1 THE SCIENTIFIC STUDY OF POLITICS

1.1 OVERVIEW

In this chapter we introduce you to some of the important building blocks of a scientific approach to studying politics. As you can already tell from reading the first chapter of the Third Edition of *The Fundamentals of Political Science Research* – which we will refer to as "*FPSR*" from here on – data are an important part of what we do both to explore the political world and to test hypotheses based on causal theories. An important part of working with data is learning how to use a statistical software package. In the sections that follow, we introduce you to the R program and some basics that you will need to get up and running. In doing this, we also introduce some general principles of good computing practices for effectively working with data.

1.2 "A WORKBOOK? WHY IS THERE A WORKBOOK?"

You might be asking yourself this question, and it's perfectly fair to do so. Allow us to try to explain how this workbook fits in with the main *FPSR* text.

As you will see in the weeks and months to follow in your class, the main textbook will teach you about the use of statistics in political science, mostly by using equations and examples. So yes, in some ways, it will feel rather math-y. (And we think that's cool, though we realize that it's not everyone's cup of tea.) One of the ways that people learn about the practice of statistics is to use computer software to calculate statistics directly. To that end, many instructors want students to learn to use a particular computer software package so they can begin to conduct statistical analyses themselves.[1] We have discovered through years of teaching that this transition between equations in a book and software output on a computer screen is a very difficult one. The goal of this software companion book is to make this connection stronger, even seamless.

If we are successful, this book will do two things. First, it will teach the nuts and bolts about how to use R. Though many (perhaps most) students today are quite computer-literate, we believe that having a reference guide for students to learn the techniques, or for them to teach themselves out of class time, will be helpful. Second, and more importantly, this software guide will provide explicit hand-holding to you as you learn

[1] This particular software companion book teaches students to use R, but we have also produced parallel books for instructors who wish to have their students learn SPSS or Stata.

to connect the key principles from the main text to the practical issues of producing and interpreting statistical results.

Each chapter of this software guide works in parallel with that of the main *FPSR* text. So when you learn the equations of two-variable regression analysis in Chapter 9 of the main text, you will learn the details about using R to estimate two-variable regression models in Chapter 9 of this companion book. And so on. In the end, we hope that the very important (but perhaps rather abstract) equations in the text become more meaningful to you as you learn to estimate the statistics yourself, and then learn to interpret them meaningfully and clearly. Those three things – formulae, software, and interpretation – together provide a very solid foundation and basic understanding of social science.

1.2.1 Reading Commands in This Workbook

Throughout this workbook, when we present R commands in their text versions, we will write out commands in typewriter type to help distinguish actual R commands from our instructional text. For instance,

```
summary(Reg1)
```

is an example of a command that we would issue if we had previously estimated a regression model which we called "`Reg1`" and we wanted R to produce a summary table of the results. Occasionally, we will also use italics within a command line to indicate a place in the code where something needs to be placed, but where the exact text of the command depends on what the user is trying to accomplish with that command. So, for instance, in the next chapter where we demonstrate one way to install a new package using the "`install.packages`" command, we will write the following code:

```
install.packages("PackageName")
```

where "*`PackageName`*" is not what one should enter into the command line (there is not a package named "*`PackageName`*"), but rather a placeholder written in italics to indicate that one should enter in that part of the code the name of the package that they want to install.

1.3 GETTING STARTED WITH R AND RSTUDIO

To get started with R and RStudio, we recommend that you set yourself up in front of a computer that has both of the programs installed with a copy of the third edition of *FPSR* close by. You should also have the set of computer files that accompany this text (which you can download from the text's website, `www.cambridge.org/fpsr`) in a directory on the computer on which you are working. You will get the most out of this workbook by working in RStudio as you read this workbook.

The instructions in this book can help you learn R whether you use a Windows-based PC or a Mac. Once the program is launched, R in RStudio works identically, no matter which platform you use. Mac users should be aware, though, that our screenshots will

come from a Windows-based PC. Some of those screenshots that involve finding and opening files on your computer, therefore, will look somewhat unfamiliar to Mac users, but we assume that Mac users are at least somewhat used to this. Overall, the differences between running R in RStudio on Windows compared to a Mac are minimal.

Finally, we wrote this book while using versions 4.0.3 of R and the free version of RStudio 1.3.1093. Particularly for the statistical fundamentals you will learn in this book, the differences between versions are not severe. But, since both programs are available for free and easy to install, we recommend that you use the versions that we are using or a newer version of these programs.

1.3.1 Launching RStudio

When you are sitting in front of a computer on which RStudio and R have been properly installed, you can launch the program by double-clicking on the RStudio icon or by finding the RStudio program on your start menu. At this point, you should see one large window like that in Figure 1.1. Within this main window, you will see three other windows labeled “Console” (on the left side), “Environment” (on the top right side), and “Plots” (on the bottom right side). If you are seeing all of this, you are ready to go.

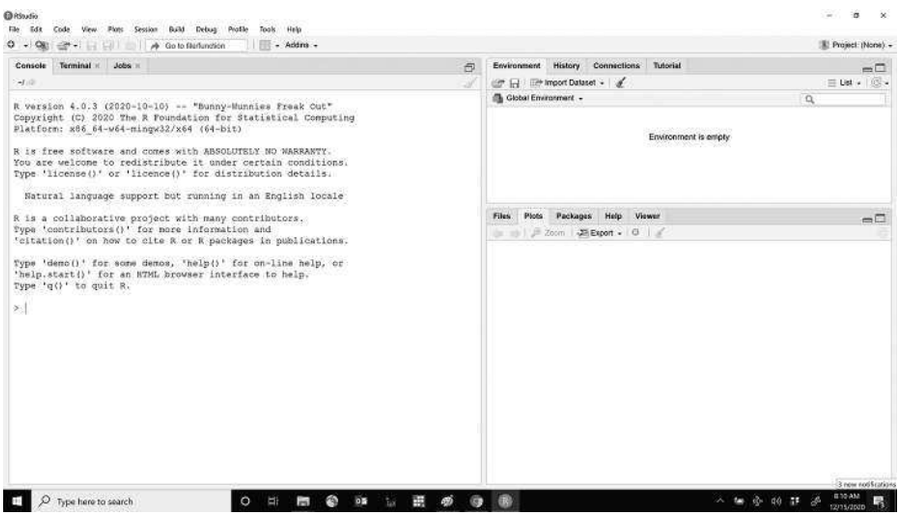

Figure 1.1 RStudio initial launch

1.3.2 Getting R to Do Things

In almost any mainstream statistical program today, there are multiple ways to accomplish the same tasks. In R in RStudio, the two most common ways to execute commands are by typing them in the console next to the blue > symbol or by typing them into a

script in the editor, selecting the command, and hitting the "run" button. The choice of which of these options to use is a matter of personal comfort. But, as we discuss below, no matter which way you choose to get R to do things, you need to keep track of what you are doing. We now discuss the two ways to get R to do things by showing an example of opening a data set. We recommend that you try both, but especially the example of using a script in this section.

Typing Commands in the Console Window

You can type commands directly into the console window that you see on the left side of RStudio when you launch the program. These commands are typed in one at a time and are executed by the program when you hit the "Enter" button on your keyboard.

So, if you want to load the data set "EcoVote" which is an R-format data set (with the ".RData" suffix), you would type the following command into the "Console" window and hit the "Enter" key on your computer:

```
load("C:/MyFPSRrFiles/EcoVote.RData")
```

To check whether you have done this correctly, you can click the tab labeled "Environment" in the upper-right window of RStudio. If you have successfully loaded the data, your RStudio will look like Figure 1.2 where we now see the name of the data set that we were attempting to load and a brief summary of how many observations and variables are in the data set.[2]

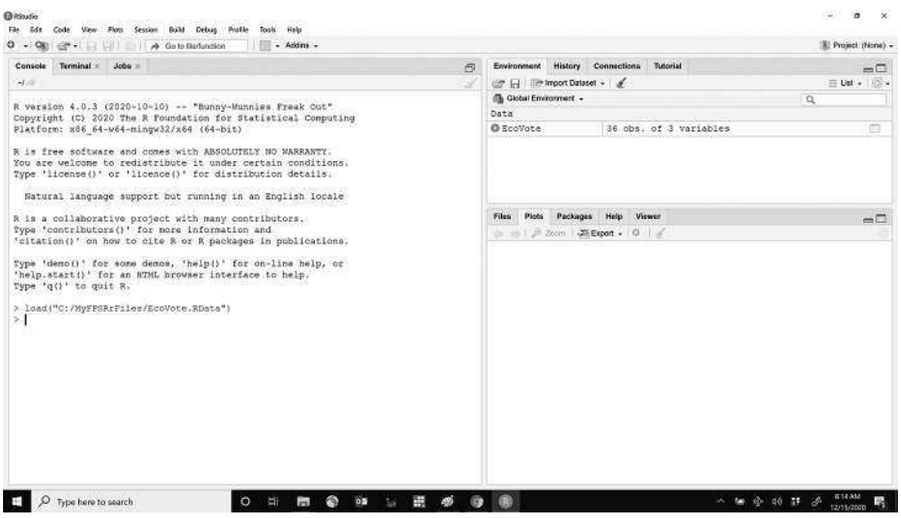

Figure 1.2 Data set successfully loaded into R

[2] The location of files is often a stumbling block for beginner users of a statistical software package. To keep things simple, we recommend that you create a folder on your computer's C drive named "MyFPSRrFiles" and put all of the files that you have downloaded from `www.cambridge.org/fpsr` into that folder. If you are unable to do this, then on a computer using a Windows operating system you can find the exact name of the location of a file by right-clicking on that file, left-clicking "Properties," and then looking at the entry to the right of "Location." This filepath, or location, can be selected, copied, and pasted directly into your console window (or script) to insure that it is exactly right. But, it is important to note that the slashes to tell R the location of a file are forward slashes instead of the usual Windows convention of using backslashes.

Using a Script File

A second way to issue commands in RStudio is to use a script file. While this method of working will seem a little bit clumsy at first, it is our preferred method of working in RStudio for reasons that we will explain below. To start work with a new script file, you need to left-click on the "file" tab in the very upper-left corner of the program, and then from the menu that drops down select "New File," followed by "R Script." Once you have successfully done this, your RStudio should look like Figure 1.3. We will eventually cover a lot of different things that one can do with a script file, but for now, all that we want you to do is to type the following command into the new script file:

```
load("C:/MyFPSRrFiles/EcoVote.RData")
```

Once you have typed this command into the script-file editor, you can then select the entire line – you can do this by left-clicking at the beginning of the line and then moving to the end and releasing the left mouse button – and then click on the "Run" button at the right side of the top of the script-file editor window that has a green arrow pointing from a box toward the word "Run." Clicking on this button tells the program to execute the selected line or lines of code.[3] In Figure 1.4, we show what this will look like right before you click on "Run." Once you have done this correctly, your RStudio should look like Figure 1.5. And, to further check that you did this correctly, when you left-click on the "Environment" tab in the upper-right corner window, you should see the same contents in that window as what you see in Figure 1.5.

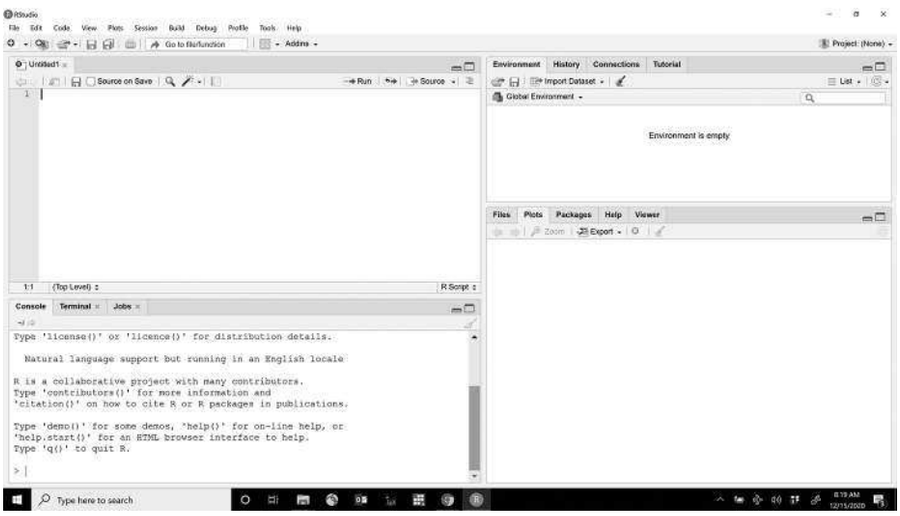

Figure 1.3 Script editor window open

[3] This also works for selecting multiple lines of code, as we do later in this file. It is worth noting two additional things. First, you don't have to select the entire line of code to submit it as long as your cursor is located somewhere in the line that you wish to run. And second, instead of clicking the "Run" button, you can run code by hitting the Control and Enter keys at the same time (Ctrl + Enter).

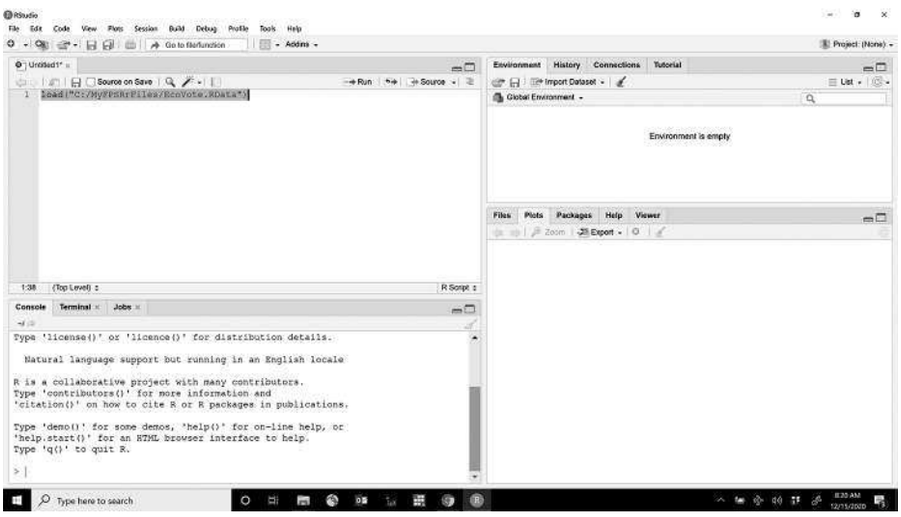

Figure 1.4 Preparing to execute a command from the script-file editor

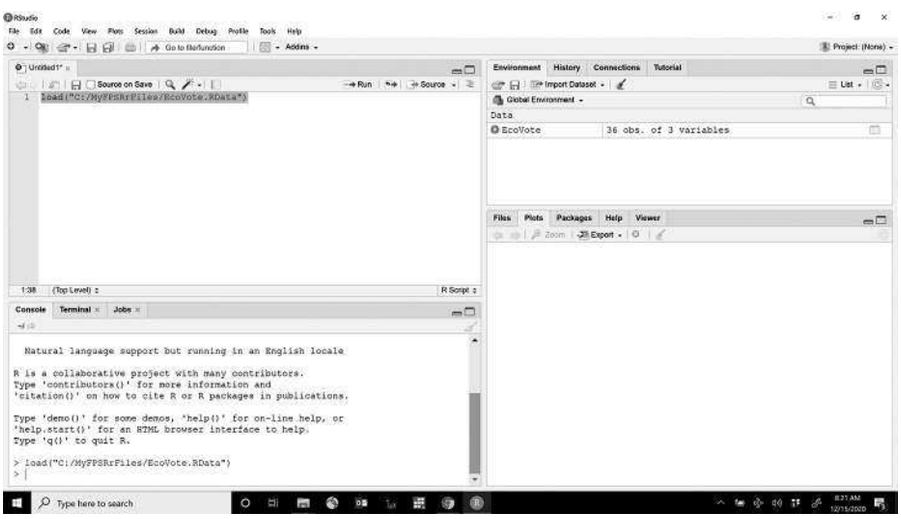

Figure 1.5 RStudio after having successfully run a command from the script-file editor

1.3.3 Initially Examining Data in R

Now that we have shown you two different ways to get a data set into R, we want you to take a look at the data that you have loaded into the program. These data are from a famous study of economic voting conducted by Ray Fair (Fair 1978). They contain values of economic growth and incumbent party vote from US presidential elections between 1876 and 2016. To get an initial look at these data, click on "EcoVote" in the Environment window in the upper-right corner of RStudio. Once you have done this, your RStudio should look like Figure 1.6. Each column in this spreadsheet contains

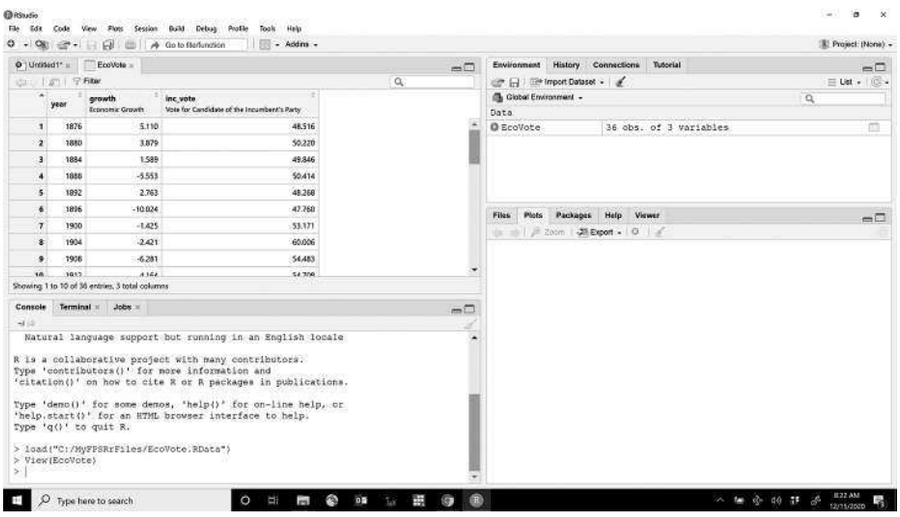

Figure 1.6 Initially examining data in RStudio

values for a single variable and each row contains data from a single election. You are now ready to proceed to the end-of-chapter exercises.

1.3.4 Adding Notes to Script Files and Saving Them

The main advantage of script files is that they can be saved. This allows users to go back later and see exactly what they did. Once you start writing longer script files, it is a good idea to make notes in them about what you are doing and why you are doing it. We can make notes in script files by starting the line on which we want to write notes with a hash symbol (#). When you ask RStudio to run a command line that starts with #, R reads that symbol as "ignore everything to the right of this symbol on this line of code" and, appropriately, does nothing. This is known as "commenting out a line" of code. This is helpful when we want to submit an entire script file to R at one time but still have notes about what we did. The programs that are available on the webpage for our book (`www.cambridge.org/fpsr`) contain commented-out lines that explain what is going on in them.[4]

In Figure 1.7 we have clicked back on the R script that we created to load our data set and saved it as a file named "LoadData.R." Notice that we have added a comment at the top of the script file, preceded by a # that tells us what this script is. It is normally a good idea to add additional lines of comments that identify the person who wrote the code and the date on which the script file was created.

[4] You may also want to use the # symbol to add a note to part of a line of code. In this case, you would just put whatever note you want to add to the right of the # and anything that you don't wish to comment out to the left of the #.

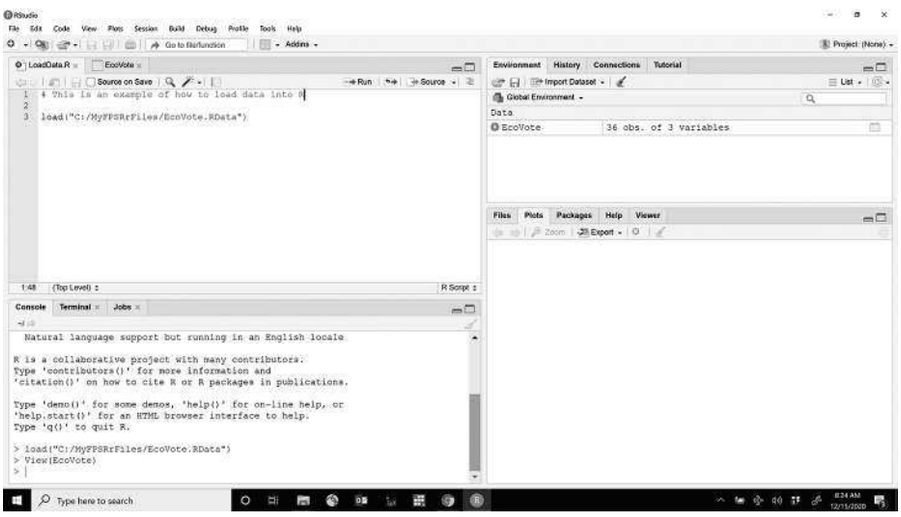

Figure 1.7 Script with comment line saved

1.4 EXERCISES

1. Go through all of the steps described above. Once you have the data set open (so that your computer looks like Figure 1.6), do the following:
   (a) Look at the values in the column labeled "growth." This is Fair's measure of percentage change in real GDP per capita. Do the following:
      i. Identify the year with the highest value for this variable.
      ii. Identify the year with the lowest value for this variable.
      iii. What does it mean if this variable goes up by 1?
   (b) Look at the values in the column labeled "inc_vote." This is Fair's measure of the percentage of major party votes cast for the party of the president at the time of the election. Now do the following:
      i. Identify the year with the highest value for this variable.
      ii. Identify the year with the lowest value for this variable.
      iii. What does it mean if this variable goes up by 1?

2 THE ART OF THEORY BUILDING

2.1 OVERVIEW

One of our emphases in the book has been on producing new causal theories, and then evaluating whether or not those theories are supported by evidence. In this chapter, we describe how to explore sources of variation – both across space, and across time – to get you started thinking about new explanations for interesting phenomena. We also help you explore how new theories can be built upon the existing work in the literature. Before we can explore data, however, we need to do a little bit of work in R to get access to the commands that we will use.

2.2 R PACKAGES

One of the great things about R is that the user community is constantly producing new tools for everyone using the program. These new tools are made available in the form of programming modules called "packages." Once you have R and RStudio installed on your computer, you can add additional packages in at least the following three different ways:

1. By left-clicking on the "Tools" located in the upper-left corner of RStudio, and then selecting "Install Packages." This will launch a pop-up window in which you should type the name of the package that you wish to install and then left-click the "Install" button.
2. By left-clicking on the word "Packages" at the top of the bottom-right window. You then see a list of packages that you already have installed on your computer. From there, you can select the search window (next to a magnifying glass) to allow you to search for a particular package and install it.
3. By running the following command (either from the console or from a script file): `install.packages("PackageName")`

At the time that we are preparing this guide, there are over 10,000 packages available for R. In order to do all of the things that we discuss in this book, you will need the following packages:

- tidyverse
- gmodels
- labelled
- psych
- lm.beta
- car

In Figure 2.1 we can see a script written to install all of these packages. Note that, depending on the speed of your computer and your connection to the Internet, this script might take some time to run. You can tell that it has finished when you see a blue > symbol at the bottom of the console window like what we see in Figure 2.1.

It is worth noting that every time you start an RStudio session, you will need to tell the program which packages you wish to use. This is done with a "library" command. For instance,

```
library(tidyverse)
```

tells R to load the package "tidyverse," which is a package that provides access to a large set of commands that all follow a similar structure.[1]

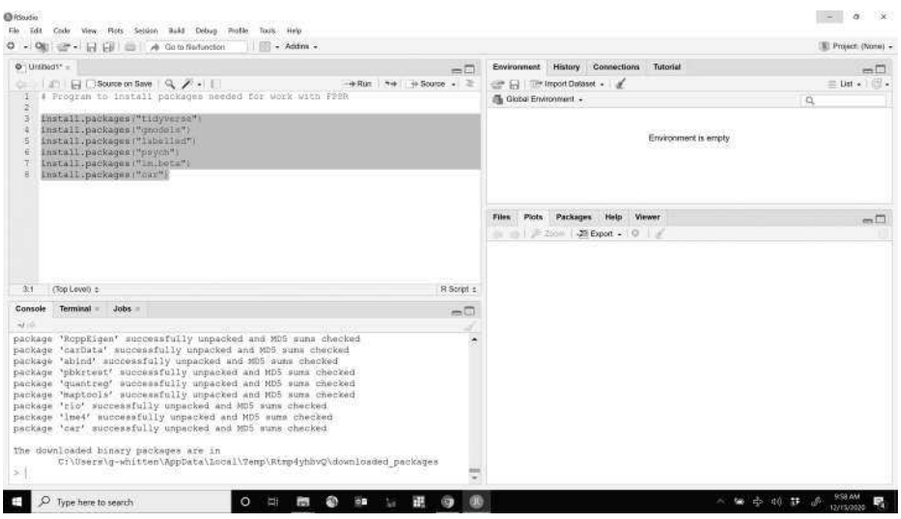

Figure 2.1 Packages installed by script

2.3 EXAMINING VARIATION ACROSS TIME AND ACROSS SPACE

As we discuss in Section 2.3 of *FPSR*, one way to develop ideas about causal theories is to identify interesting variation. In that section, we discuss examining two types of

[1] Tidyverse is actually a collection of packages. For more information on this, visit `www.tidyverse.org`