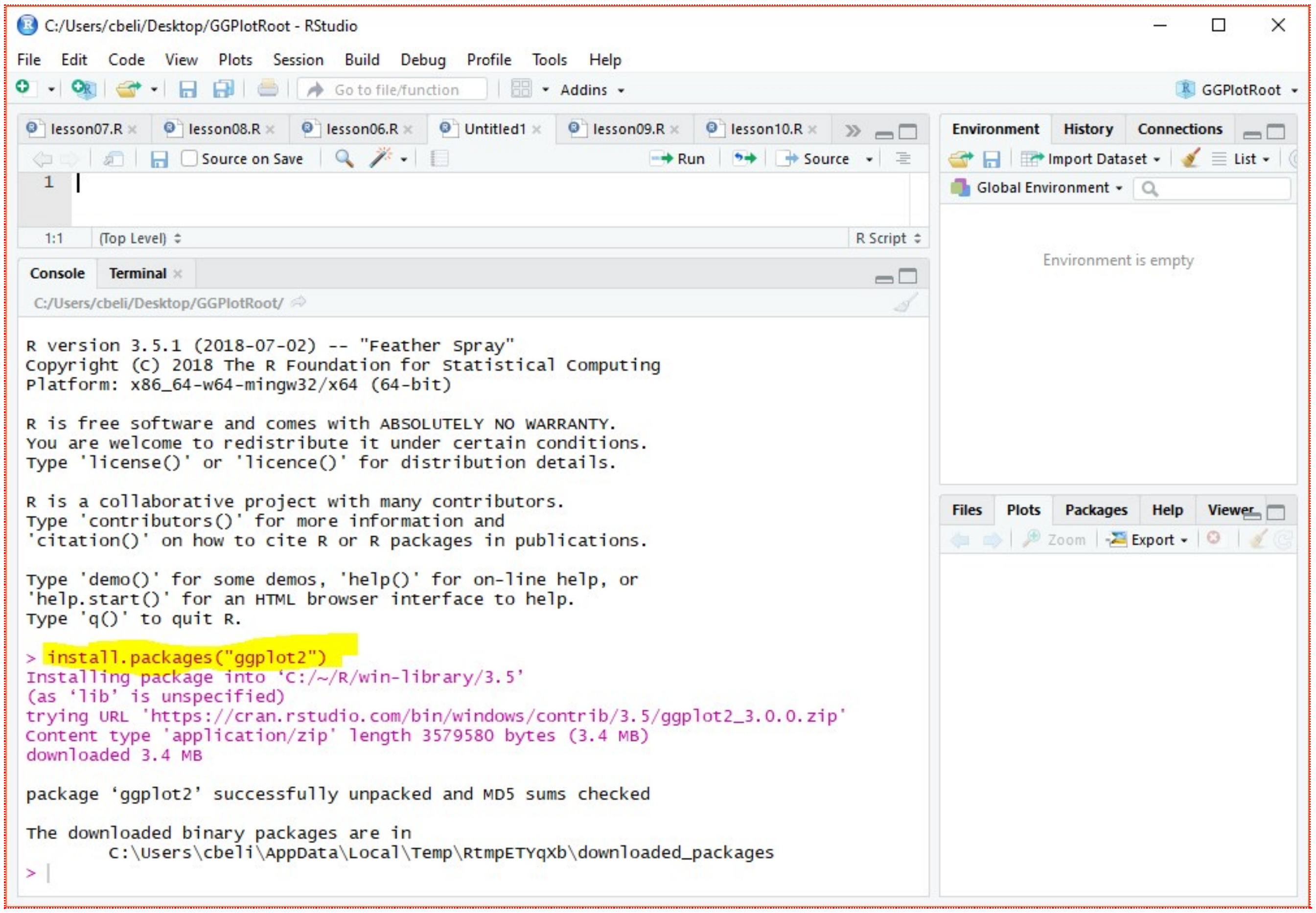
1. In your File Manager (not in RStudio), right-click on the Root Folder
2. Choose ***Compress "<Root Folder>"***
3. A zipped file named ***<Root Folder>.zip*** is created with all the contents of the Root Folder

12 - Extension: Installing packages using R

In R, we can install packages using ***install.packages()***. So, if you wanted to install GGPlot2, you would type:

```
install.packages("GGPlot2")
```

The Console Window in RStudio is an R interface, and you could type in the above line to install the ***GGPlot2*** package.

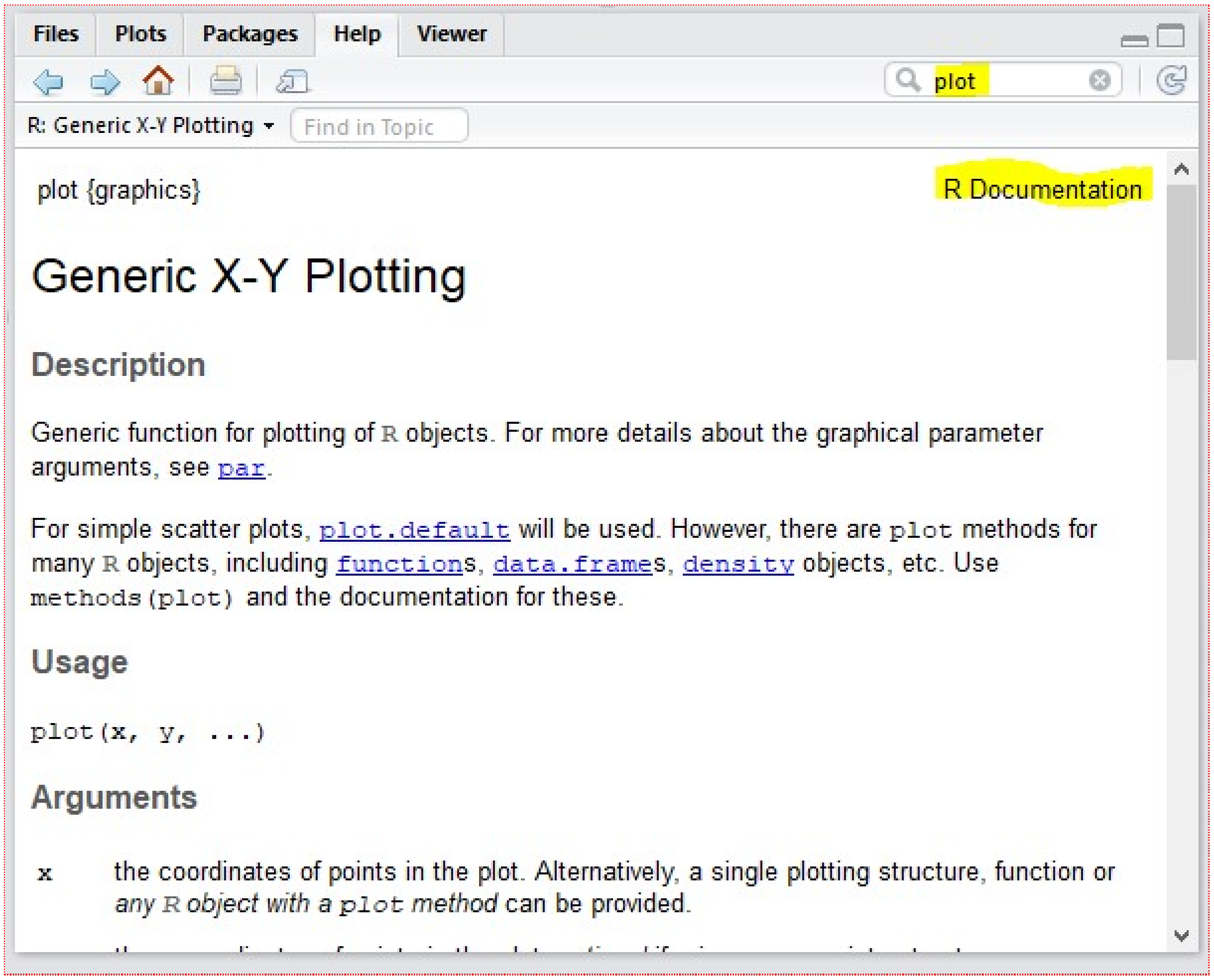


```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

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.

> install.packages("ggplot2")
Installing package into 'C:/~/R/win-library/3.5'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/ggplot2_3.0.0.zip'
Content type 'application/zip' length 3579580 bytes (3.4 MB)
downloaded 3.4 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\cbeli\AppData\Local\Temp\RtmpETYqXb\downloaded_packages
> 
```

Fig 17: Using the RStudio Console to install a package

13 - Extension: The Help Tab

The Help Tab is essentially an intelligent online search through the R documentation. So, if you type **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.***



plot {graphics} R Documentation

Generic X-Y Plotting

Description

Generic function for plotting of R objects. For more details about the graphical parameter arguments, see `par`.

For simple scatter plots, `plot.default` will be used. However, there are `plot` methods for many R objects, including `function`s, `data.frame`s, `density` objects, etc. Use `methods(plot)` and the documentation for these.

Usage

```
plot(x, y, ...)
```

Arguments

x the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any* R *object with a* `plot` *method* can be provided.

Fig 18: The R plot help page in the online documentation.

The page that appears in the **Help Window** (*fig 18*) 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. The script you are developing in this class are meant to be fully executed -- just like programming in C. So we will always be using the ***Source*** button.

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 19*).

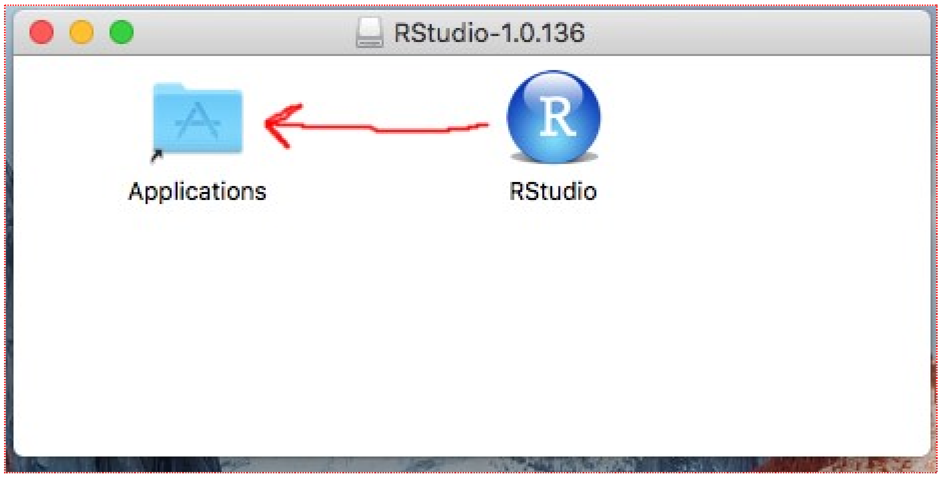


Fig 19: Opening RStudio DMG file

3. In the RStudio window above (*fig 19*) drag the RStudio file to the Applications folder

Unmount the RStudio device by clicking the Eject button -- highlighted below (*fig 20*)

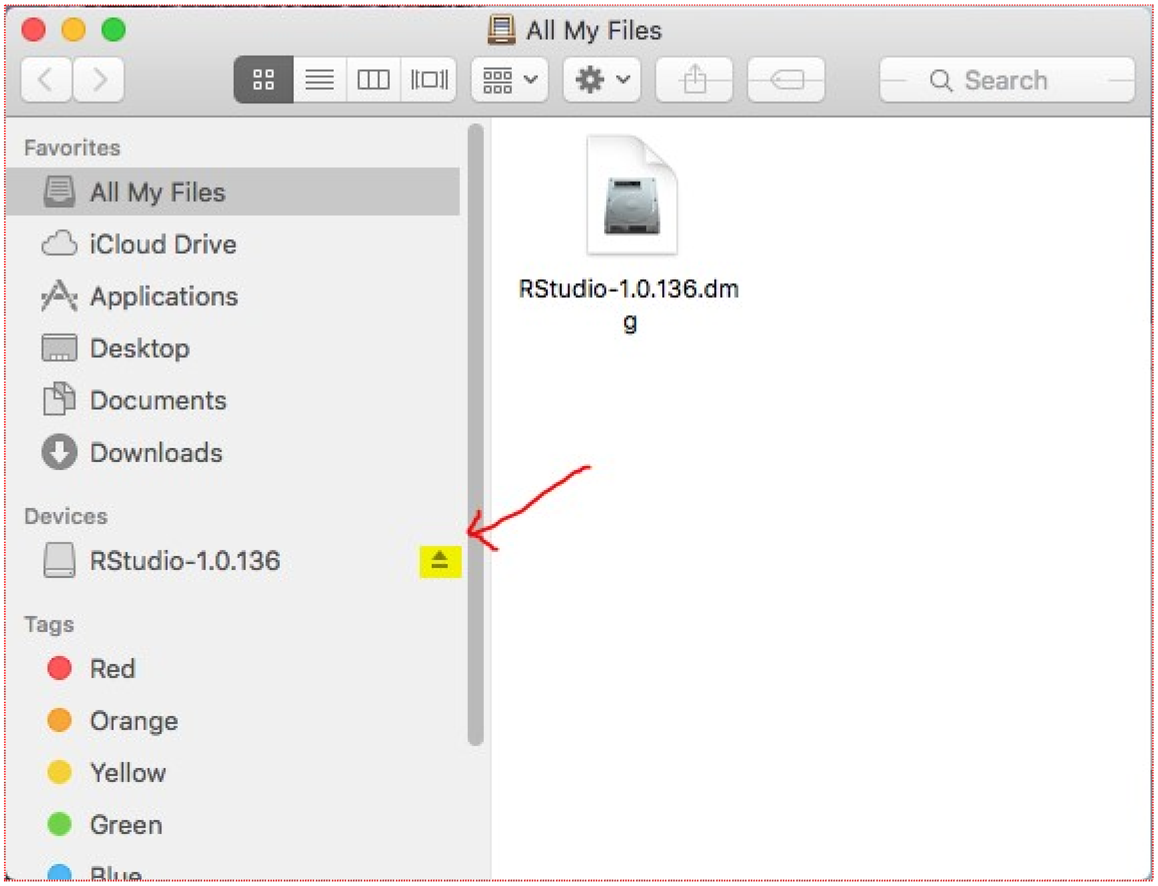


Fig 20: Unmounting the RStudio device (very important!)

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

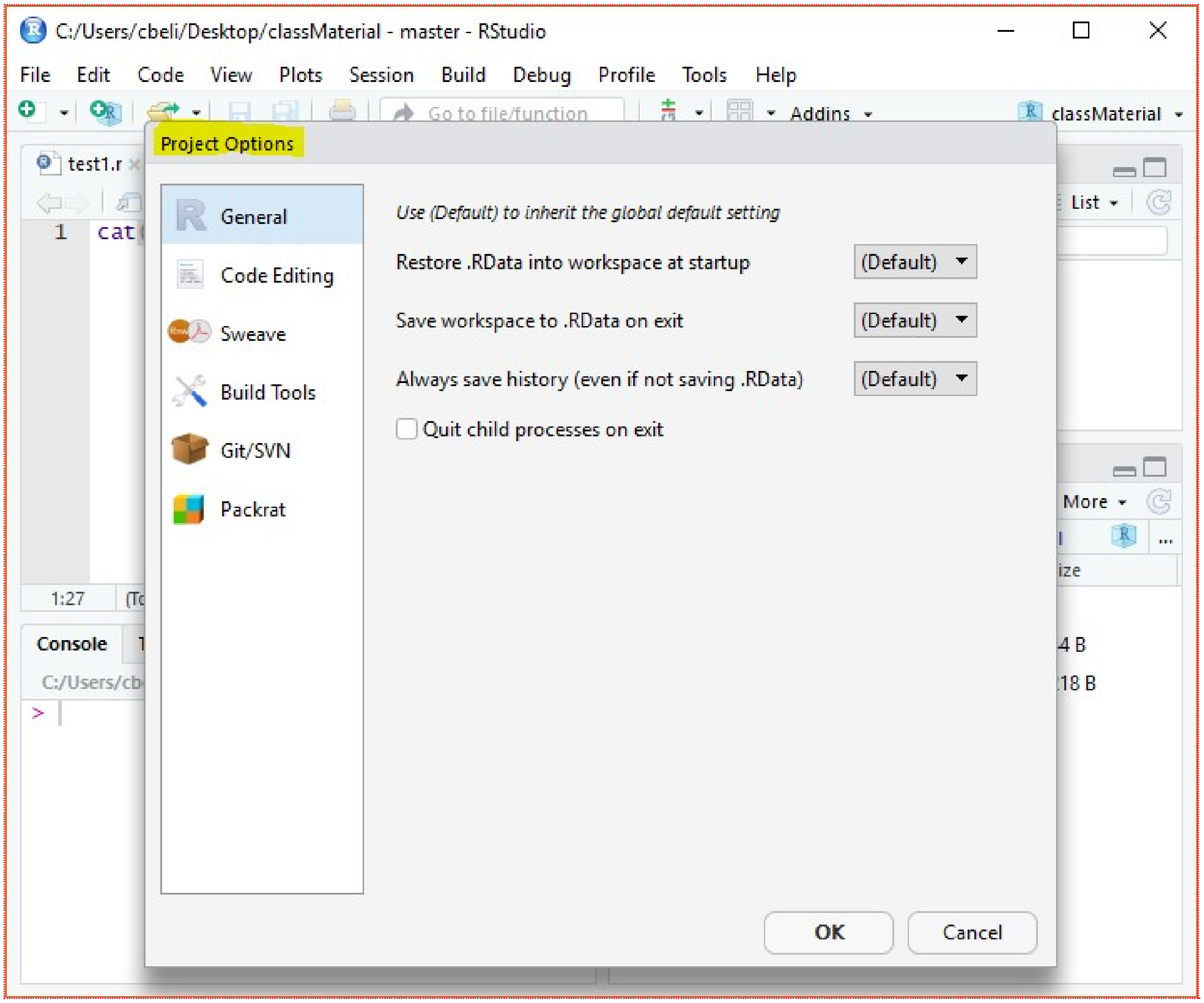
Many people use the first method. In this class we use the second method. The first method of creating a script file is good if you want to test something out. The second method is better for organizing larger projects or sharing your code with others.

Without getting into gory detail of the issues, problems with the first method occur when you reading from another file, like a data file. This is because the first method does not explicitly declare a Root Folder like an RStudio Project does.

R Programmers often get around the lack of a Root Folder issue by setting the working directory in code.

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 21*), which you can also get to by clicking on ***Tools -> Project Options.***



Use (Default) to inherit the global default setting

Restore .RData into workspace at startup (Default)

Save workspace to .RData on exit (Default)

Always save history (even if not saving .RData) (Default)

Quit child processes on exit

Fig 21: 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.