Using the Statistical Software Package R

Simon Heuberger* Ryan T. Moore†

Friday 2nd September, 2022 at 11:52

1 Background

Political scientists use statistical software packages to conduct quantitative analyses. One such software package is R. R is free and open-source.

Statistical analyses can be done using R alone, but it is more user-friendly to use an R platform, or IDE (*integrated development environment*). One such platform is RStudio.

R and RStudio are separate pieces of software, but we will use both to conduct analyses.

We will only open RStudio. R will run automatically from within RStudio.

There are three ways to use RStudio:

- Download and install it on your computer
- Access it through an on-campus computer in one of the computer labs
- Access it via AU's Virtual Computing Lab (VCL)

We recommend that you download and install RStudio on your computer. If you experience any problems or have any questions setting everything up, please contact us, or ask the class community for help on Slack.

If you don't have a computer, we recommend that you use one of the computers in an on-campus lab, where R and RStudio are pre-installed. A list of on-campus computer labs can be found here.

It is possible to use the Virtual Computing Lab, but the application of RStudio through the VCL can sometimes be tricky and unstable.

*Department of Government, American University, `sh6943a` (at) `american` (dot) `edu`

†Department of Government, American University, Ward Circle Building 226, 4400 Massachusetts Avenue NW, Washington DC 20016-8130. tel: 202.885.6470; fax: 202.885.2967; `rtm` (at) `american` (dot) `edu`; `http://ryantmoore.org`.

2 Download and Installation On Your Computer

R and RStudio need to be downloaded separately. Generally, we want to keep our versions of R and RStudio up-to-date.

R

If you are a Mac user, go to the Mac download website. Left-click on the installer "R-3.4.3.pkg" and download it to a folder of your choice. If a newer version of R has become available for download since we wrote this guide (e.g. "R-3.4.4"), then download the newer version.

If you are a Windows user, go to the Windows download website. Left-click on the installer "Download R 3.4.3 for Windows" and download it to a folder of your choice. If a newer version of R has become available for download since we wrote this guide (e.g. "R-3.4.4"), then download the newer version.

Once the download is complete, open the downloaded file, and follow the installation instructions. You can install R in a folder of your choice.

RStudio

Go to the RStudio website. At the bottom of the page are several installers.

If you are a Mac user, left-click on the installer "RStudio 1.1.414 - Mac OS X 10.6+ (64-bit)" and download it to a folder of your choice. If a newer version of RStudio has become available for download since we wrote this guide (e.g. "RStudio 1.1.415"), then download the newer version.

If you are a Windows user, left-click on the installer "RStudio 1.1.414 - Windows Vista/7/8/10" and download it to a folder of your choice. If a newer version of RStudio has become available for download since we wrote this guide (e.g. "RStudio 1.1.415"), then download the newer version.

Once the download is complete, open the downloaded file, and follow the installation instructions. You can install RStudio in a folder of your choice.

3 Access Through On-Campus Computers

All computers in on-campus labs have R and RStudio preinstalled. You do not need to install R or RStudio manually.

On-campus computers run the Windows operating system.

Saving Work on On-Campus Computers

It is best to save files to an online cloud storage system (such as Google Drive or Dropbox) or to an external device (such as a USB stick). If you do not have access to a cloud storage system, you can create a free Dropbox account here. **Never save files to the Desktop. Any files saved on the Desktop will be lost after logging off.**

4 Access Via AU's Virtual Computing Lab (VCL)

The Virtual Computing Lab (VCL) allows you to remotely access software, such as R and RStudio, from your computer through the university network. Through the VCL, you have 24/7 access on and off campus, from anywhere. All you need is a reliable internet connection.

You do not need to install R or RStudio manually. Both are preinstalled through the VCL.

The VCL runs on PCs and Macs.

There are two versions of the VCL: a desktop client that you need to install on your computer, and a browser version that does not require installation. Both versions offer the same access.

Browser Client

Detailed instructions on how to access the browser version of the VCL can be found here.

You can exit the client by simply logging off after you have saved your work.

Desktop Client

A detailed download and installation guide for Mac users can be found here.

A detailed download and installation guide for Windows users can be found here.

On both Windows and Mac, a Windows interface will open upon VCL launch.

You can exit the client by simply logging off after you have saved your work.

Saving Work in the VCL

Since the VCL runs remotely, access to your local hard drive is restricted.

In the Desktop Client, files can be saved to an online cloud storage system (such as Google Drive or Dropbox) or to an external device (such as a USB stick), or to your local hard drive.

In the Browser Client, all files must be saved to an online cloud storage system. In the Browser Client, **you do not have access to your local hard drive or USB storage**. If you do not have access to a cloud storage system, you can create a free Dropbox account here.

In order to access your local hard drive and USB storage in the Desktop client, simply follow the instructions in the installation guide links provided above. After this set-up, simply click on the "Computer" icon in the VCL interface and your local hard drive will be listed. **Only save files to your local hard drive, never the VCL Desktop. Any files saved on the VCL Desktop will be lost after logging off.**

5 The Basics of RStudio

The Basic Workflow

Whenever you start a new analysis, you'll go through the following steps:

1. Open RStudio
2. Create a new file for your code (a `.R` file)
3. Write your analysis code in the `.R` file
4. Execute the code to perform your analysis
5. Save the `.R` file
6. (Repeat steps 3 to 5 many times)

We recommend you restart R often during development. This ensures that there are not extra objects in your workspace or packages loaded that you've forgotten about. If you use RStudio, use the menu item Session - Restart R or the associated keyboard shortcut Ctrl+Shift+F10 (Windows and Linux) or Command+Shift+0 (macOS).

Opening RStudio

On Your Computer

Open the RStudio application in your chosen installation folder.

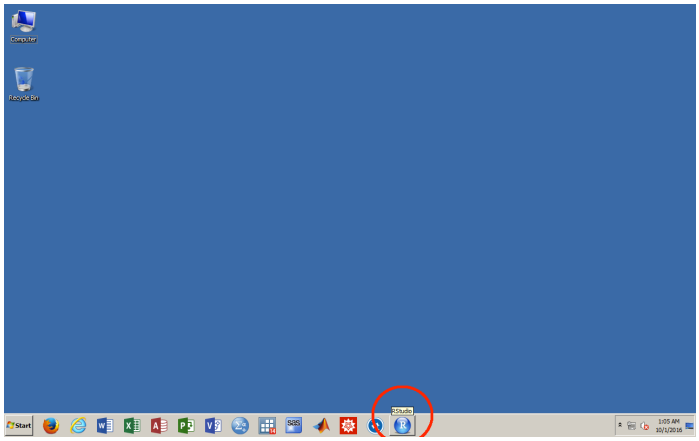
On an On-Campus Computer

Log on with your AU username and password and click on the "RStudio" button.

Via the VCL

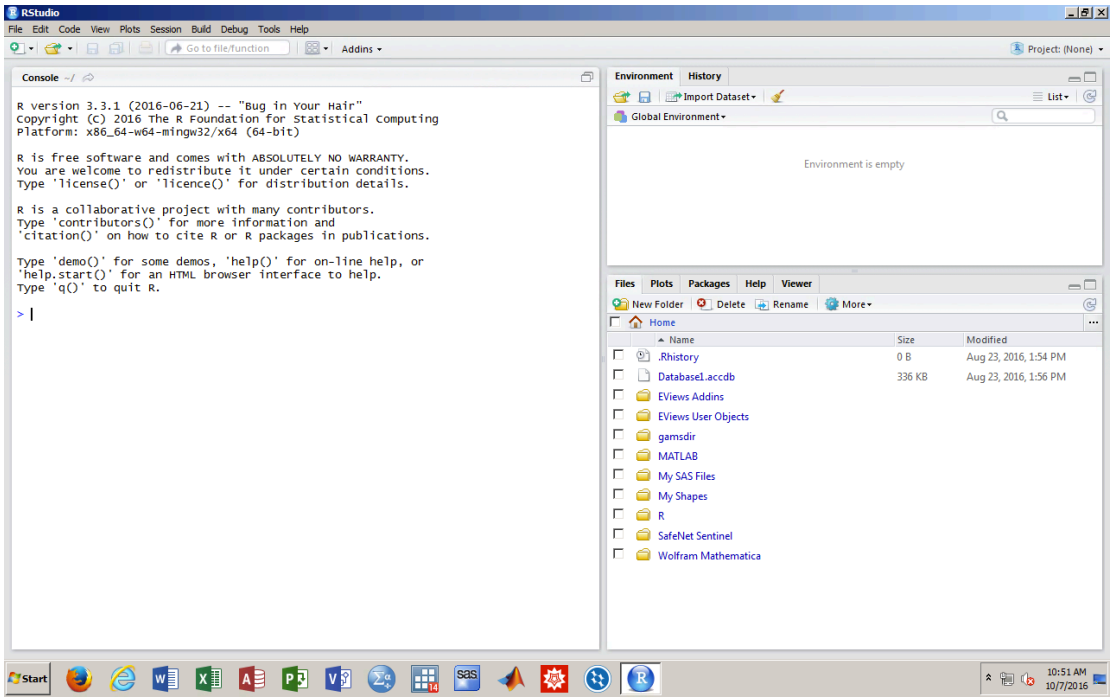
With the Desktop Client, open the VMWare Horizon Client in your chosen installation folder. You will be logged onto a Windows interface. Click on the "RStudio" button.

If you wish to use the Browser Client, simply follow the instructions above (and here). You will be logged onto a Windows interface. Click on the "RStudio" button.

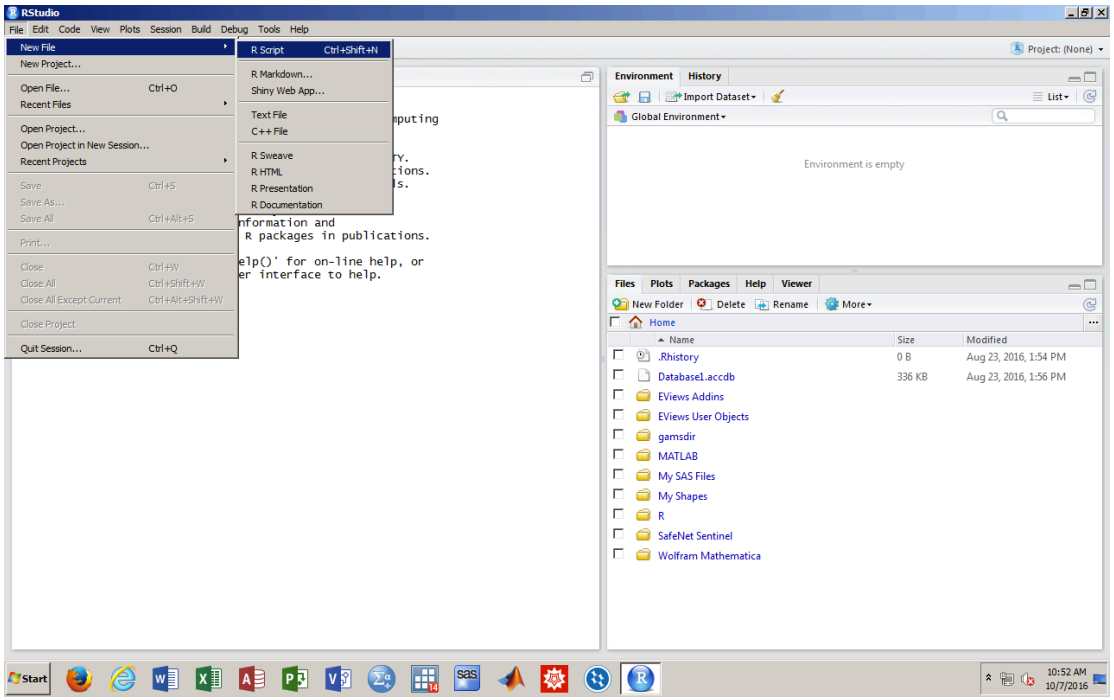

Computer
Recycle Bin
Start
RStudio
1:05 AM
10/1/2016

Creating a `.R` File

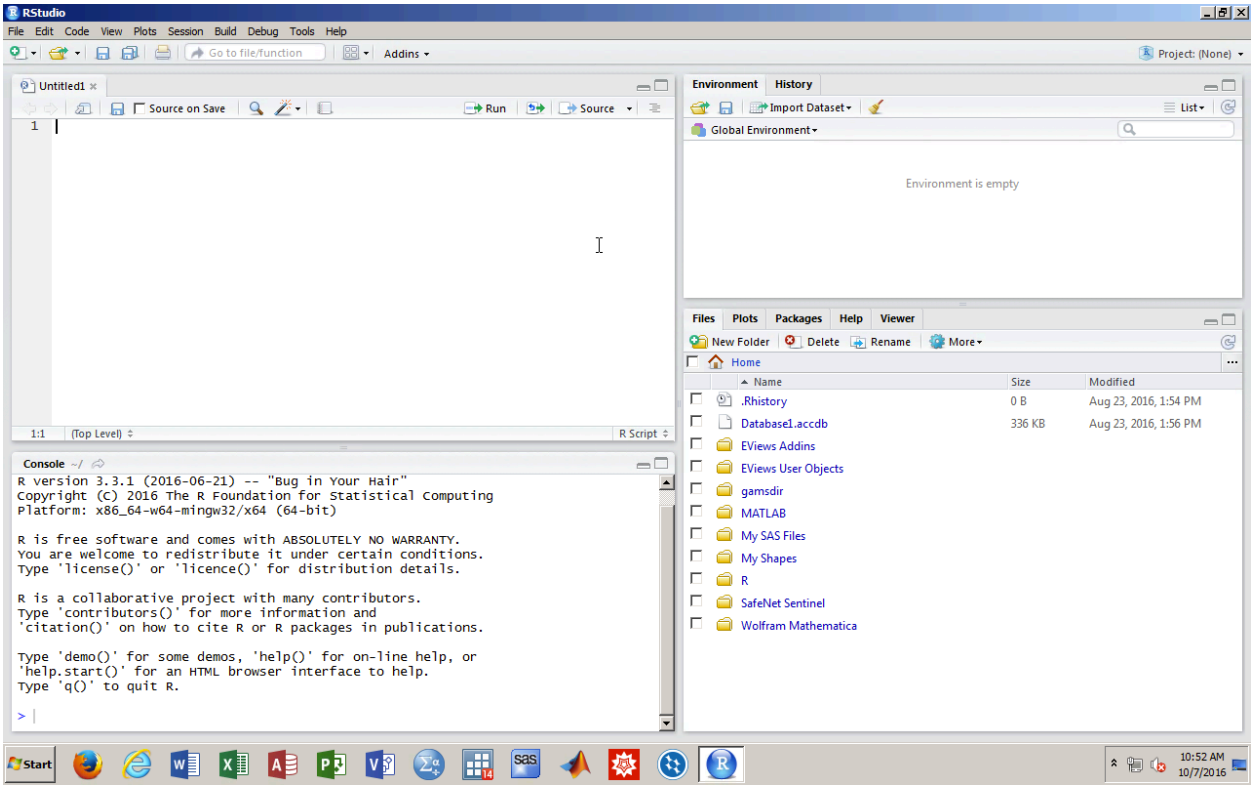
Below is the default RStudio interface.



The first thing we do is open a new R Script.



Then the RStudio interface will look like this:



The top left window is the new **R Script** file. This is where you type and save all of your code. The saved file is called a `.R` file.

The bottom left window is the **Console**. This is where all of your code will be executed. You can ignore the text that comes preloaded in the console. It is just software specifics.

The top right window shows you the **Environment** and the **History** tabs. **Environment** lists all the data you have loaded in your current session. **History** lists all of the code commands you have used in your current session.

The bottom right window shows you the **Files**, **Plots**, **Packages**, **Help**, and **Viewer** tabs. **Files** lists all the files and folders that are in your default workspace (how to set the workspace is shown below). **Plots** lists all the plots you have created in your current session (how to create a plot is shown below). **Packages** lists all the R packages you have loaded (how to load packages is shown below). **Help** shows on information on any R package if you request it. **Viewer** can be used to show local web content (you will most likely not use this).

How Does RStudio Work?

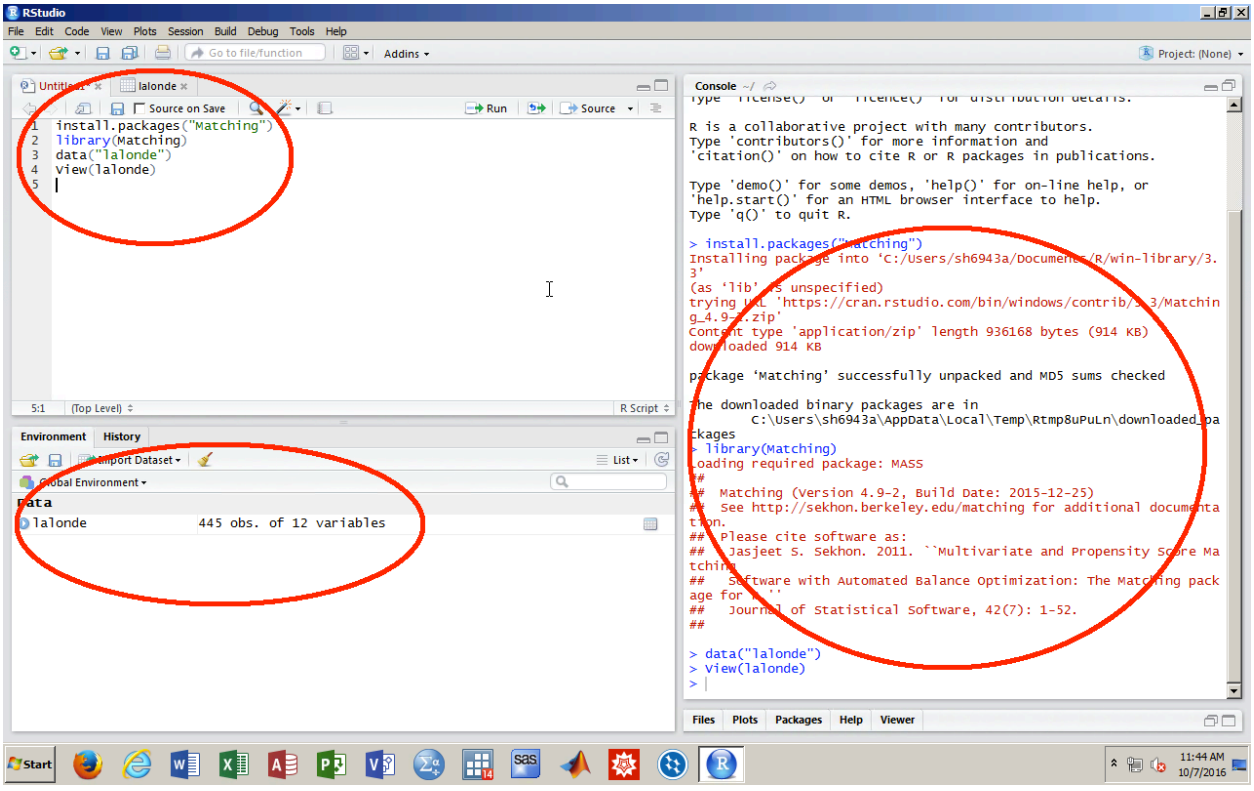
R is code-driven statistical software. The tabs in the task bar are kept to a minimum, which means that you will enter all code and conduct all analyses by typing on your keyboard.

You write all code in the `.R` file (i.e. the R Script file you created above) and save this file frequently. This allows you to easily recreate/resume your analyses at a later time.

The code you write is executed in the Console. The output of your code will also appear in the Console. Each new line in the Console is indicated by a blue > sign.

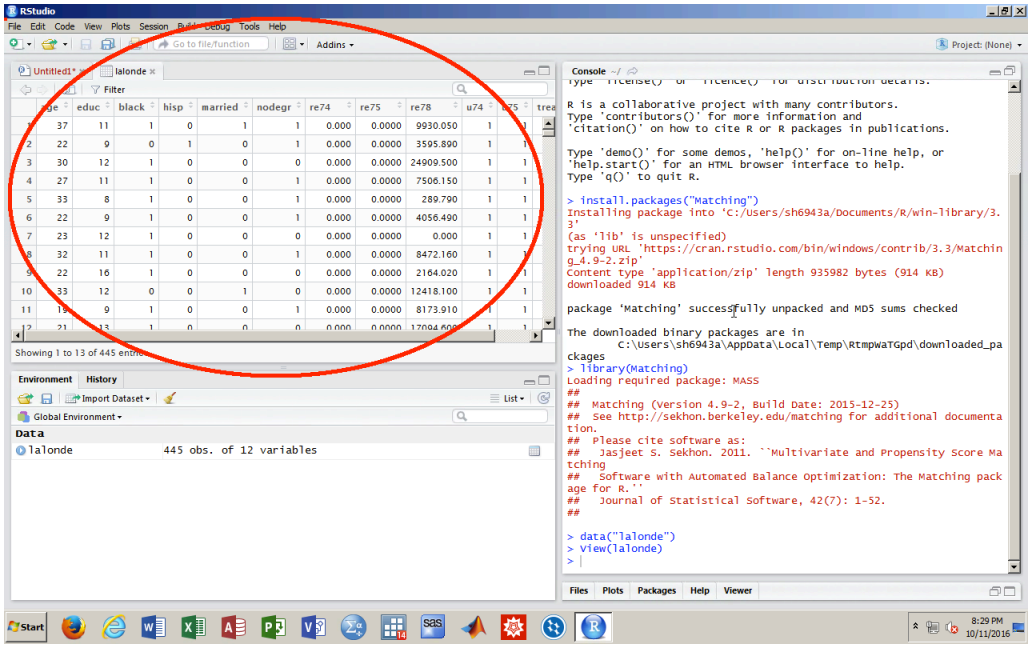
What is Code?

Code consists of commands. There are thousands of commands in R. Below is an example of code. It consists of the commands to install the add-on R package `Matching`, load the library for this package, load a data set called `lalonde` included in this package, and look at this data set. (*Note: We have rearranged the RStudio windows here to provide a better picture. You can do this via View/Panes/Pane Layout. This is not necessary to run the commands.*)

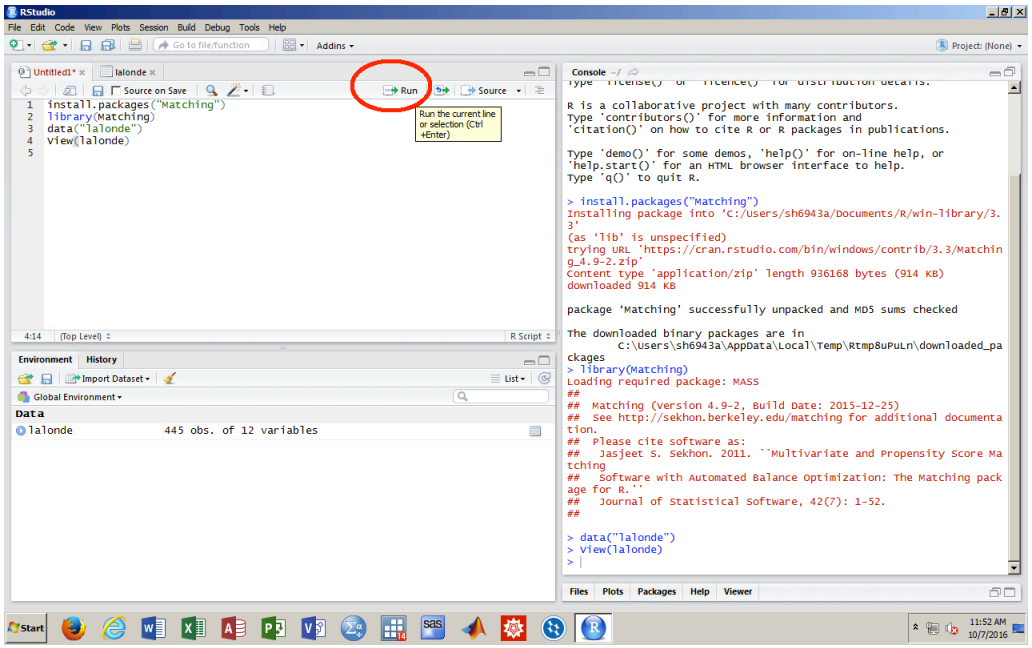


We have typed the commands into the `.R` file (top left window). These commands have been executed in the Console (right window). After loading the `lalonde` data set, it appears under the Environment tab (bottom left window), listing the number of variables and observations contained in the this data set. After executing `View(lalonde)`, a new tab opens up which shows the data set in rectangular form (see below). Note that we did not

type anything into the Console. All the commands were typed into the `.R` file and executed in the Console. That way, we have a record in the `.R` file of our analysis.



To execute commands, you can click on the "Run" button, copy and paste them into the Console, or use a keyboard shortcut.

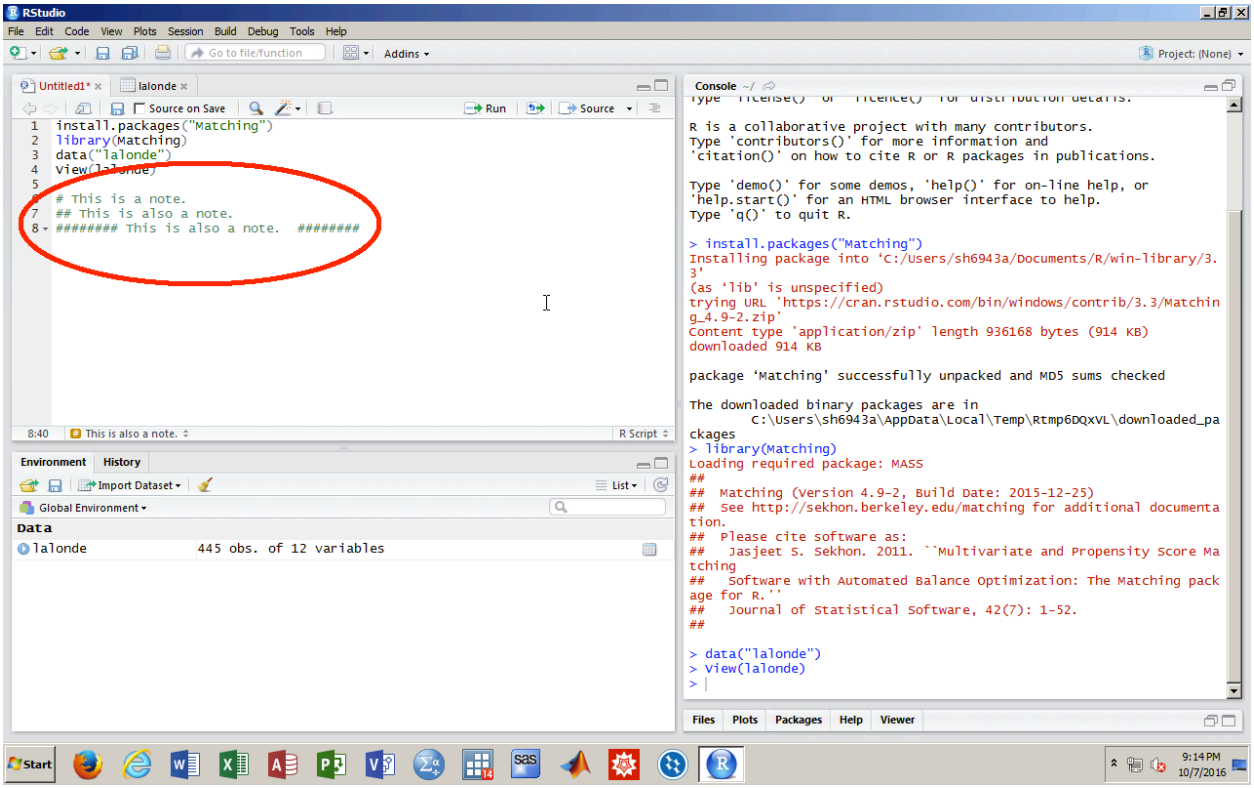


Some Useful Keyboard Shortcuts

	Windows	Mac
Open new R Script	Ctrl+Shift+N	Command+Shift+N
Execute Command	Ctrl+Enter	Command+Enter
Select code sections	Ctrl+Arrow	Command+Arrow
Undo	Ctrl+Z	Command+Z
Cut	Ctrl+X	Command+X
Copy	Ctrl+C	Command+C
Paste	Ctrl+V	Command+V
Select All	Ctrl+A	Command+A
Run entire current document	Ctrl+Alt+R	Command+Option+R
Run from document beginning to current line	Ctrl+Alt+B	Command+Option+B
Run from current line to document end	Ctrl+Alt+E	Command+Option+E
Save active document	Ctrl+S	Command+S
Clear console	Ctrl+L	Ctrl+L
Restart R Session	Ctrl+Shift+F10	Command+Shift+0

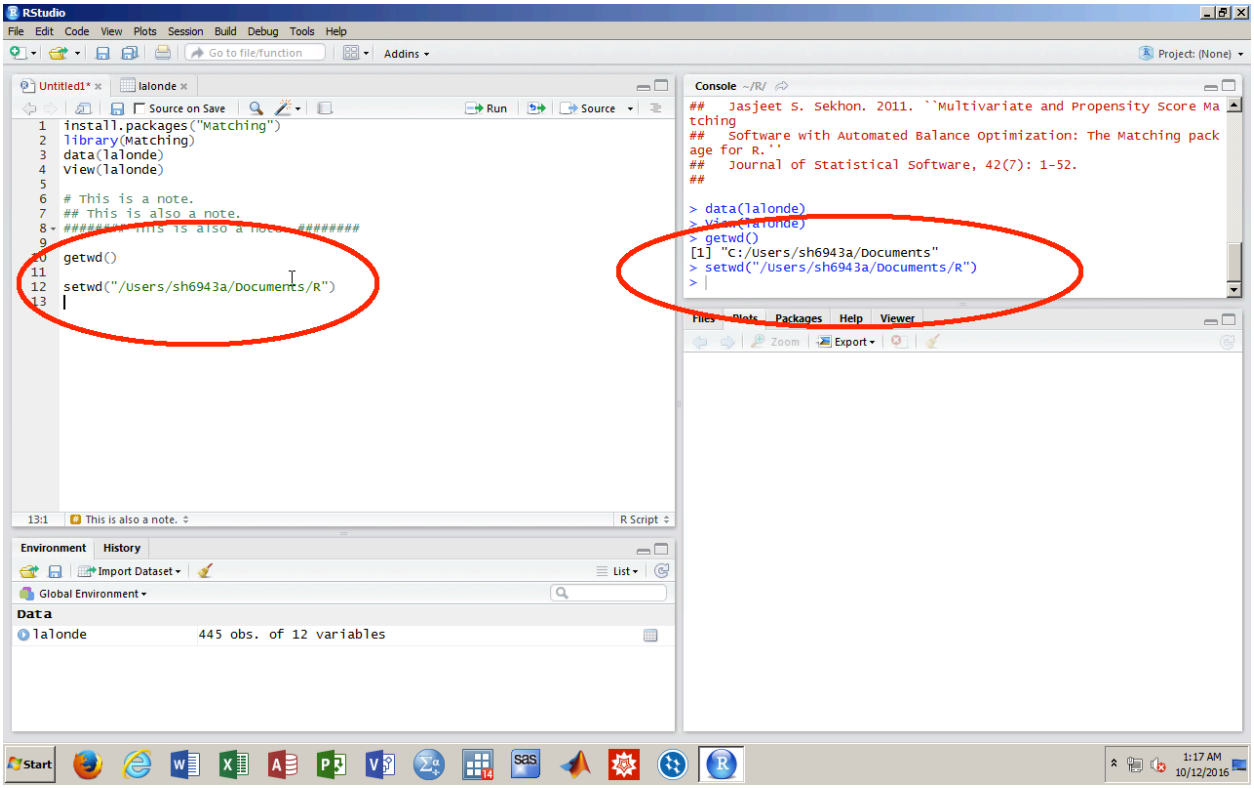
Organization

Keep your `.R` file tidy. This includes writing notes in your code (for us, for collaborators, and most importantly, for your future self). This allows you and anyone looking at your code to easily figure out what you are doing. Notes can be inserted by using the # symbol. Notes are shown in green font and are not executed by RStudio. You can use as many hashtags as you wish.



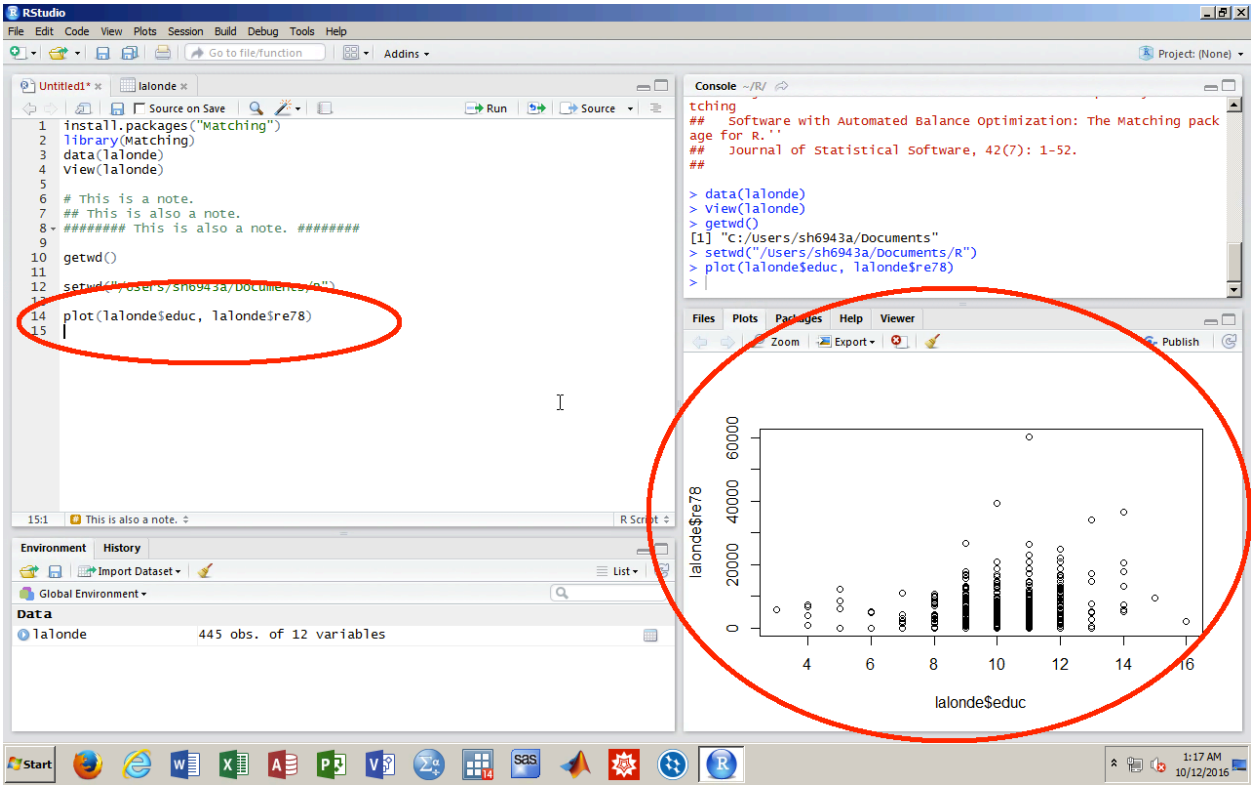
Setting the Working Directory

RStudio is always pointed at a folder on the computer you are working on, be that on your local machine or via the VCL. This folder is called the **Working Directory**. You can find out which directory RStudio is pointed at by running the `getwd()` (get working directory) function. To change your working directory, use the `setwd()` function and specify the path to the desired folder. Note that the desired destination folder needs to already exist in order to set it as the working directory. It is good practice to start your R session by setting the working directory (or at least checking it). That way, you know where any files or figures you save are located.



Creating And Saving A Plot

Creating a simple plot is easy. In the example below, we plot the variable `educ`, which shows years of education, on the x-axis, and the variable `re78`, which shows income earnings in 1978, on the y-axis. Both variables are in the LaLonde data set used above. Any plots you create will appear in the Plots tab.



Saving a plot can either be done via Plots/Save as PDF or with this code:

```
> pdf("~/Folder/file_name.pdf")
> plot(lalonde$educ, lalonde$re78)
> dev.off()
```

The first command specifies the location where you want to store ("Folder") and what you want to name the resulting PDF ("file_name"). This starts the creation of the PDF.

The second command follows the structure of plot(x,y). This means the variable before the comma (`educ`) will be plotted on the x-axis, and the variable after the comma (`re78`) will be plotted on the y-axis. We need to include `lalonde$` with the variable name, otherwise RStudio does not know what data set we are referring to (we could have multiple data sets loaded at the same time). As you recall, `lalonde` is just the name of our data set in this example, and `educ` and `re78` are the variable names in this data set. The `$` simply follows the R command structure of `data_set_name$variable_name`.

The third command ends the creation of the PDF. `dev.off()` tells RStudio that we are finished creating the PDF. Without `dev.off()`, RStudio will try to incorporate any following lines of code into the PDF as well (which will lead to errors).

Saving a `.R` File

Simply click on the "Save" button and choose a folder. Alternatively, you can use the saving keyboard shortcut shown above, or use this command:

```
> save(file="d:/file_name.R")
```

`.R` vs. `.RData` Files

`.R` files save the content you typed into an R Script. When you open a saved `.R` file, it will contain all of your previously written code, but it will not show any of the analyses you have conducted or plots you have created. You will need to execute all the commands in the R Script to conduct/create those again. We recommend saving your work as a `.R` file and re-executing commands when you open it.

`.RData` files are usually used to store data in a native R format, but also can store your R history. By default, RStudio will ask you whether you want to save the `.RData` file upon exiting. We recommend you do not. This setting can be changed if you have installed RStudio on your own computer. It cannot be changed if you use RStudio via the VCL.

Installing and Loading Packages

You can install and load any R packages with the following commands:

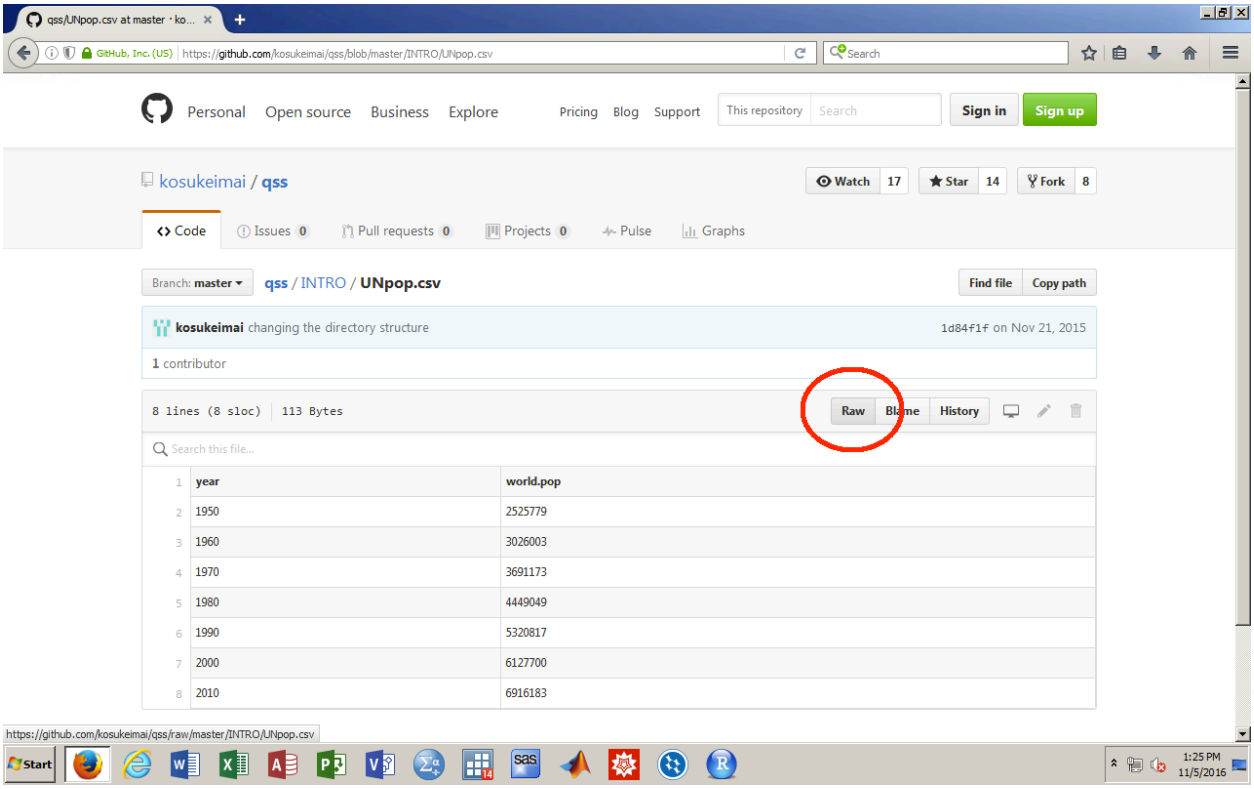
```
> install.packages("package_name")
> library(package_name)
```

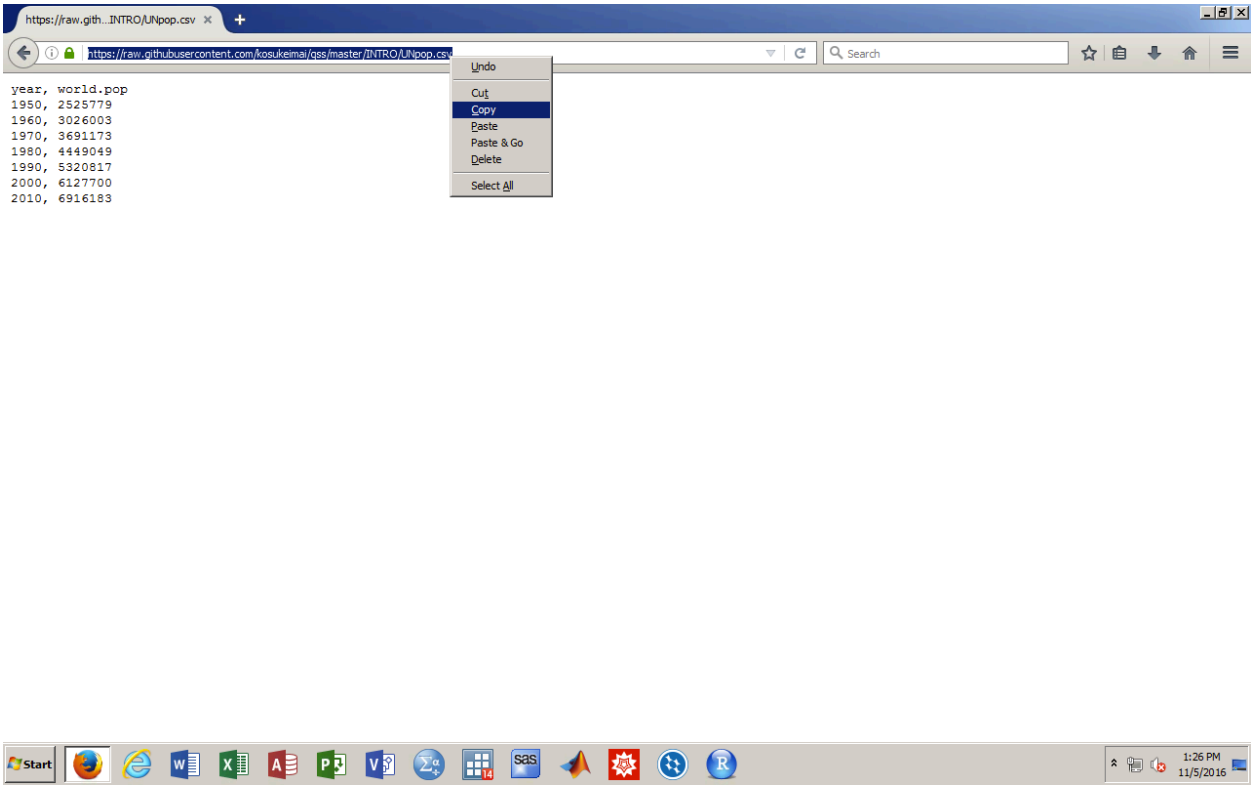
Note that package names are case sensitive and that quotation marks are required for the first command, but need to be omitted for the second. If you use R on your own computer, installing is only necessary once, but loading needs to be done each time for every new R session. If you use R via the VCL, both installing and loading are necessary for every new R session.

Loading Data From the Imai Book

This example walks you through the loading of the `UNpop` data (page 22) into RStudio. You can download the data and then read it into R, but it is much better to read it straight from the web. That way, your code is much easier to replicate later on, and you will always have the most up-to-date data available. The downside, however, is that it requires an Internet connection.

First, go to Imai's folder on GitHub. Then select the appropriate subfolder (in our case the INTRO folder). Then select the data you are interested in (here the UNpop.csv file). Then click on the "Raw" button and copy the resulting web address.


qss/UNpop.csv at master · ko...
https://github.com/kosukeimai/qss/blob/master/INTRO/UNpop.csv
Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up
kosukeimai / qss
Watch 17 Star 14 Fork 8
Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs
Branch: master qss / INTRO / UNpop.csv
Find file Copy path
kosukeimai changing the directory structure
1d84f1f on Nov 21, 2015
1 contributor
8 lines (8 sloc) 113 Bytes
Raw Blame History
Search this file...
year world.pop
1950 2525779
1960 3026003
1970 3691173
1980 4449049
1990 5320817
2000 6127700
2010 6916183
https://github.com/kosukeimai/qss/raw/master/INTRO/UNpop.csv
Start
1:25 PM
11/5/2016


https://raw.gith...INTRO/UNpop.csv
https://raw.githubusercontent.com/kosukeimai/qss/master/INTRO/UNpop.csv
Search
year, world.pop
1950, 2525779
1960, 3026003
1970, 3691173
1980, 4449049
1990, 5320817
2000, 6127700
2010, 6916183
Undo
Cut
Copy
Paste
Paste & Go
Delete
Select All
Start
1:26 PM
11/5/2016

Now open RStudio, use the `read.csv()` command, and paste the web address into the parentheses. Note that you need to add quotation marks around the link. The resulting command looks like this:

```
> UNpop <- read.csv("https://raw.githubusercontent.com/kosukeimai/qss/master/
   INTRO/UNpop.csv")
```

This will read the data into an R data frame labelled "UNpop". Note that you can name the R data frame in any way you please, but descriptive names are better. You could name this data frame "df" or "my_data", or "X" by just changing the text on the left side of the assignment arrow (`<-`).

To load any other data set, simply repeat this process and change the link and the name in the `read.csv()` command.

Software Maintenance

On Your Computer

RStudio will save all changes you make to the settings, such as changing the pane layout. Periodically, you may want to update both R and RStudio. This is done via Help/Check for Updates in RStudio and R/Check For R Updates in your installation of R.

On an On-Campus Computer / Via the VCL

If you use RStudio via in an on-campus computer or via the VCL, RStudio will always have the default settings upon startup. Any changes, such as changing the pane layout or setting the working directory, will need be to re-done each time upon startup. There is no need to update R or RStudio.

Submitting Problem Sets and Labs

You will submit your solutions to the problem sets in printed-out hard copy. You may also submit them to the course Blackboard site, under Content/Assignments/Problem Sets. For the in-class labs, you will submit your solutions to the course Blackboard site, under Content/Assignments/Labs. Both online submissions need to be in PDF format.

In order to create these PDF documents, you will use a template we have created for you. This template is called a `.Rmd` file, and there are two versions on my teaching page. These files are allow the combination of text and R code into one compiled document. A `.Rmd` file is more complicated than a `.R` file; below we walk you through all the sections of the `.Rmd` template.

Note: In order to successfully compile the `.Rmd` template on your local machine, you need to install a couple R packages. You may not need to run these commands through the VCL as the packages may be pre-installed.

Here are the steps for compiling directly to PDF:

1. `install.packages("rmarkdown")`
2. `install.packages("tinytex")`

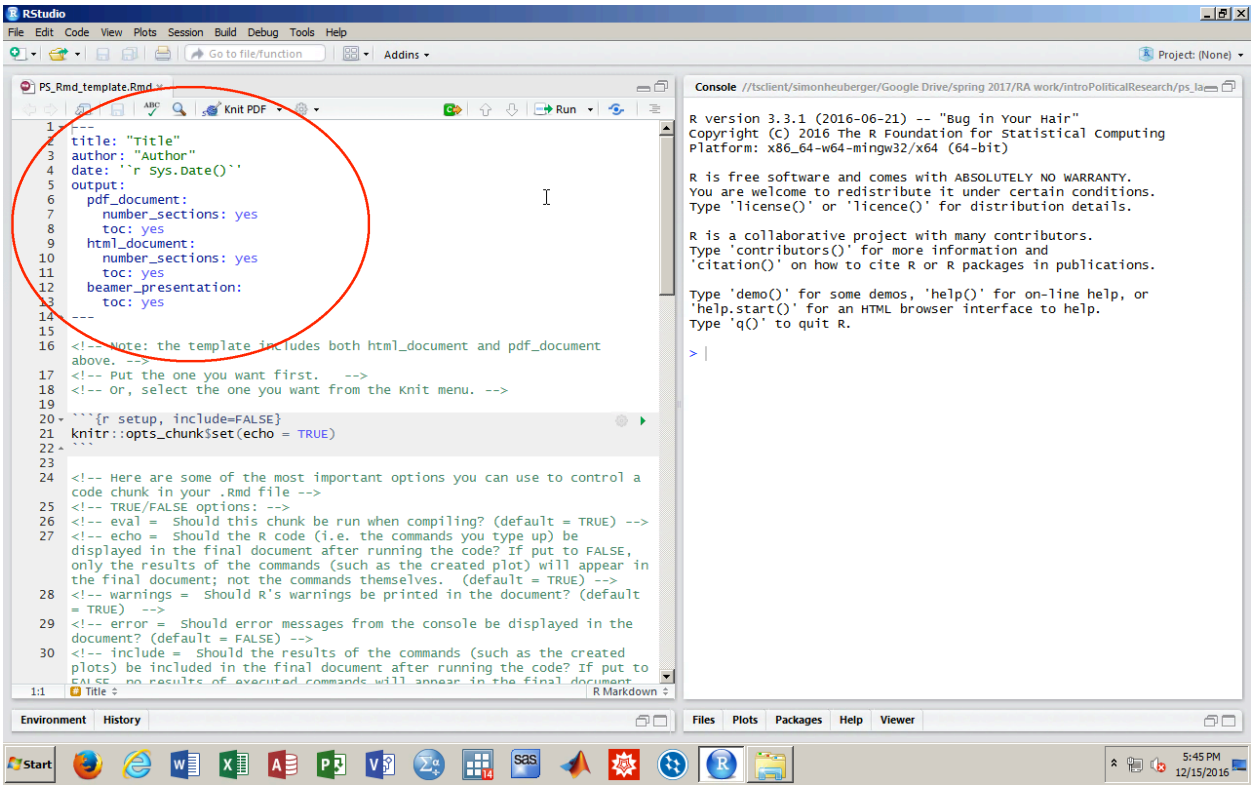
3. `library(rmarkdown)`
4. `tinytex::install_tinytex()`

   This will take a couple minutes. Pay attention to any warnings/notices that arise, including those at the R prompt.
5. Restart RStudio (not just R, but the full RStudio)
6. To compile your file, use the "Knit to PDF" option under the RStudio Knit button,
7. Open the created `my_ps_solutions.pdf` in whatever you use to view PDFs.

Note that for 6 above, you can, alternatively,

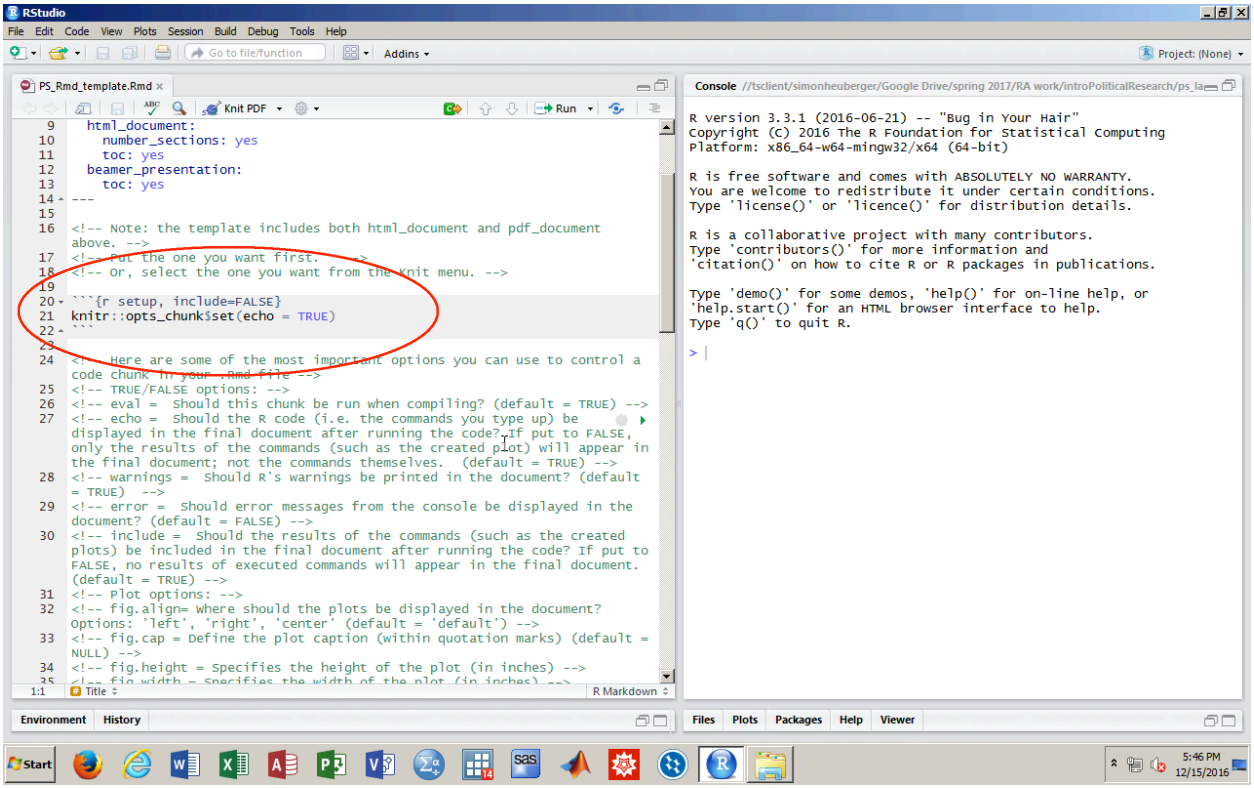
- Use the Knit button directly if your `.Rmd` file says `pdf_document` in the header instead of `html_document` (or if `pdf_document` is listed first), or
- Use `rmarkdown::render("my_ps_solutions.Rmd", "pdf_document")`
- Use `rmarkdown::render("my_ps_solutions.Rmd")` if your `.Rmd` file says `pdf_document` in the header instead of (or above) `html_document`.

6 A `.Rmd` File, and Some Examples

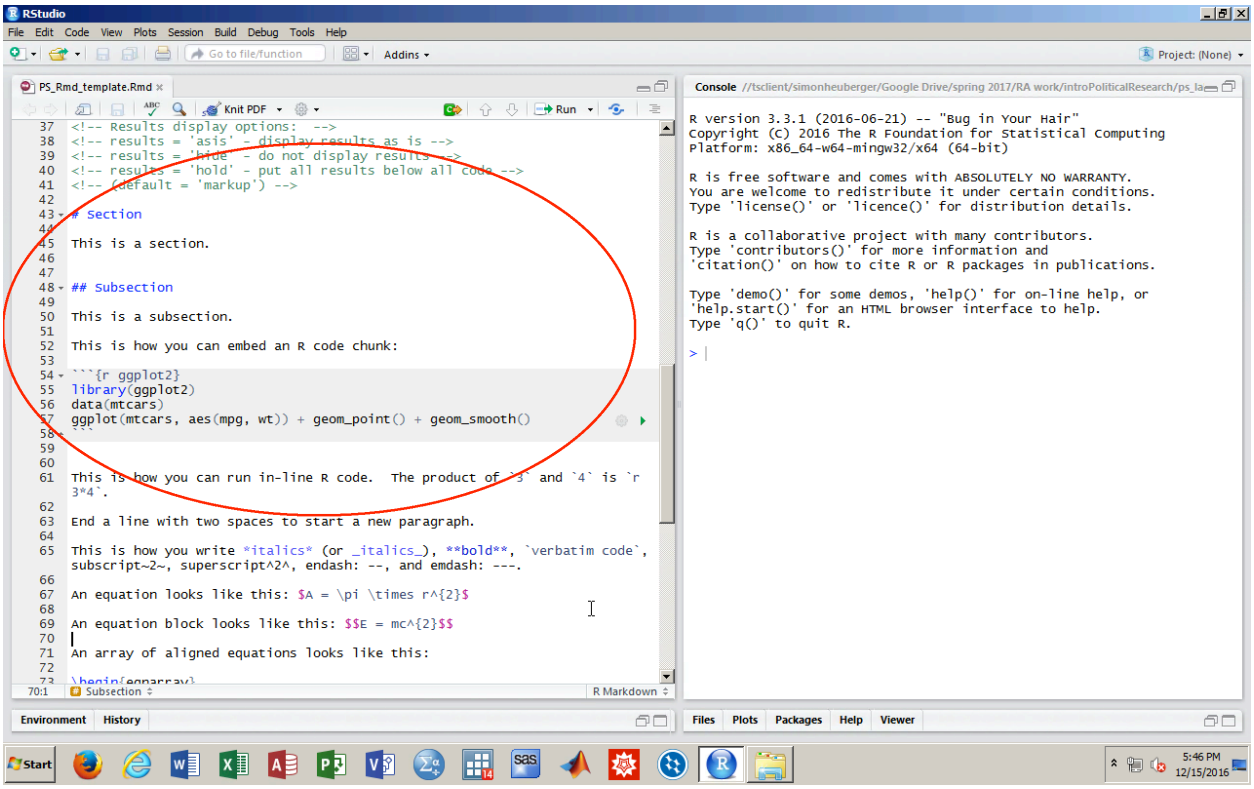


This section is called the preamble. Here you set the title, author, and date of the document. You also set the output document type. The document types available are HTML, PDF, and beamer. Beamer is used to create PowerPoint-style presentation slides. The output type that is listed first is the one used for creation. In our case, this is PDF. The command "number_sections" should be set to "yes". It creates numbered sections in the document.

If you are working on an on-campus computer or via the VCL, you may not be able to create a PDF directly. You need to create an HTML document first (by listing "html_document" as the output type) and then print the ensuing HTML document as a PDF. If you are working on your local machine, you can compile directly to PDF.

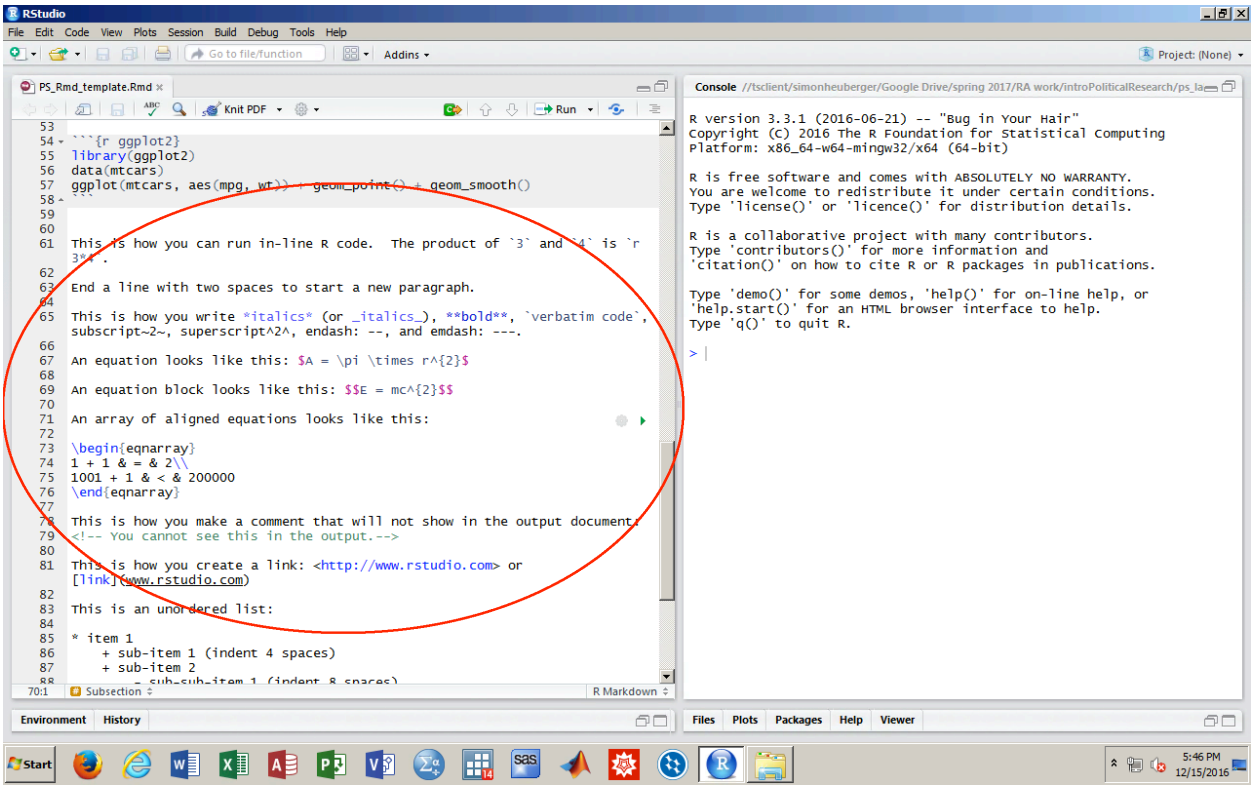


This section is called a code *chunk*. Here you specify how R should deal with the code you write. There are several options to control the appearance of code in your document. They are all listed with explanations in the `.Rmd` file for your reference, and we will talk about those you'll need. These explanations are written as comments. Comments are preceded by <!-- and ended by -->, appear in green font in the `.Rmd` file, and will not appear in the final document.

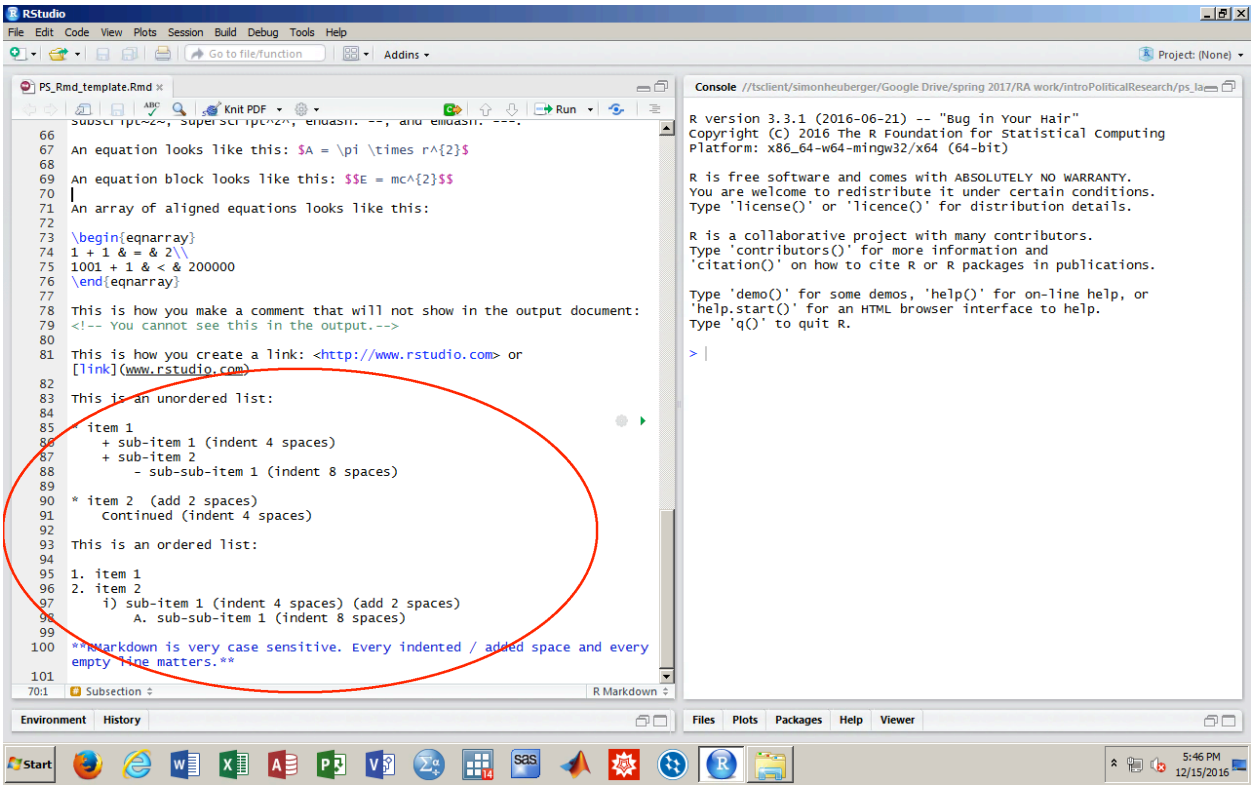


The command "# Section" creates a section. The command "## Subsection" creates a subsection. The grey chunk labeled "ggplot2" first loads a package which creates publication-quality plots, then loads a data set called "mtcars", and finally plots two variables "mpg" and "wt" from this data set. Because the default chunk option is "include = TRUE", this plot is shown in the final document.

(The code above is an alternative to the plotting technique we showed with the `lalonde` data. As we get into data analysis, we'll provide more specific syntax.)



This section shows you how to type mathematical expressions as well as bold, italic, superscript, etc. expressions. It also shows you how to make a comment and how to create a web link.



The final section shows you how to create ordered and unordered lists. **Lists require indented spaces (i.e. spaces before the expression) and added spaces (i.e. spaces after the expression) in order to be displayed properly**. For example, the expression "+ sub-item 1" in the unordered list needs to be indented by 4 spaces. The expression "i) sub-item 1" in the ordered list needs to be indented by 4 spaces AND needs to have 2 spaces added afterwards. **RMarkdown is very case sensitive. Every indented / added space and every empty line matters.**