Getting started with R

Before beginning our hands-on work in R, we first need to install R in our computers. We will also install RStudio, which is the environment where you will develop your programs!

Note: If you are already familiar with R, feel free to keep on using whatever environment you prefer.

Note: Things might look a little different in your computer than in mine. Don’t let that worry you!

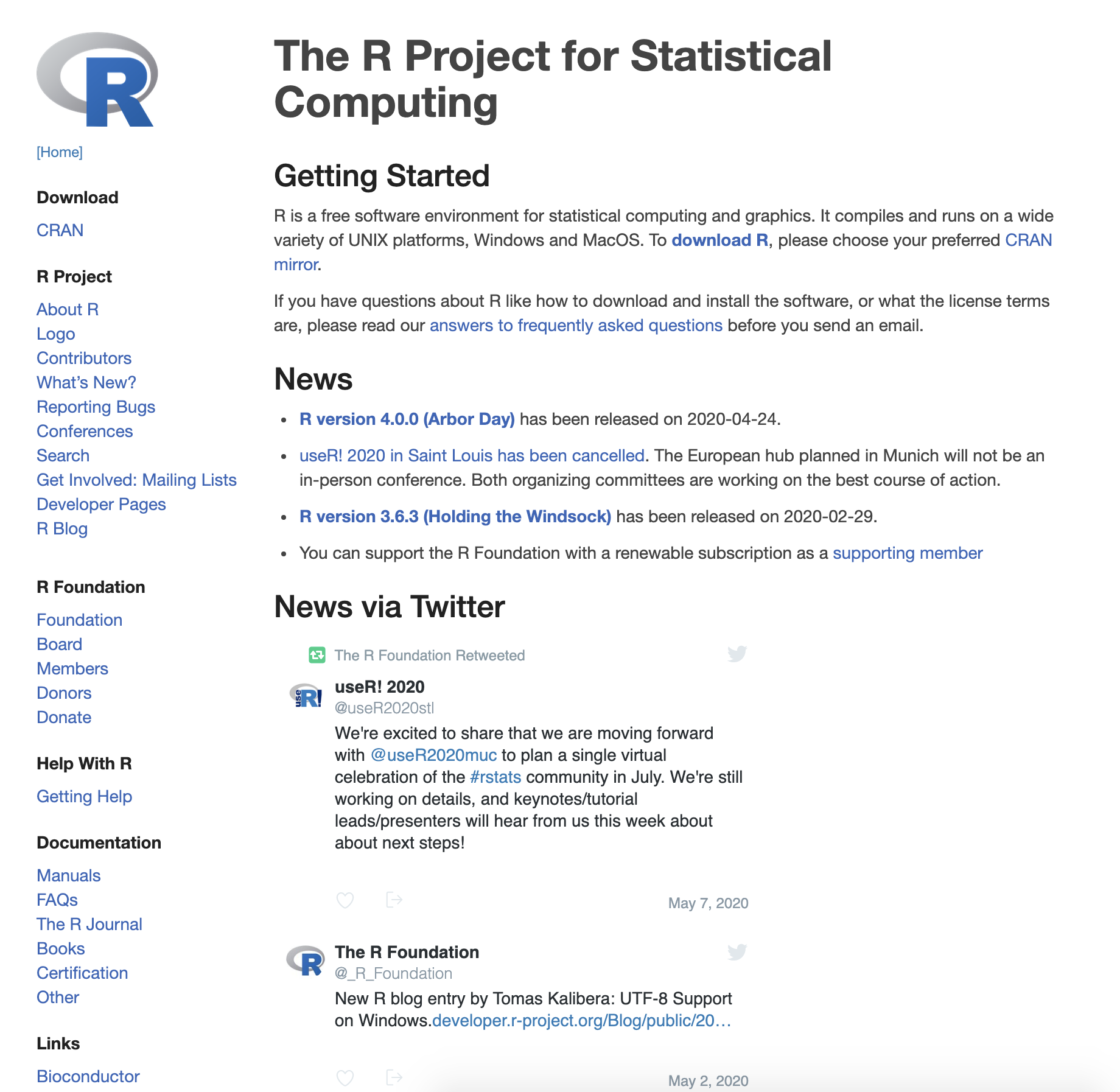
Note: When going through this document, it’s crucial to follow the instructions in order.

**Please reach out to the TA if you fail to follow any of the steps in this document.**

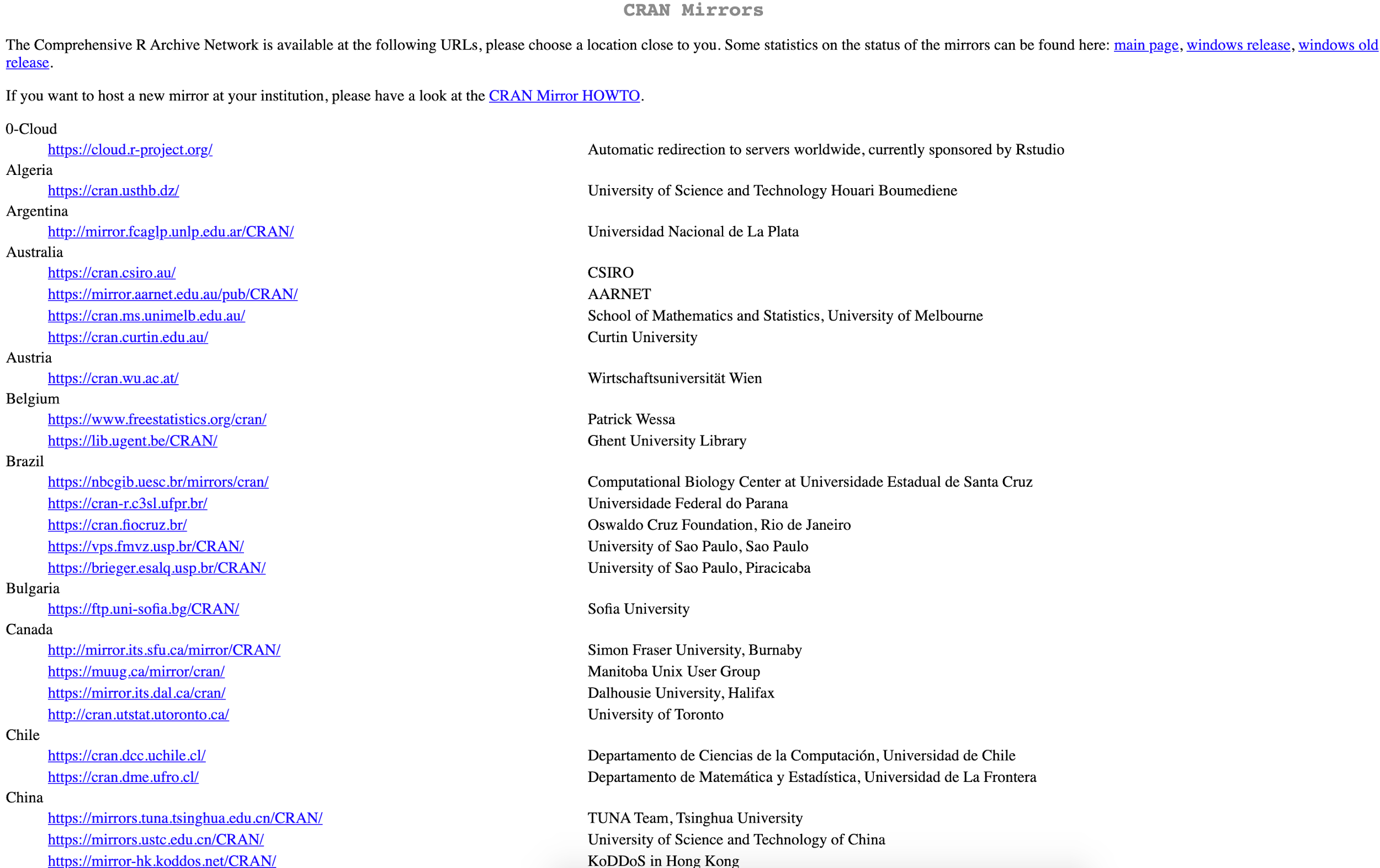
**It is important that you complete all steps before coming to the first class.**

## **Installing R**

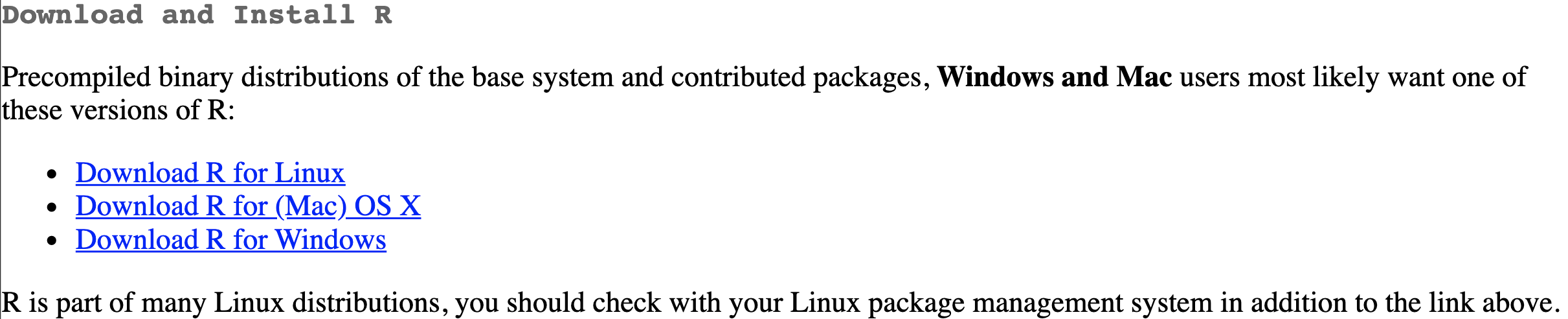
R is free, open source software for statistics. It is available for almost every popular operating system (Windows, Mac, Linux). The home of R is <https://www.r-project.org/>. Here is how this webpage looked on May 18th, 2020 on my computer:



Go to the [download page](https://cran.r-project.org/mirrors.html). You will see many links. These links are called “mirrors,” and are multiple locations that the Comprehensive R Archive Network (CRAN) is using to spread out the download load. Arbitrarily click on any one mirror (for example you can pick e.g., <http://mirrors.nics.utk.edu/cran/>---this is an example only, and you can select any mirror link from the list).



From this page, select the download that matches your operating system (Linux, Mac, or Windows). You want to select one from the “Precompiled binary distributions” list. You do not need the “source code”. From there, follow the instructions for your operating system.



Install R similarly to how you would install any program on your computer.

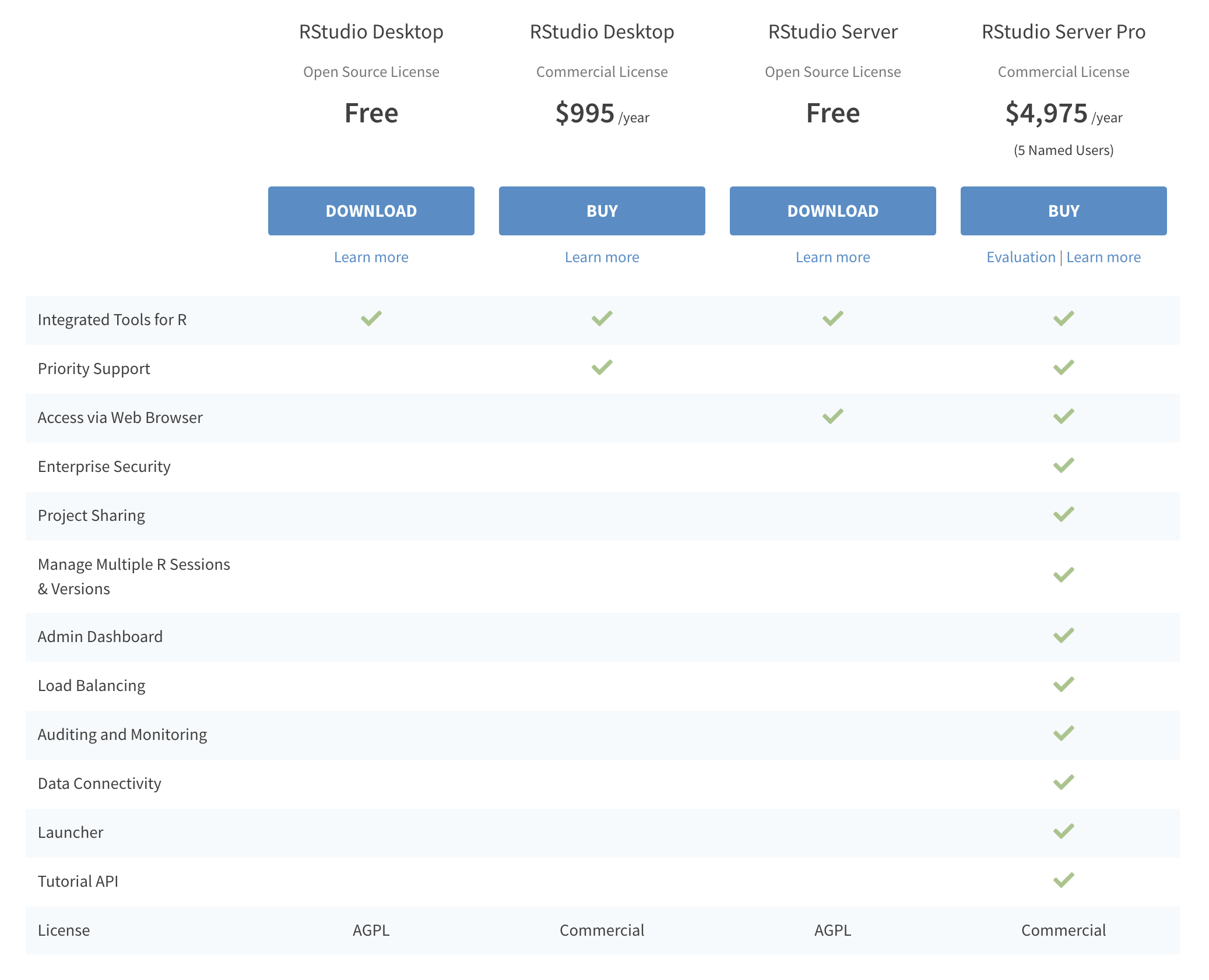
Note: For Windows users, it is recommended to check Save version number in registry during installation so that the R extension can find the R executable automatically.

## 

# **Rstudio**

## **1. Installing RStudio**

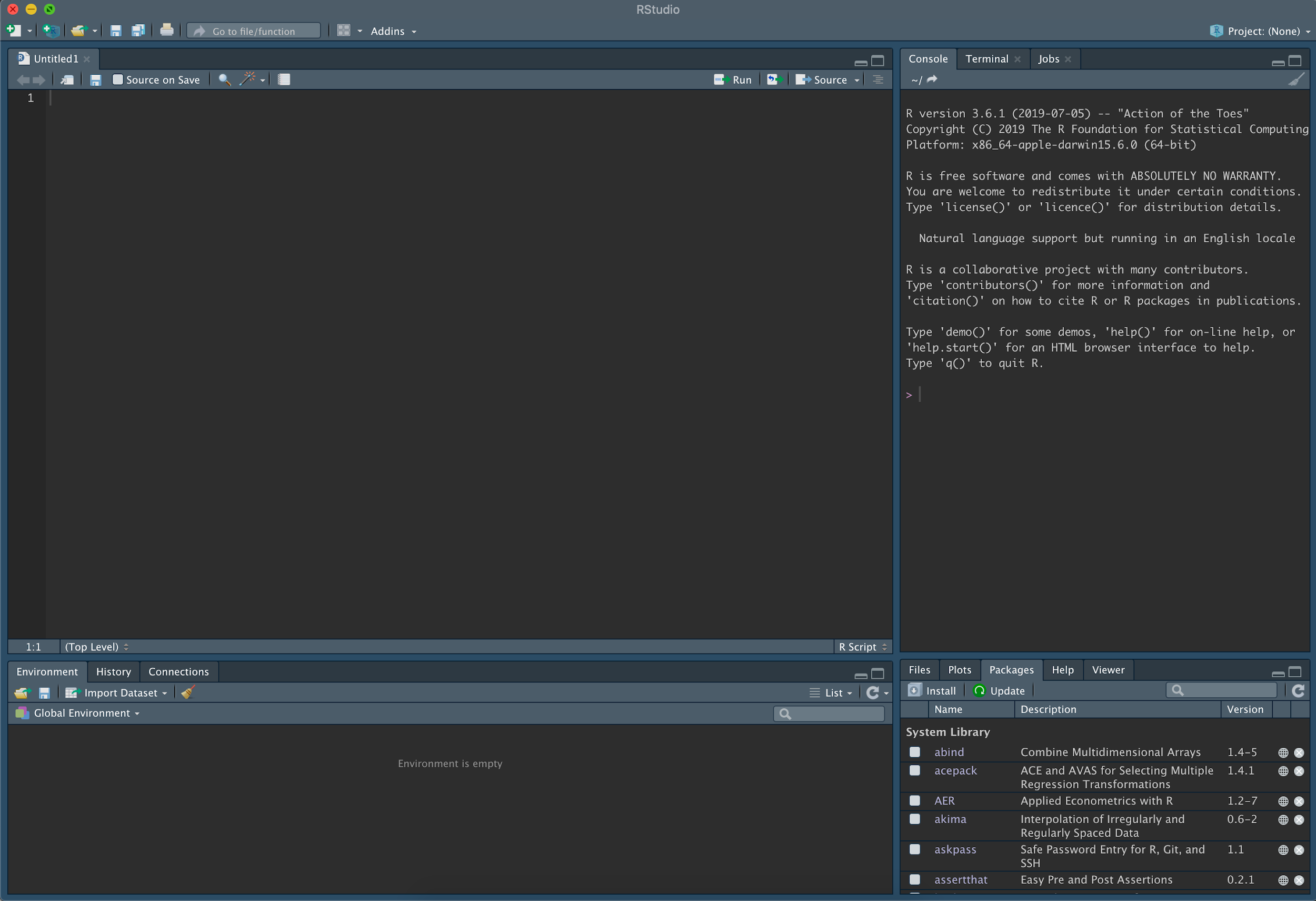
By itself, R is a bit unfriendly, and hence we will avoid using it directly. We will instead use *RStudio*,because it provides a nicer user interface. It is free and available from <https://www.rstudio.com/>, where it can be downloaded from https//[www.rstudio.com/products/rstudio/download/](http://www.rstudio.com/products/rstudio/download/). Install the free “RStudio Desktop: Open Source License” (on the left). Please make sure you have the base R installed before approaching this step!



Install RStudio similarly to how you would install any program on your computer.

## **2. Running RStudio**

Run Rstudio on your computer. Note that your screen may look different depending on the operating system (Windows, Mac, Linux), R version, and R distribution. Here is how it looks on my computer:



I prefer a darker background, but Rstudio will have a white background on your computer when you fire it up for the first time. You can change its looks by going to Rstudio->Preferences->Appearance. Regardless, your screen will consist of four panes/areas (you can also rearrange these panes as you like). These panes are:

1. Source ﬁles (upper left in the screenshot above)

R works by executing functions that you tell it to execute. These are ﬁles that you can use to save R functions that you use. You can have multiple ﬁles open at the same time. At the top of this window, there is a toolbar with shortcuts to many operations. For example, clicking “run” will execute the function that your cursor is on. Typically, you’ll want to save the functions you use into a ﬁle so that you can return later and execute them again. We’ll talk more about this later.

2. Console (upper right in the screenshot above)

The upper right quadrant contains the console. This displays the functions you execute from a source ﬁle as well as the results of that function. You can also just type functions here directly if you do not want to save them in a ﬁle.

3. Environment (lower left in the screenshot above)

The environment displays the current contents of R’s memory. As you work, you’ll see this area change to reﬂect how the functions you execute affect what is in R’s working memory. There is also a tab for “History” that lists all of the functions that R has previously executed. This is helpful if you remember a function you used but can’t remember it exactly.

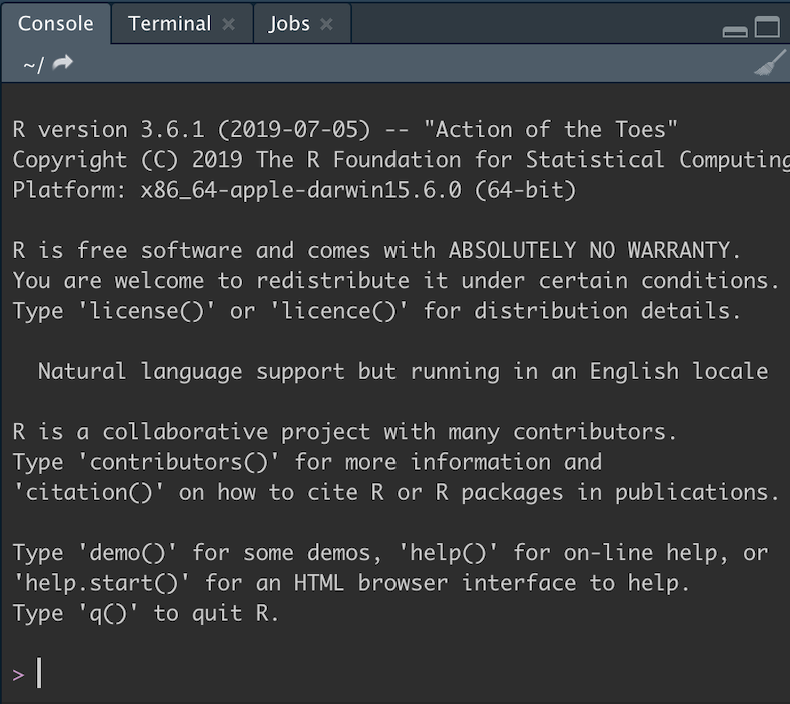
4. Other tabs (lower right in the screenshot above)

This pane contains several tabs. The “Files” tab displays the contents of your computer’s hard drive. The “Plots” area displays the results of plotting (these are not displayed in the Console like the results of non-plotting functions.) The “Packages” area lists collections of functions, called packages, that are used to extend R— we’ll go over that later. The “Help” tab provides help for functions.

*This quick overview does not cover the entire interface, but is intended to provide a first orientation. Try to familiarize yourself with it as much as you can, but do not worry if not everything is clear; you will become more and more comfortable using RStudio as we are progressing through our semester!*

## **3. Running your first R code**

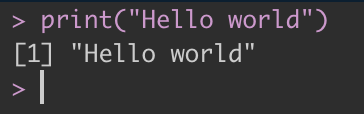
Let’s next run our first very simple code in R. Focus on the console pane, by clicking anywhere next to the “ > “ prompt; you will now see the “ | “ character flashing next to “ > “. Here is how it looks on my computer.



This is the place where you can write R commands. This means that your R is running and is awaiting for you to give it commands. Write the command

print("Hello world")

and press enter. You should get the following output:



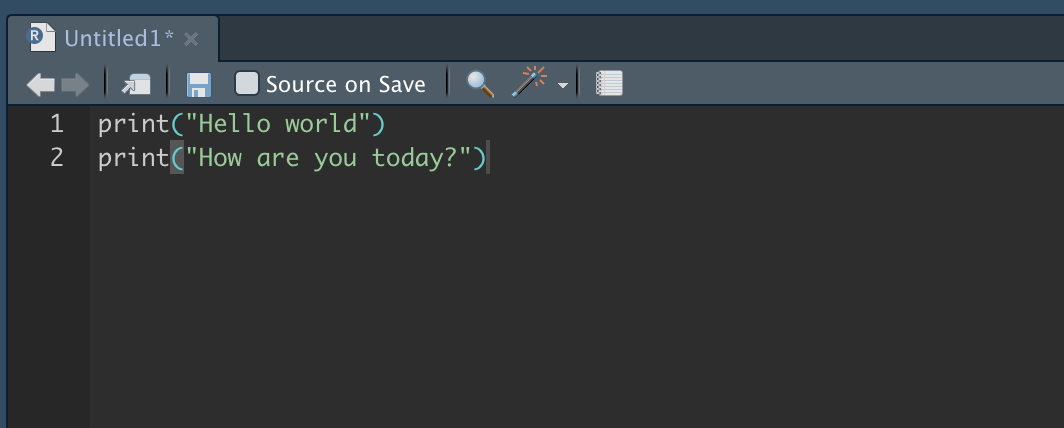
You just told your R to use the function **print** to display the string *Hello world*. It obeyed, and it printed out exactly what you told it to, and now it’s again waiting for your next command, indicated by the flashing “ | “ character.

Congratulations! You just wrote your first R script!

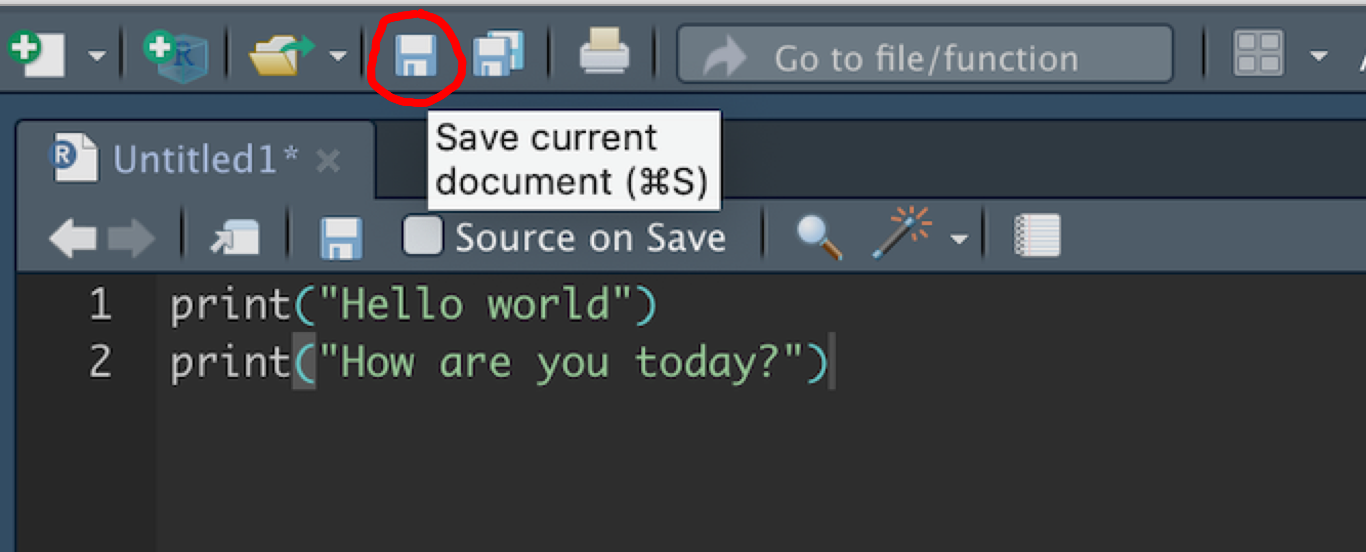
## **4. Making your first RScript**

Sometimes we want to run more than one command, and/or save the commands we run for later access. Script files in R solve exactly this problem.

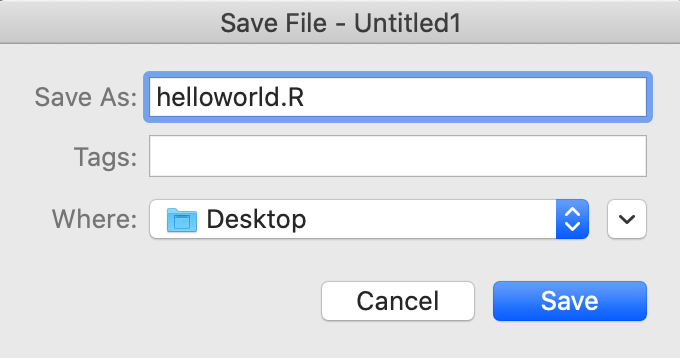
Let’s make a simple R script that first prints *Hello world*, and then prints *How is your day*? To do so, first go to the source files pane, and write the two commands in separate lines.



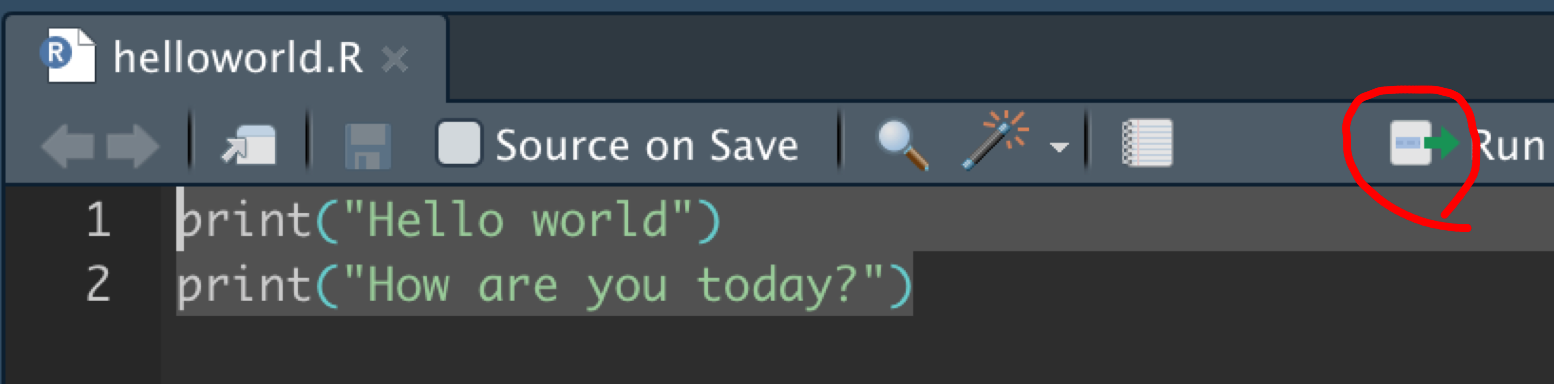
Notice that the file name (*Untitled*) now has an asterisk next to it. This means that changes have been made, but the changes have not been saved. Save the changes by clicking on the floppy disk icon icon.



Choose a name followed by the suffix *.R* (I picked *helloworld.R* ) and a location (I chose to save it on my *Desktop*).

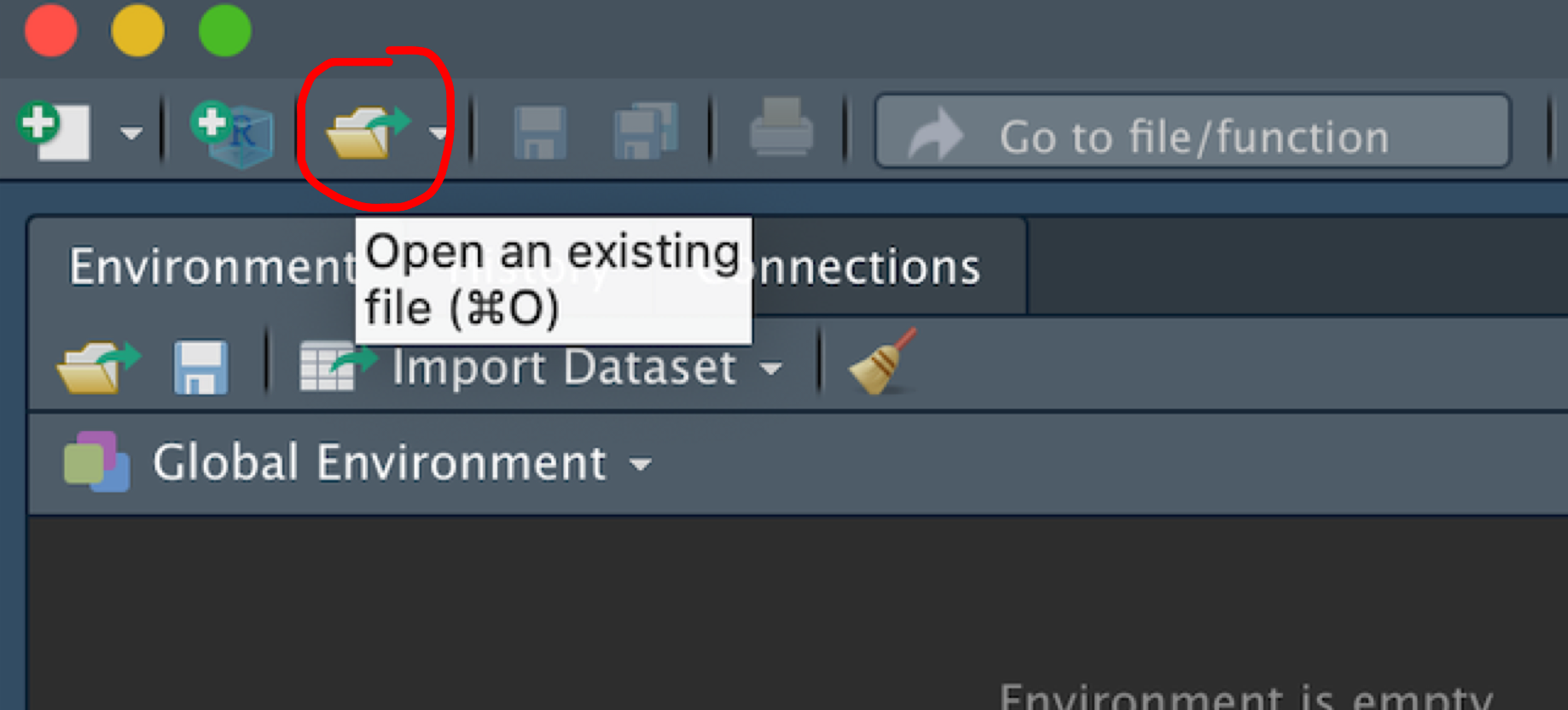


To run your script, simply select the lines you want to run (similar to how you would select multiple files on your computer), and press the Run button.

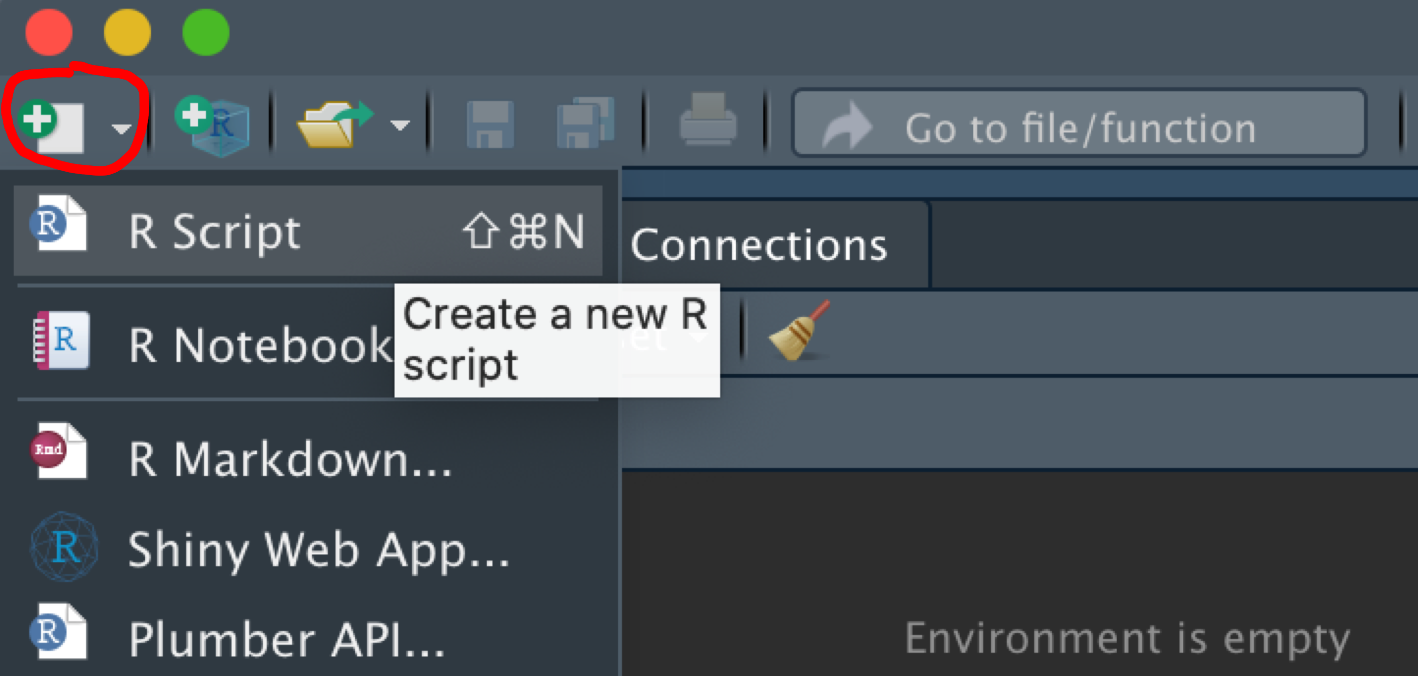


The commands you ran and the output of the code will appear on your Console pane.

You can now close your script (similar to how you close tabs in a browser, just click on the x button next to your tab) and reopen it by clicking on the folder icon on the upper left, finding the script file, and opening it.



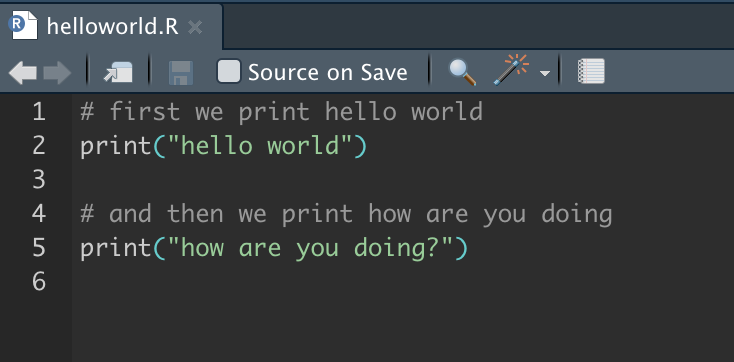
You can create new scripts by clicking on the green cross icon, and selecting *R Script*. Pretty easy!



The last thing that we need to know about R scripts is how to add comments. Comments are free-form language remarks on what (parts of) our code are doing. They are really useful for two main reasons: (i) it allows our collaborators to understand what we are doing, and (ii) it allows us to remember what we are doing when we review our code in the future.

Codes can be added simply by putting the # character anywhere on a line. Whatever follows the # character will be a comment, and R will treat it as such, and not try to run it as if it were code.

Continuing with our simple example, I added some comments to our first script.

****

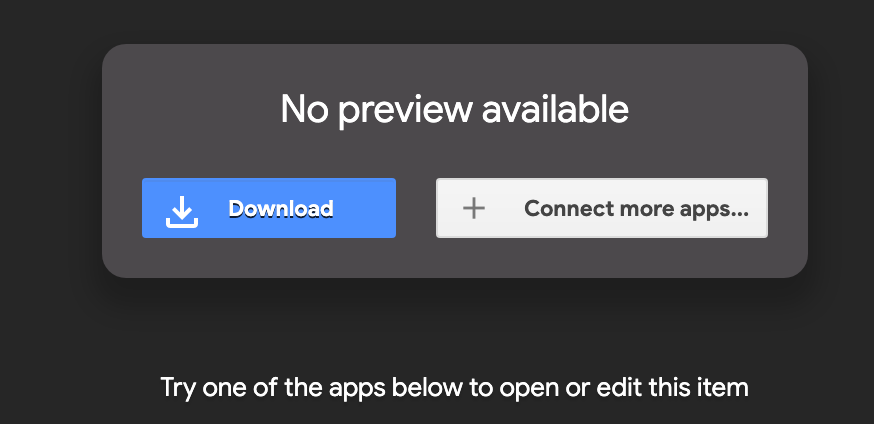
Adding informative comments to your code might seem like a drag initially, but it **will** **save you tons of time**.

## 

## **5. Downloading and running an RScript**

Sometimes you will want to download, find, and open scripts that have been written by others. This can be a little daunting at first, but it becomes very easy very soon --- once you get some practice.

First, download a script that I have written from [this link](https://drive.google.com/file/d/1Mb-LKYKTlJCE9x6ab0B8WD57jMFlD-pt/view?usp=sharing). Upon clicking on this link, Google will tell you that no preview is available for your file; simply select Download which will download the file on your computer.

****

To open this script, we now have to find and download it. Your browser downloads files in your computer in the *Downloads* folder -- for Mac users. Find the script, and move it to your *Desktop*. Now go to RStudio, and run the command getwd( ) in the console, to see what your **working directory** is

* Your working directory is the folder that Rstudio is looking at to find files on your computer.
* Because you moved the script that you downloaded in the *Desktop* folder, if your current working directory isn’t the Desktop folder, Rstudio will be unable to “see” and open the file

There are three ways to solve this problem.

* (worst way) move the script to your current working directory, and then open it through RStudio.
* (good way) set the working directory via point-and-click: go to *Session* (at the top), then select *Set Working Directory*, then select *Choose Directory*. You can then manually set your working directory.
* (best way) change your current working directory by using the command setwd("pathname") in your RStudio console, where "pathname" is the path to your Desktop.

To set the correct pathname,

* For all Mac and Linux users, the "pathname" should be set to "username/Desktop". For me, that means "apostolosfilippas/Desktop" but for you it will be different. A short way to refer to the username is by using the tilde (~) character, and type setwd("~/Desktop") instead.
* In Windows, my pathname would be "C:/Users/apostolosfilippas/Desktop". That means that I would have to type setwd("C:/Users/apostolosfilippas/Desktop").

Once you’ve set your working directory correctly, you’ll be able to open the file as shown in the previous section. Open it and run it. You’re ready for the first class!

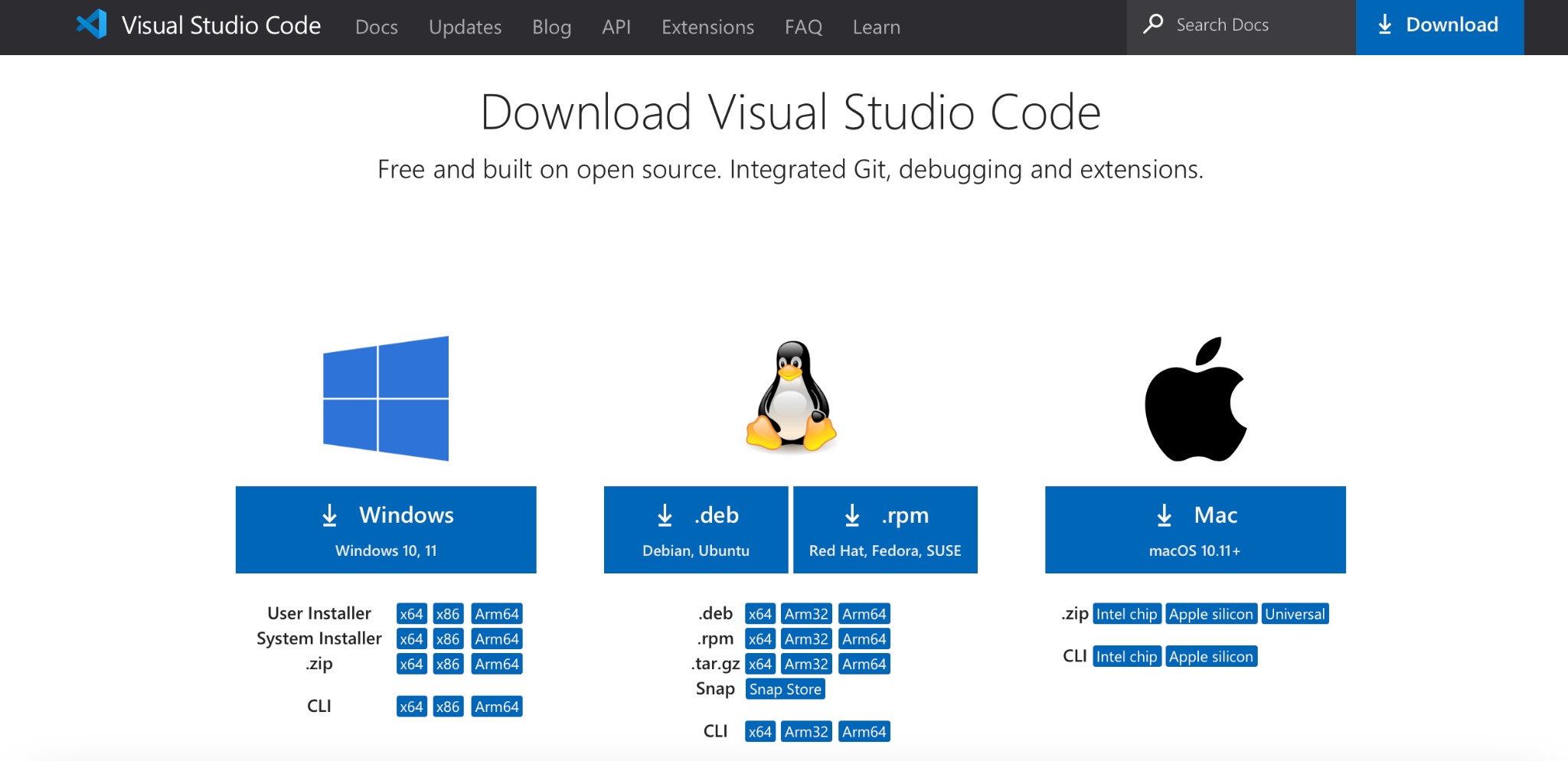
# **VSCode**

## **1. Installing VSCode**

VS Code is a fantastic IDE. It can be a great alternative to RStudio.

VS Code is lightweight, has a huge dev community, a ton of plugins, and the support of Microsoft behind it. It also integrates well with Copilot!

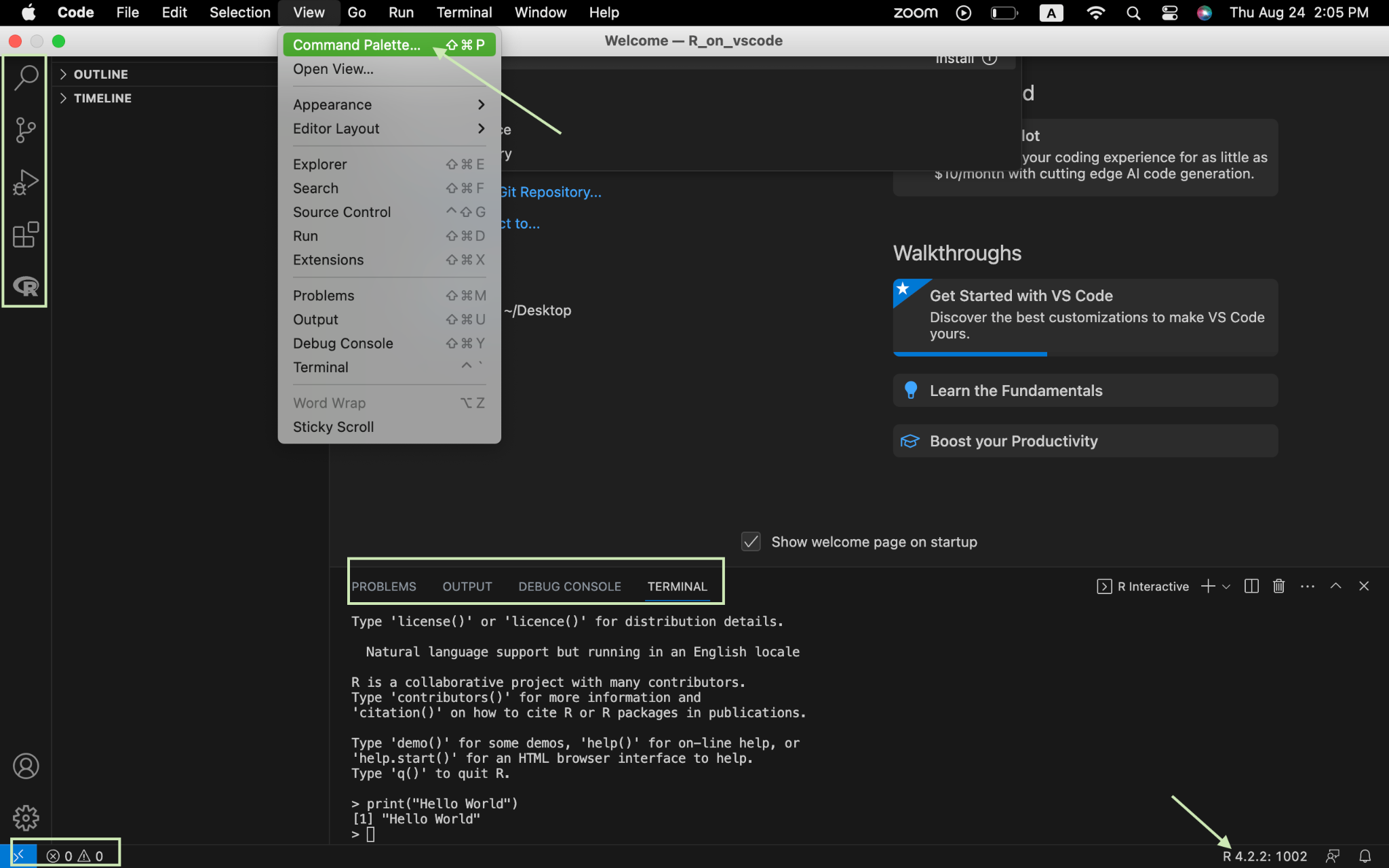
To install VSCode before, please go to this [link](https://code.visualstudio.com/download), and download the version that fits your system.



## 

## **2. Running VSCode**

Opening the VSCode, you can see the following parts:

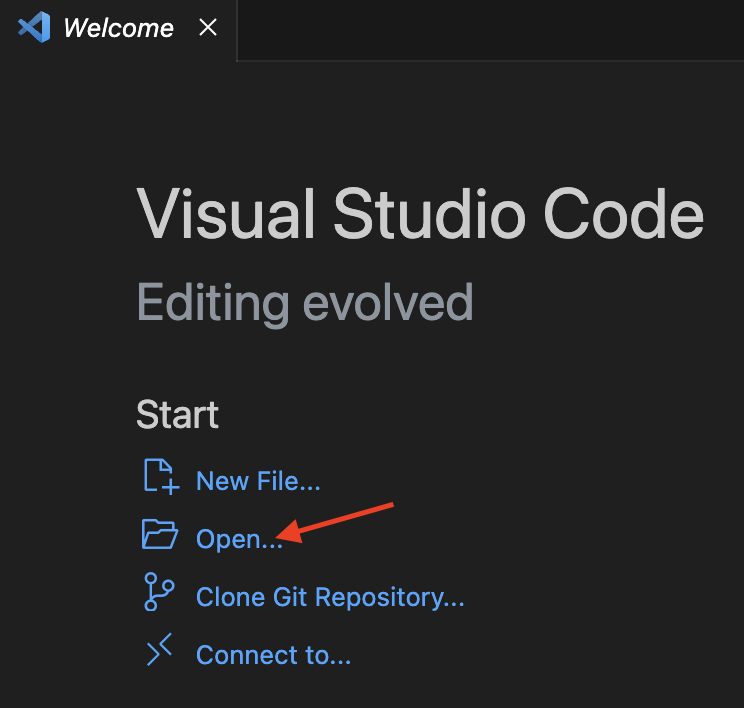


The left pane is the activity bar, for functions such as finding and replacing code, debugging, adding features, etc.

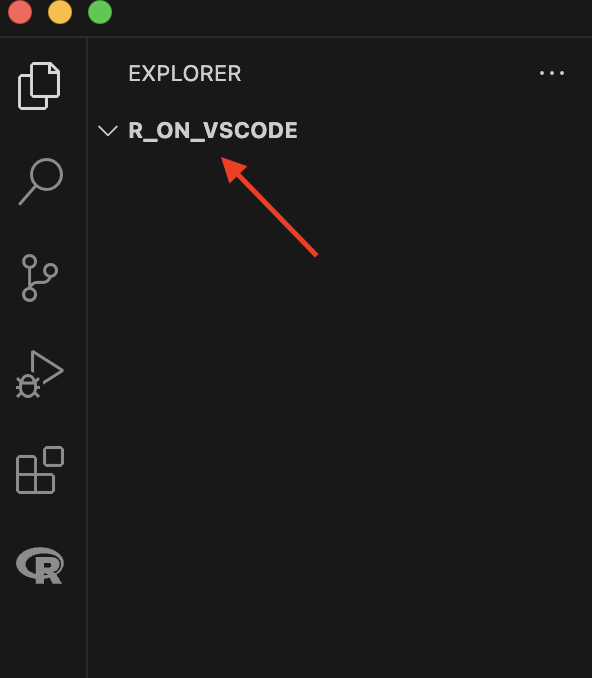
On the top we have the command palette, which is the control center for all commands in VSCode.

The bottom is the status bar. On the left, you’ll see the errors and warnings. On the right, you can see the current programming language.

To run VSCode, we should first set up the workspace. Click on open, choose the name and location you prefer.



Then you can find the new file under the explorer tab.



Please refer to [this link](https://code.visualstudio.com/docs/introvideos/basics) for more information about VSCode.

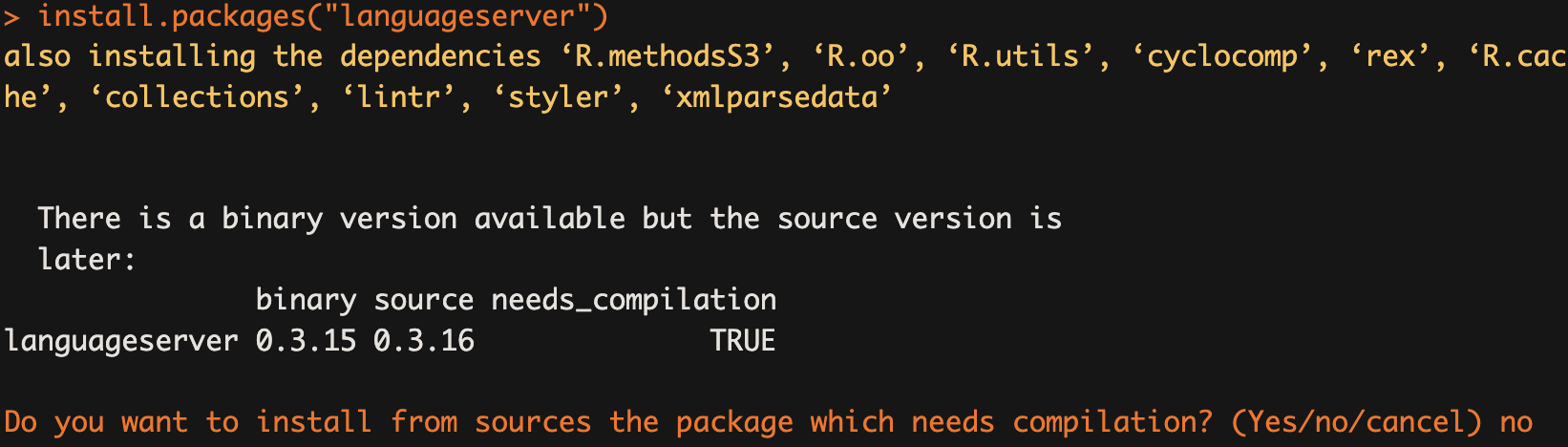
## **3. Setting up R in VSCode**

Now that we have R and VSCode installed, we can start setting up R in vscode.

First, install the languageserver package in R console using this code:

| install.packages("languageserver") |
| --- |

If this question (“Do you want to install from sources the package which needs compilation?”) pops up, reply no.



Second, install [the R extension for vscode](https://marketplace.visualstudio.com/items?itemName=REditorSupport.r). This extension would give you some hints that make programming easier.

Now you can code in R on VSCode.

Please refer to [this link](https://code.visualstudio.com/docs/languages/r) for further information.

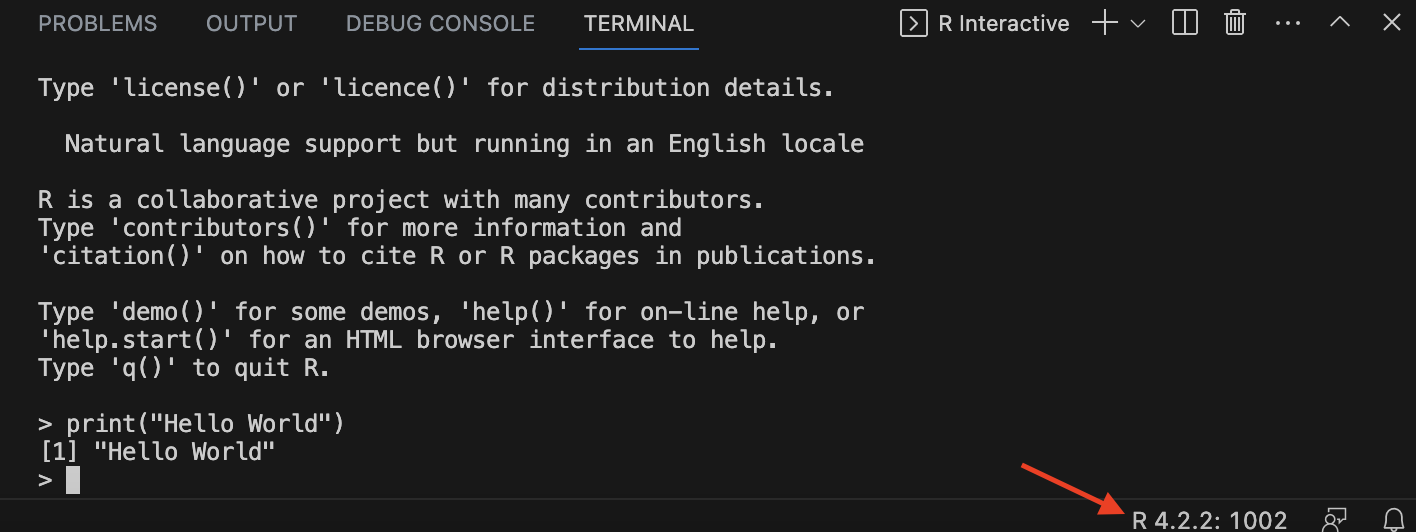
## 

## **4. Running your first R code**

To run your first R code, you can create an R terminal in the command Palette using command:

**R: Create R terminal**

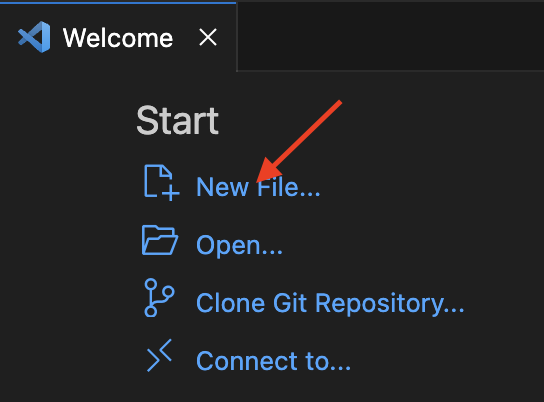
Having the programming language switched to R, you can now try out R code in the terminal.

****

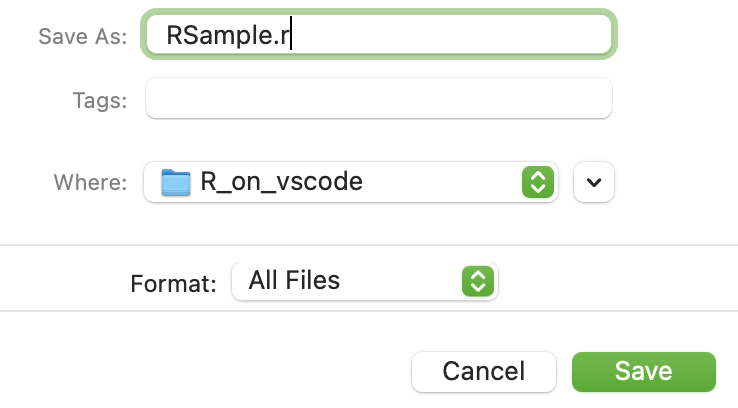
## 

## **5. Making your first R script**

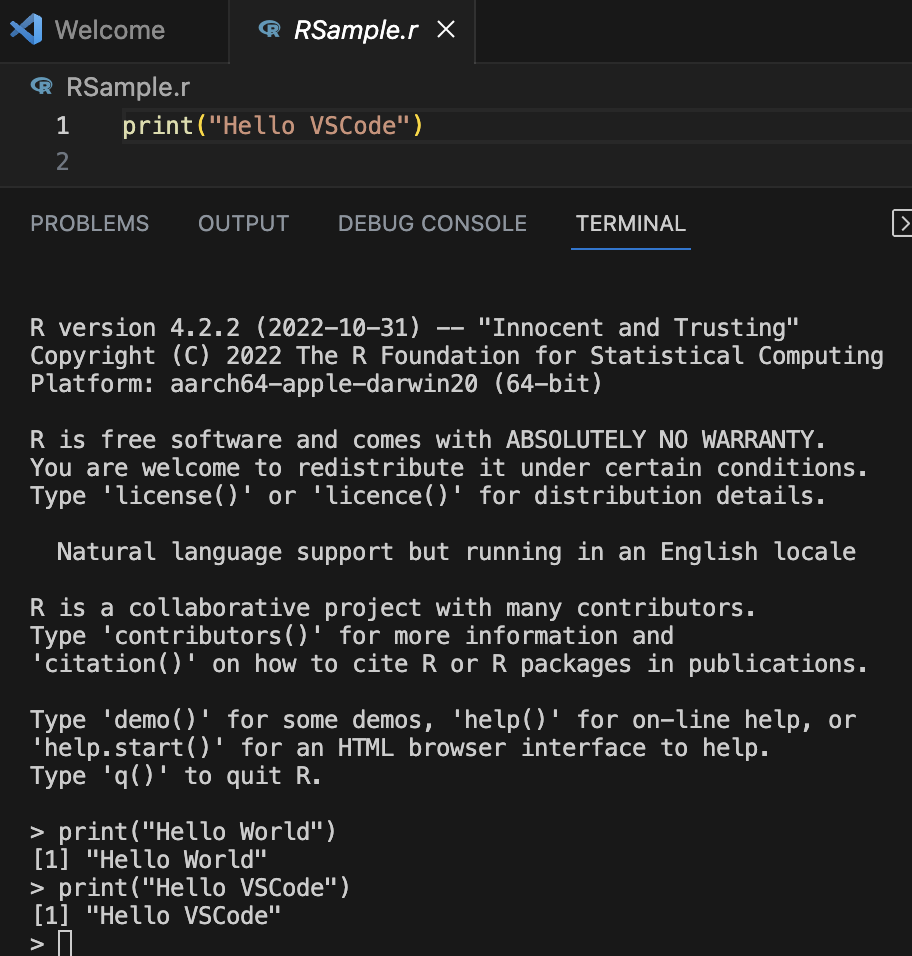
Create an R script by clicking on the new file, choose R document.



Save the document in the workspace you just created.



Now you have created your first RScript in VSCode and will be able to execute R Code.



You can choose Auto Save here to automatically save your code.

