**Introduction to Bioinformatics Tools**

*Introduction*

In botany as well as other fields in biology, it is increasingly important to have some facility with bioinformatics tools. Many of these tools are accessed from the “command line” or the “terminal”, rather than from a graphical user interface (GUI), that is using a mouse. Rather, these bioinformatics tools either require or work much better when commands are entered in text form. These tools and commands are entered through a Unix/Linux environment.

What is Unix/Linux? Unix is a family of commands used to perform tasks on a computer. Linux is a particular version of Unix. And Linux itself comes in a variety of different flavors (called distributions). Our task for today will first be to make your computer capable of accessing a Linux terminal. We will then learn how to access and navigate the terminal. Today’s lab is in preparation for a later lab where we will use bioinformatics tools through the Linux terminal.

The other goal of today’s lab is to get R and R Studio installed and functioning on your computer. Once this has been accomplished, we will install a number of what are called “packages”. When you initially install R and R Studio, you install a “base” version of the program. It has a number of functions built in. The packages are extra functions that you can add on to base R to expand its functionality. It is similar to when you buy a new computer. A new computer often comes with some programs pre-installed. Other programs you have to install yourself. Which new programs you install will depend on how you will be using the computer. Similarly, we will install the R packages that will be necessary for the work we want to do. If you continue to use R in your personal research, you will likely install more packages in order to accomplish specific tasks.

**Part I**

**Download Files**

Start by downloading the zipped folder associated with today’s lab from Moodle. This folder will have some files for us to play with while we do this introduction. Place the folder on the Desktop of your computer and unzip the folder. If you are using a Chromebook, put the folder in “My Files”.

**Part II**

**Install/Access Linux**

**Mac:** If you have a Mac, a Linux terminal is already available to you. Press Cmd+Space. This will open the finder. In the search bar type “Terminal.app”. The first application that should come up in the search is the terminal. Click on it to open it. It is probably best to make sure everything is up to date. First type “sudo apt update”, and hit Enter. The “sudo” part means run-as-administrator, so you will be prompted to enter your password. Once this is completed, type “sudo apt upgrade”, and hit Enter.

For other ways to access the terminal, check here (<https://www.howtogeek.com/682770/how-to-open-the-terminal-on-a-mac/>).

If you have either a Windows PC or a ChromeBook, you do not have access to a native Linux terminal without installing/configuring one.

To put Linux on a **Windows PC:** select the magnifying glass in the bottom left-hand of the screen. In the search bar type “powershell”. The first result should be “Windows Powershell”. In the pop-out window to the right, select “Run as Administrator”. Windows will likely ask if you want Powershell to make changes to your computer. Click “OK”. The Powershell will then open. This is a terminal but uses a language unique to Windows. In the Powershell, type the following “wsl --install -d ubuntu”. Then hit Enter. This should activate the Windows subsystem for Linux, download Ubuntu (a Linux distribution), and install Ubuntu. This will take a little time. When it is complete, you need to restart your computer. When the computer has restarted, then go back the magnifying glass in the bottom left-hand corner. Type “Ubuntu”. It should be the first result, click on it to run it. The first thing that you will need to do is setup a username and password. It is very important to remember this username and password. The Ubuntu Linux is like a parallel operating system. Once this is done, you are ready to go. It is probably best to make sure everything is up to date. First type “sudo apt update”, and hit Enter. The “sudo” part means run-as-administrator, so you will be prompted to enter your password. This is the password you just created. Once this is completed, type “sudo apt upgrade”, and hit Enter. For more information about getting Linux on your Windows computer, see here (<https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-11-with-gui-support#1-overview>).

Linux comes pre-installed on a **Chromebook.** To enable it, go to “Settings” (the gear), then “Advanced”, then “Developers”. Look for “Linux Development Environment” and turn it on. If you have never used the terminal before, you will need to do some setup. When it asks you how much space to give the Linux container, shoot for 5 Gb, but 2 Gb at an absolute minimum. You may need to restart the computer. Once that is complete, you should be good to go. To find and use the terminal, hit the search key on your keyboard. In the search window type “terminal”. It should be the first result, click on it to open it. It is probably best to make sure everything is up to date. First type “sudo apt update”, and hit Enter. The “sudo” part means run-as-administrator, so you will prompted to enter your password. If you were prompted to make a new password during terminal setup, it will be this new password. Once this is completed, type “sudo apt upgrade”, and hit Enter. For more information about getting Linus on your Chromebook, see here (<https://helpdeskgeek.com/how-to/how-to-open-the-linux-terminal-on-chromebook/>).

**Part III**

**Basic UNIX/Linux Navigation**

Open the Terminal. This will be done differently based on what system you are using Mac/Windows/Chromebook. The first thing that we want to learn how to do is to navigate in the terminal. In a GUI (Graphical User Interface) environment, you navigate by pointing and clicking a mouse. This will not work in the terminal, all you can do is type on the keyboard.

*pwd*

The first command we want to learn is “pwd” (print working directory). This requires some vocabulary explanation. A ‘directory” is a computer science term for a folder. In the terminal, you are always inside of some folder, some directory. And navigation is the act of moving between folders. Printing the working directory will tell you which folder you are currently in.

*ls*

The next command is “ls” (list). This lists all of the contents of the current directory, including any files as well as other directories/folders/

*cd*

The next command is “cd” (change directory). This command allows you to change the working directory (the folder you are currently in). The syntax is:

*cd directorypath*

Thus, you use the command but need to supply where you would like to go. Each directory in the path is separated by a “/”. To move up a level in the directory system, the following command is used:

*cd ../*

For example, given the following hypothetical directory structure:

A

C

B

E

D

You would move from directory (folder) A to directory D, with the following command:

*cd B/D/*

And you could move from directory D to directory C, with the following command:

*cd ../../C/*

Before going any further, let’s get into the folder that we downloaded at the beginning of lab. Navigating to this folder will be different depending on the system you are using.

**Mac:**

c*d Desktop/FolderName*

**Windows:**

*cd ../../mnt/c/Users/HP/Desktop/FolderName*

Note: This might be slightly different if you don’t have an HP computer. See me for assistance.

**Chromebook:**

*cd mnt/chromeos/MyFiles/FolderName*

We should now be in the folder with the files for today. Do an *ls* command to list the files in this folder, and check to see if this is true. There are several ways to view files in the terminal. We will cover four: *head*, *tail*, *less*, and *cat*. Head and tail are useful for very large files. Head will display the first ten lines of a file and tail will show the last ten lines. Less will enter an interactive viewer where you can use the arrow keys to scroll up and down in the file, and exit by pressing the “q” key on the keyboard. Cat (concatenate) can be used for a number of tasks, but here we can use it to view the whole contents of the file. In the case of a very large file, this may crash the terminal. The syntax is a follows:

*head FileName*

*tail FileName*

*less FileName*

*cat FileName*

Try each of these out on the file “1\_trnH-GUG\_tRNA\_feature.fasta”, which contains chloroplast DNA sequences of the *trnH* tRNA gene for a number of plant species.

In many cases, file manipulation can be done with a mouse in GUI. But there are times, especially when working with very large files, where it is a better idea to manipulate files from the command line. A few of the basic file manipulation commands are cp (copy), mv (move), rm (remove/delete), and mkdir (make directory/folder). The syntax for each is a follows:

*cp FileName Directory/Path/*

*mv FileName Directory/Path/*

*rm FileName*

*mkdir NewDirectoryName*

As a summary exercise, navigate back to the desktop, make a new directory, navigate back into the directory with today’s examples, and copy one of the files into the new directory.

As a final exercise, we can learn how to remove directories/folders from the terminal. This requires us to introduce a new concept when it comes to Linux, and this is that each command comes with a series of “options”. These options modify the basic functionality of the command. An option is generally preceded by a “-“, and indicated by a letter. For example:

*rm -r DirectoryName*

will remove a directory and all of its contents. And

*rm -d DirectoryName*

will remove an empty directory. You can seem more options of the rm command on the manual page (<https://man7.org/linux/man-pages/man1/rm.1.html>).

The terminal can be exited like any other program by clicking on the X in the upper right corner, or by executing the command “exit”.

**Part IV**

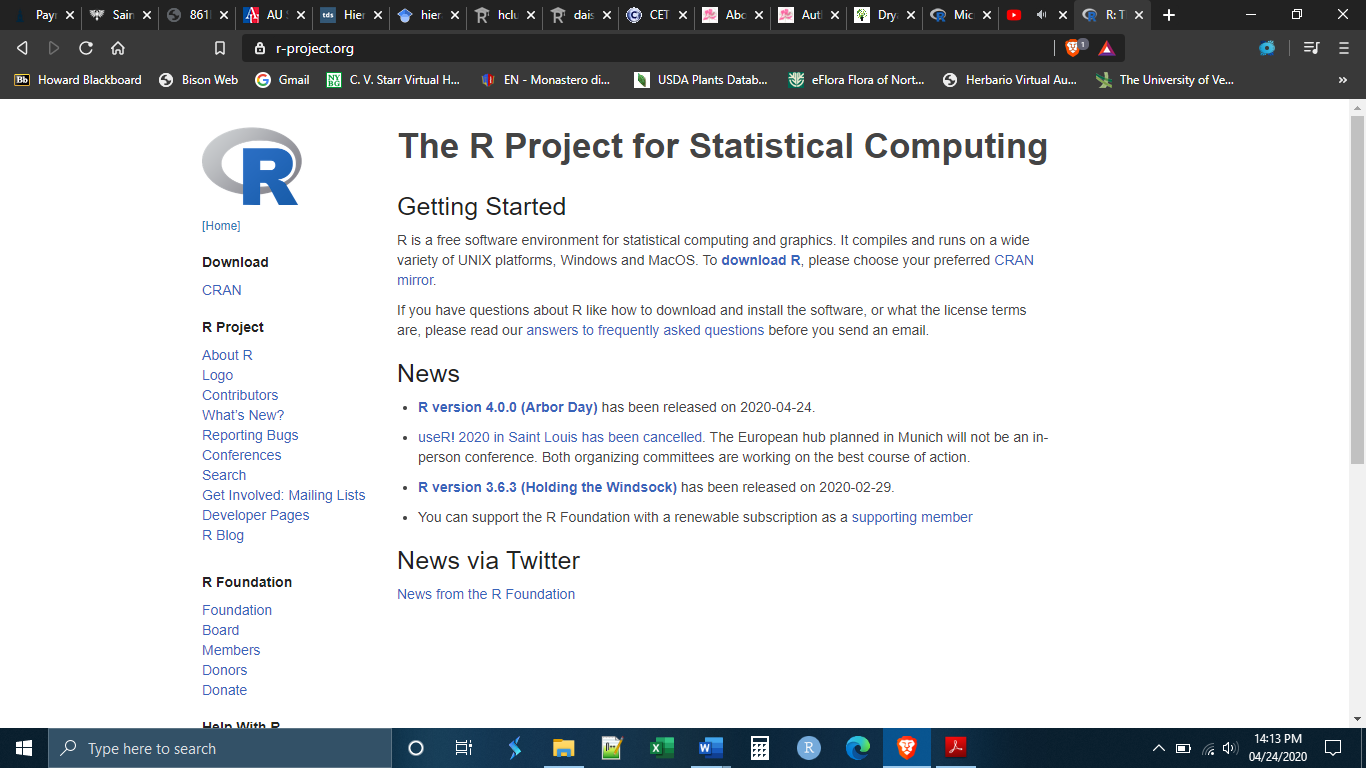
**Install R and R Studio. Install Packages.**

*Installing Base R on Mac or Windows*

For those using a **Mac** or **Windows**, installing R and R studio can be done like any other program (for those using a **Chromebook**, see starting on page 11). We will begin by installing the base R program.

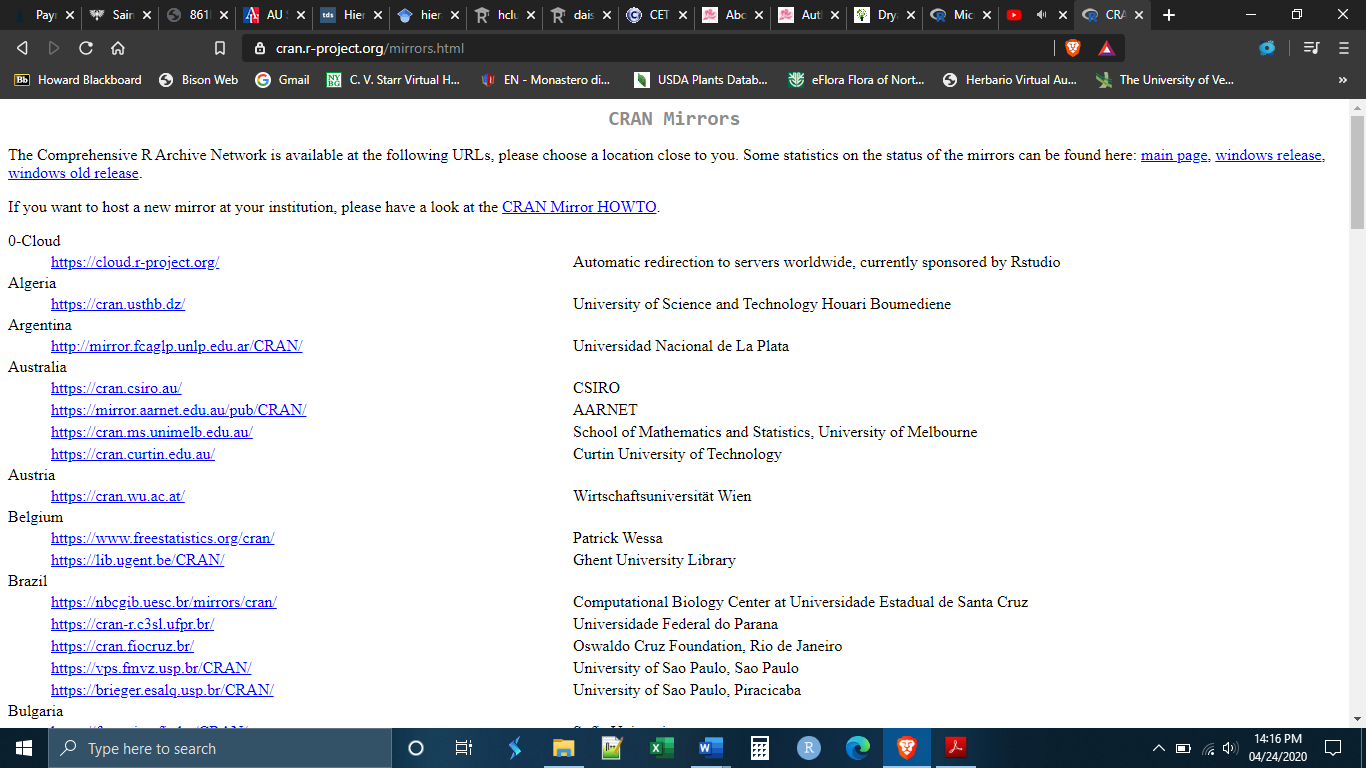
Go to the R website: https://www.r-project.org

Next, click on the blue link **download R**

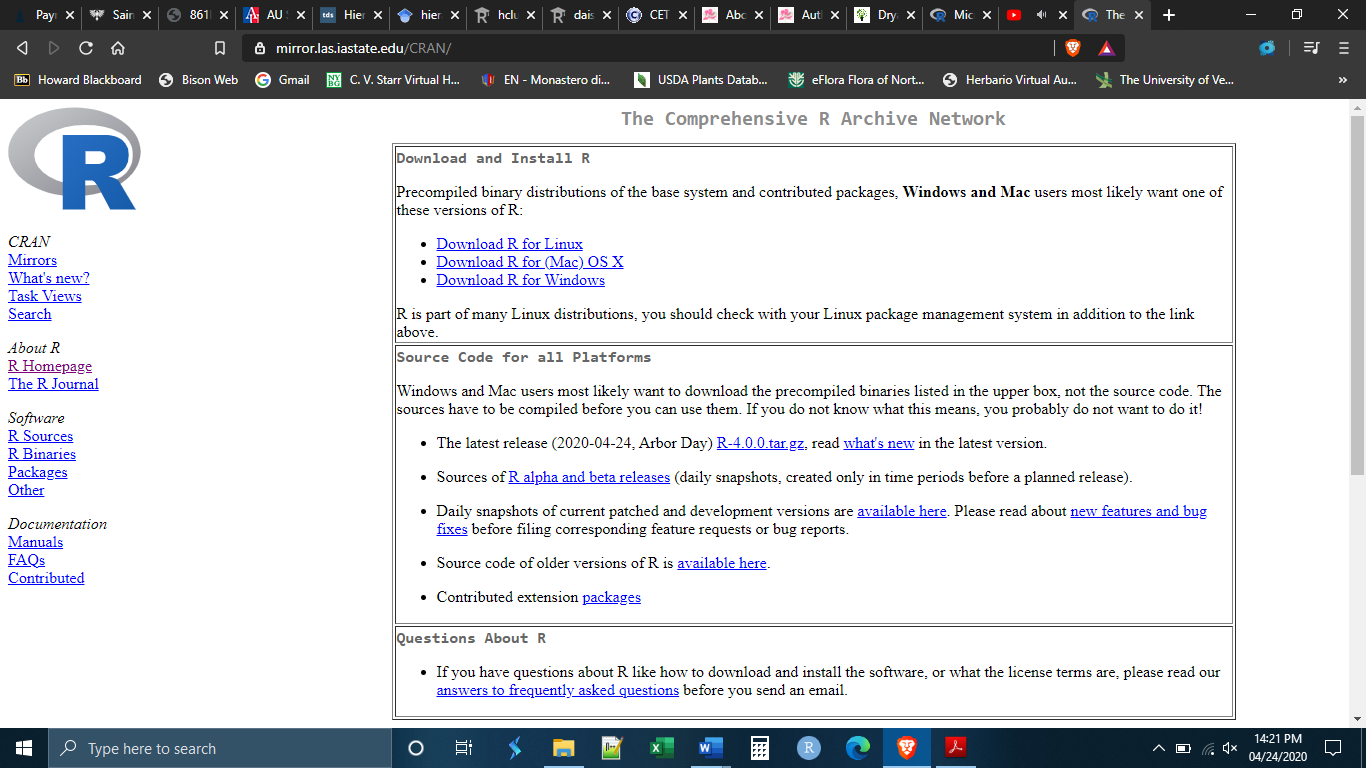


You will be taken to a new screen, asking you to choose which CRAN Mirror you want to download R from. CRAN is an acronym for “Comprehensive R Archive Network”. Basically, there are a number of places in the world where the R installation files are kept and you have to choose which one you want to download from. Hypothetically, they are all regularly updated and should be the same, but it is best practice to choose one that is physically close to you.

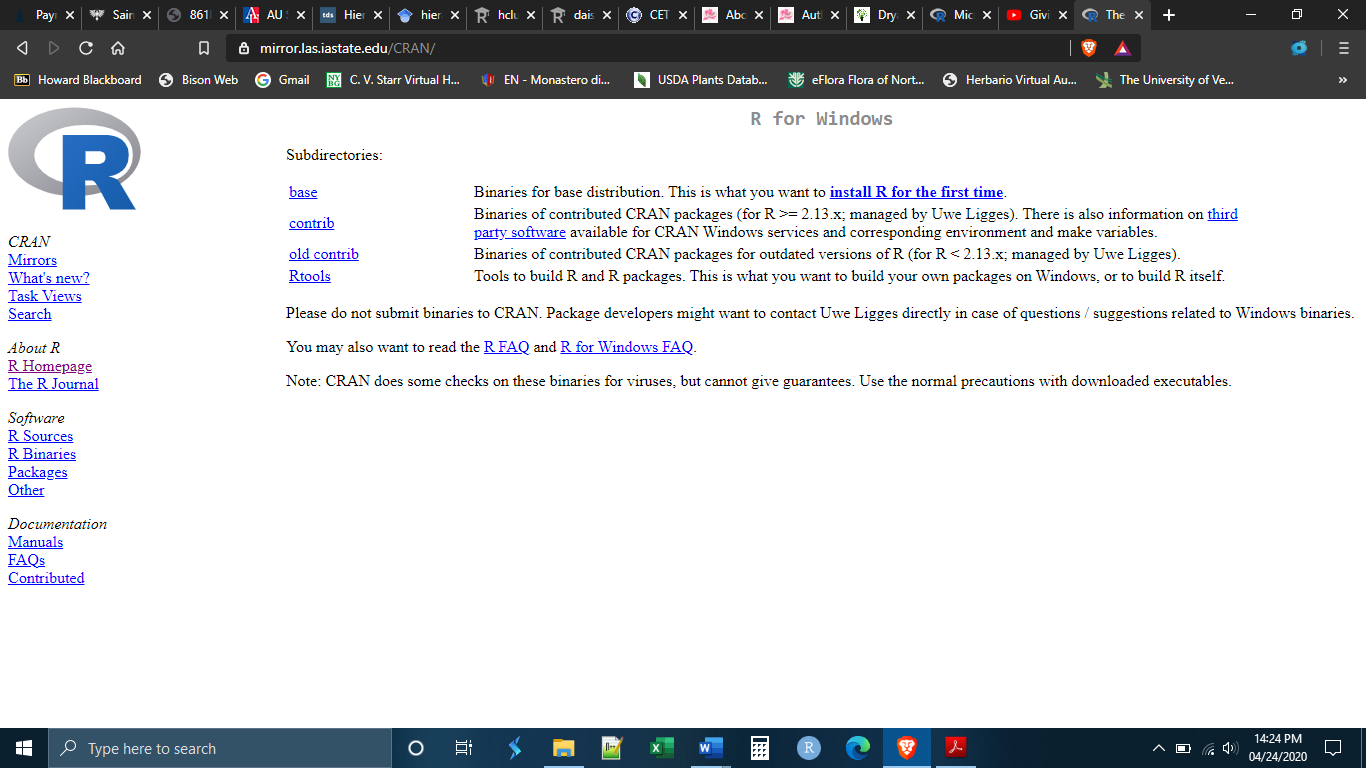
Scroll down and choose one of the CRAN Mirrors in the United States by clicking on the link.



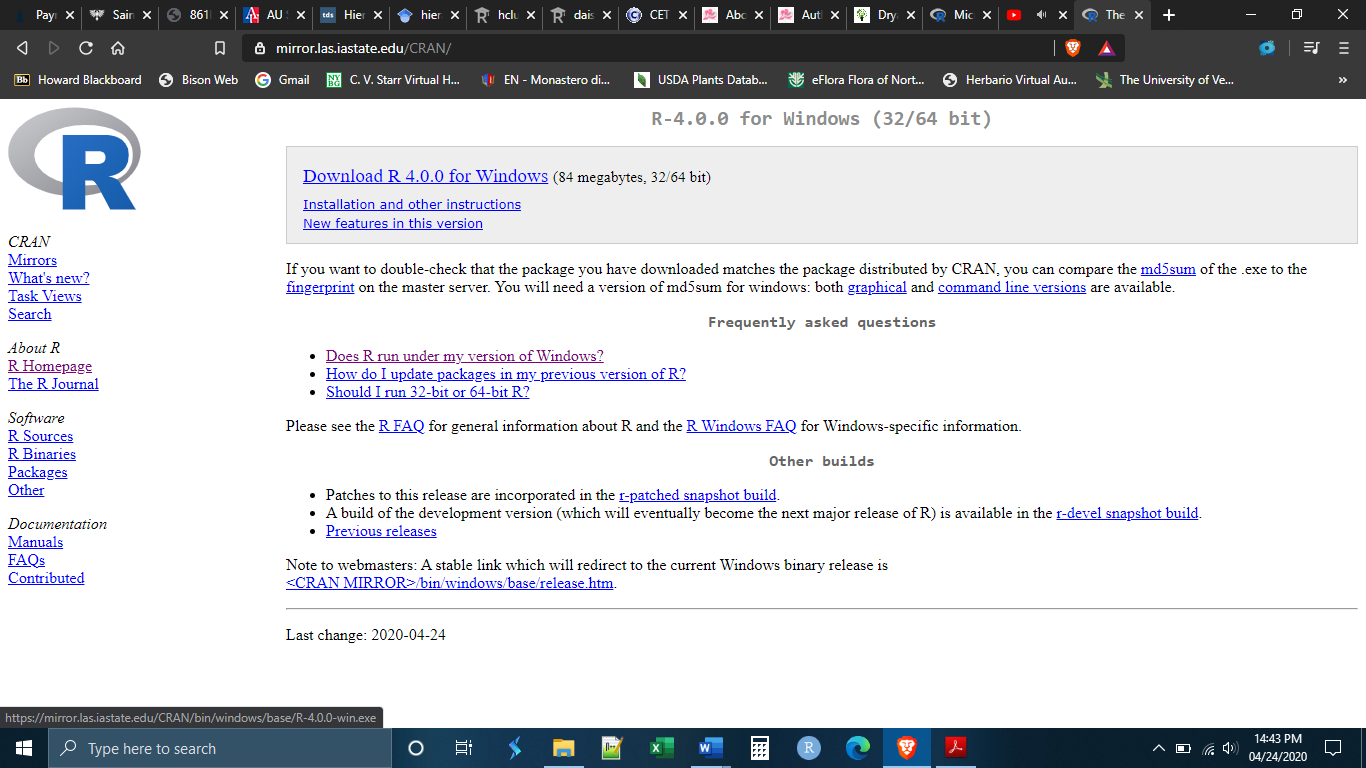
Once you have established a secure connection with a CRAN server you will get a new screen asking you which version of R you would like to install. Click on the link corresponding to the system that your computer is running (i.e. Linux, Windows, or Mac)



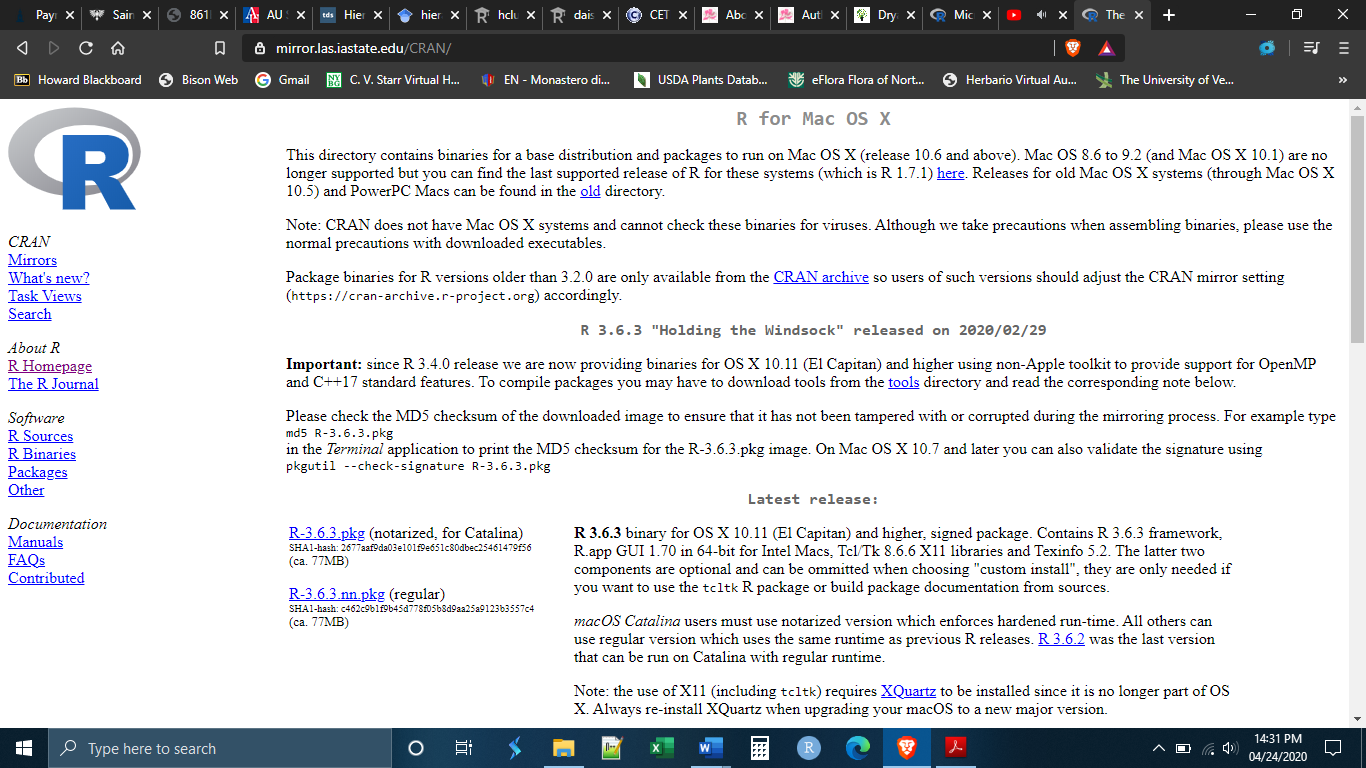
If you are running Windows, you will be sent to another screen asking which files you would like to download. Choose “Install R for the first time” or “base”.



You will be brought to this page, click on “Download R 4.0.0”



If you are using a Mac, the most recent release of R, has two different versions depending on Mac OS. You can select the appropriate version of R based on your OS. Alternately, you can choose to install the most recent version of R that works on all Mac OS - R 3.6.2.



Once you have downloaded the file, go to your downloads folder, or wherever you have set downloads to go, and run the .exe file to install the software. You may be prompted to give R permission to install itself, click yes.

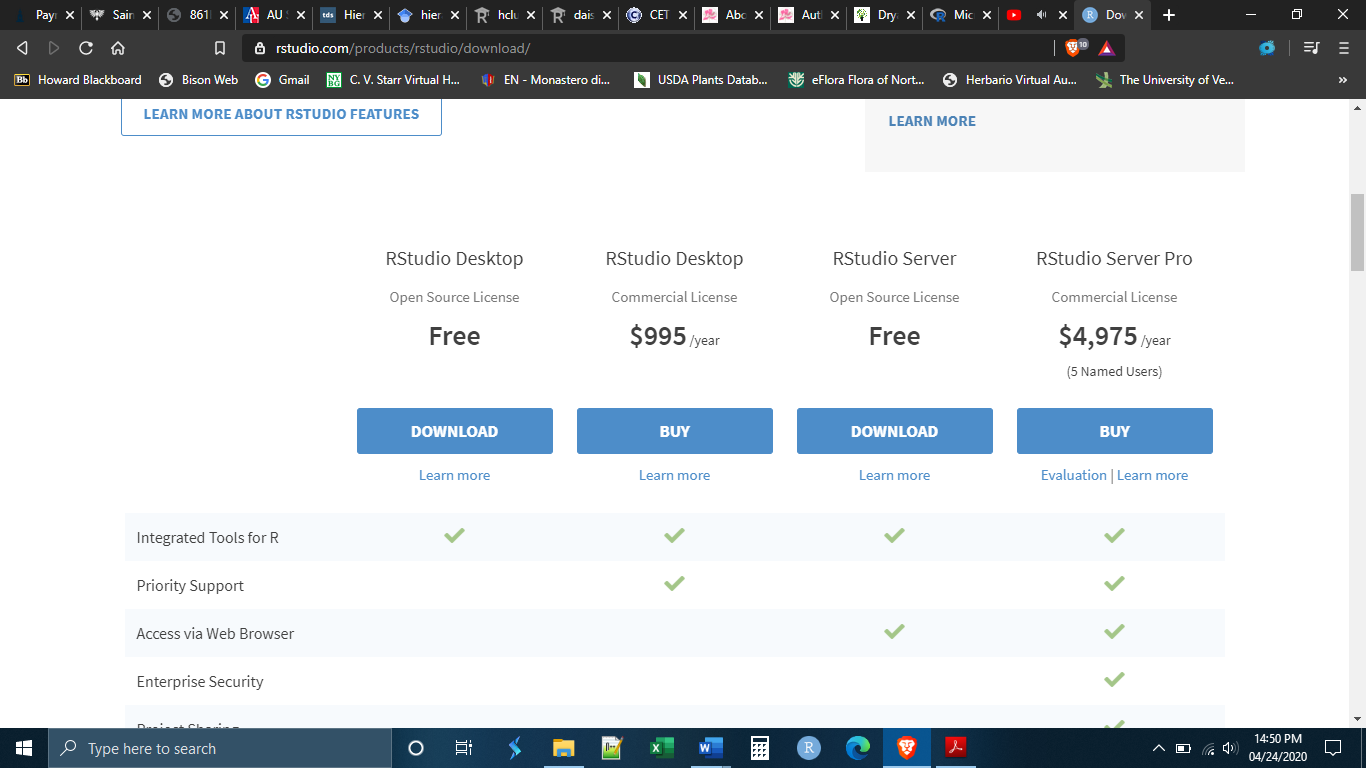
This will install base R on your computer.

*Installing R Studio*

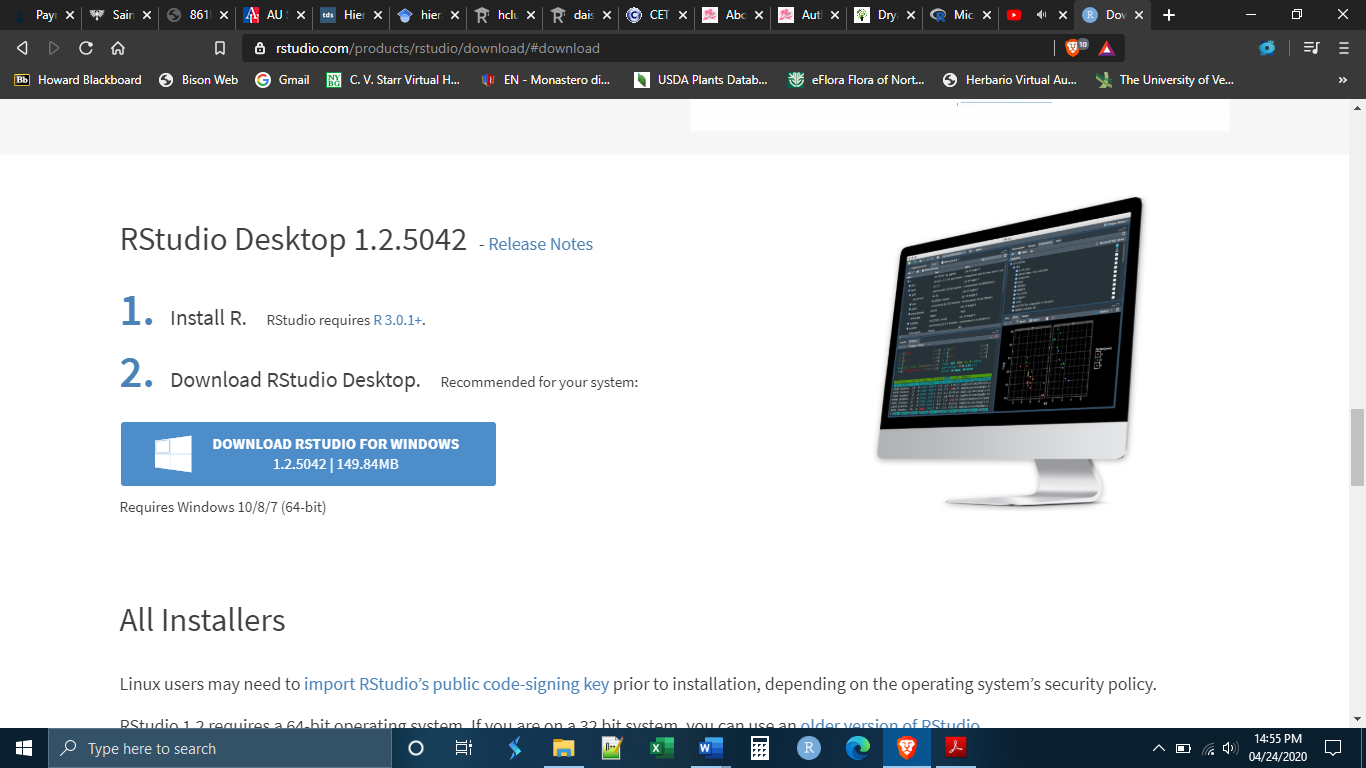
You can run base R alone, without ever installing R Studio. Yet, base R has a number of inconveniences and limitations, and, because it is simply a command line prompt, it can be a bit intimidating especially for beginners. R Studio is a user interface built on top of base R that makes using R feel more like using other programs that we are used to. It also makes doing a number of things in R much easier. In this class, we will only use R within R Studio.

In order to install R Studio, go to the website: https://rstudio.com/products/rstudio/download/

Scroll down the page until you get to the download selection. Choose R Studio desktop version.



The website will make a download recommendation for your particular system. Click on the blue download button to download the installation files. Notice that step 1 is to install base R. We have just done this so there is no need to do it again. Remember that R Studio is built on top of base R. You cannot run R Studio without base R.



Once you have downloaded the file, go to your downloads folder, or wherever you have set downloads to go, and run the .exe file to install the software. You may be prompted to give R Studio permission to install itself, click yes. You may also be asked if you want to create a desktop icon, this is up to you but may be helpful in finding R Studio in the future.

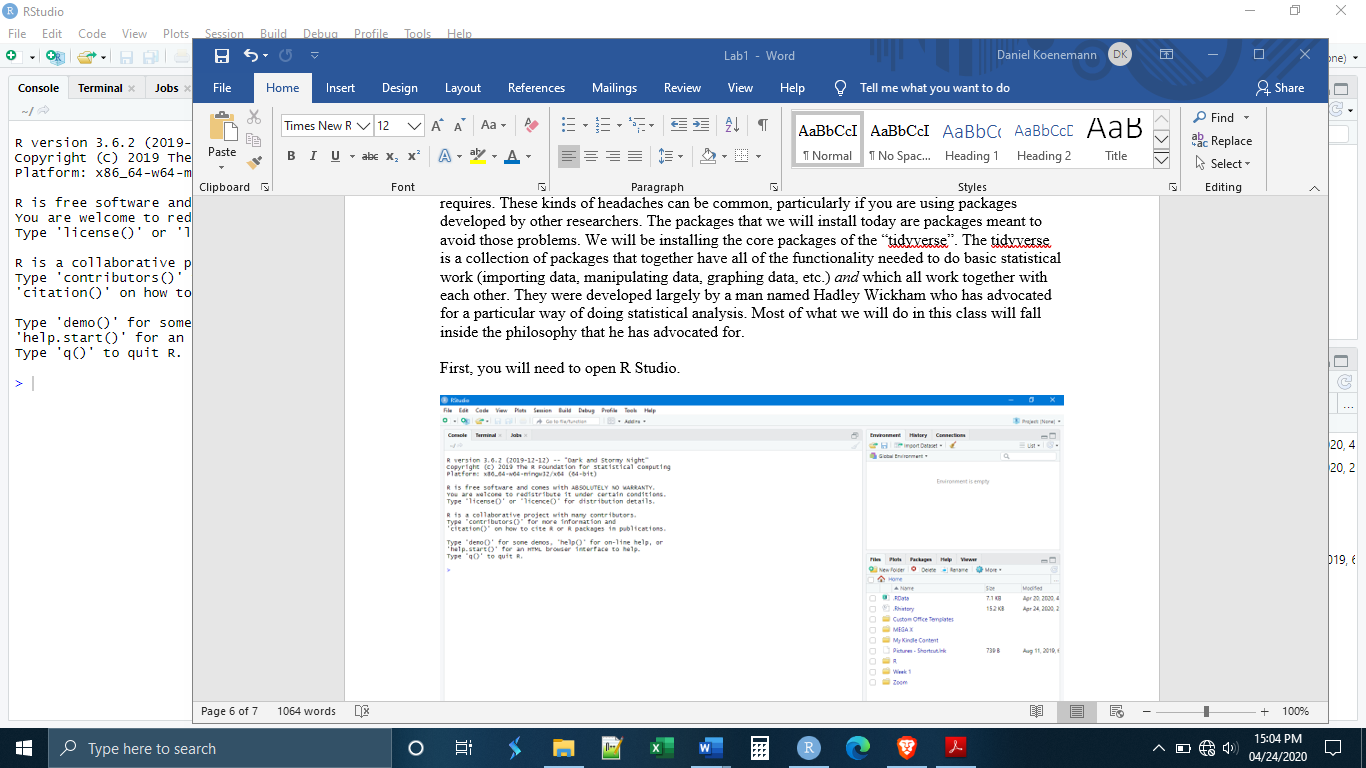
This will install R Studio on your computer.

*Installing Packages*

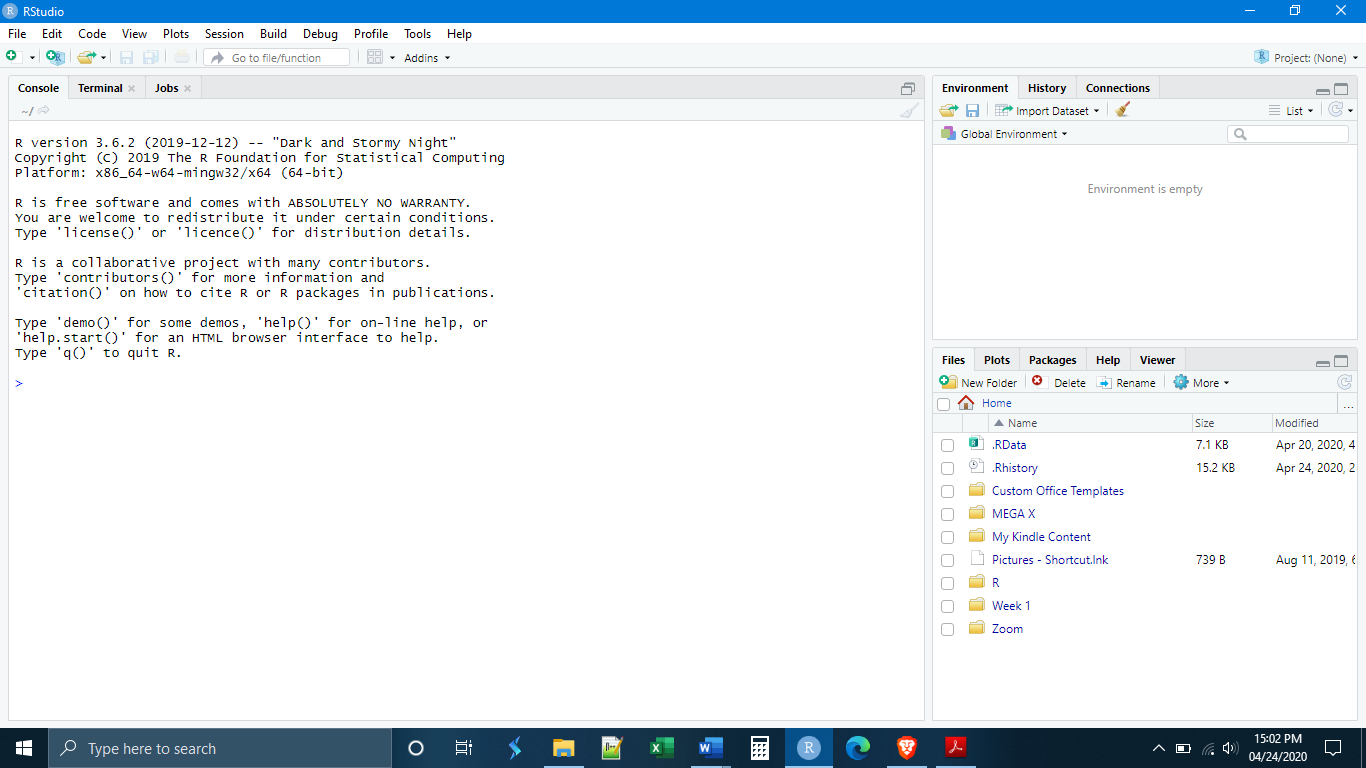
As stated above, packages greatly expand the functionality of R. There are more than 15,000 R packages that you can download and use, and there are more developed every day. Because R is free and open source, packages can be written by ordinary researchers in order to perform tasks that they perform frequently. Some researchers make their packages publicly available to download and use through secure repositories like GitHub. In many cases, packages have been approved by the R Project and can be downloaded and installed directly within R.

As you can imagine, because the majority of the R packages are written by different people, most of whom do not know each other, not all of the R packages work well together. You might want to use one package and then take the output and feed it into another package, only to find out that the first package does not output in the format that the second package requires. These kinds of headaches can be common, particularly if you are using packages developed by other researchers. The packages that we will install today are packages meant to avoid those problems. We will be installing the core packages of the “tidyverse”. The tidyverse is a collection of packages that have all of the functionality needed to do basic statistical work (importing data, manipulating data, graphing data, etc.) *and* which all work together with each other. They were developed largely by a man named Hadley Wickham who has advocated for a particular way of doing statistical analysis. Most of what we will do in this class will fall inside the philosophy that he has advocated for.

To install the packages, you will first need to open R Studio. If you made a desktop icon, you can open R Studio like any other program by double-clicking the desktop icon. If you did not make a desktop icon you will need to find where your computer has installed it. It should have been installed with your other programs. In Windows, you can try the search bar if you cannot find it. On a Mac you can search through the “Finder” icon.



Once opened, R Studio will look something like this:



We will have a full tour of the R Studio layout later, for now I simply want to install the packages. You will see that most of the screen is taken up by a large panel on the left side. This is the console, where commands can be entered. It is here that we will install our packages. To do this, execute the following lines of code. Each line needs to be typed (or copy-pasted) into the console, then press the “enter” key on your keyboard, then wait for the package to be installed. Each line must be done one at a time.

*install.packages(“rmarkdown”)*

*install.packages(“tidyverse”)*

*install.packages(“magrittr”)*

*install.packages(“data.table”)*

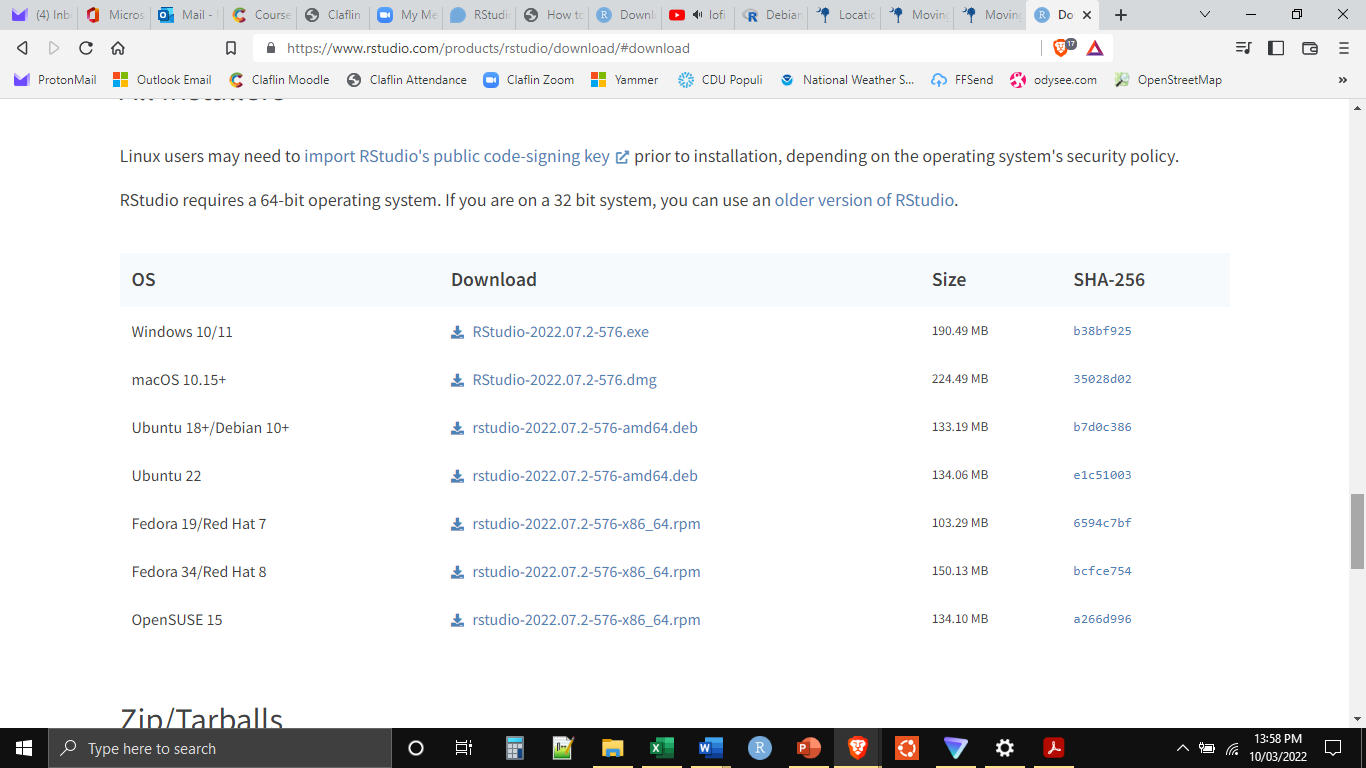
**Installing R and R Studio on a Chromebook**

The Linux terminal will need to be up and running in order to install R and R Studio. The first step is to install the base version of R. Open the Linux terminal and type the following command:

*sudo apt-get install r-base r-base-dev libnss3*

This should install R and one of the core “dependencies” (other files and programs Linux needs in order to install R). If Linux flags more dependencies, ask me to help and we can figure out what other dependencies need to be installed.

The next step to is installing R Studio, which is installed on top of base R. First, go to the R Studio website and download R Studio (<https://www.rstudio.com/products/rstudio/download/#download>). Scroll down, and download the version that will work with Debian Linux (which is what you are running).



The downloaded file will end with .deb (Debian package). Copy and paste this into the home folder in your Linux file system app, which should appear as one of the folders in your system. Then go back to the terminal and run the following code:

*sudo apt-get install ./rstudio-2022.07.2-576-amd64.deb*

This should install RStudio. Moreover, it should appear with your other icons as a “Linux app”. It is easiest just to open and run R Studio from this icon. Before doing anything else, delete the “rstudio-2022.07.2-576-amd64.deb” file, as we no longer need it. You can do this manually or by typing the following into the Linux terminal:

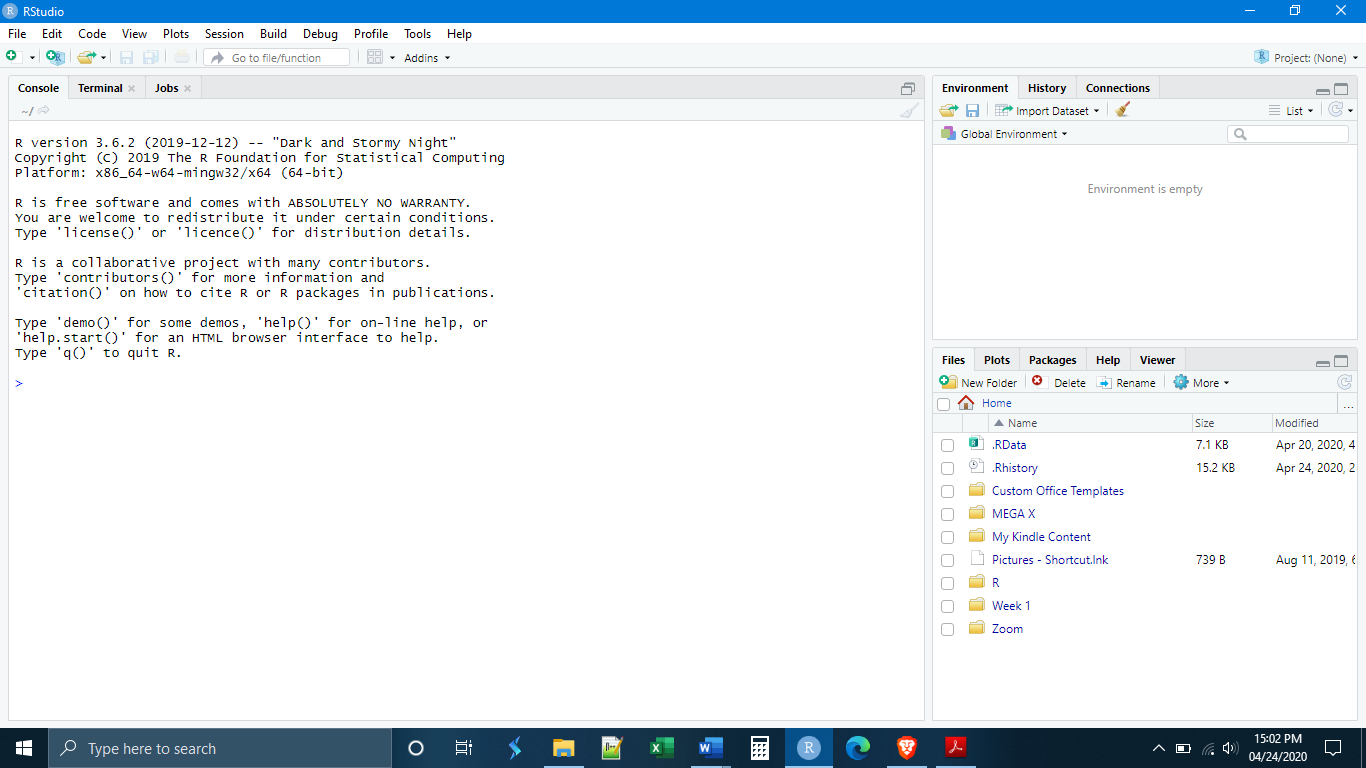
*sudo rm rstudio-2022.02.1-461-amd64.deb*

*Installing Packages*

As stated above, packages greatly expand the functionality of R. There are more than 15,000 R packages that you can download and use, and there are more developed every day. Because R is free and open source, packages can be written by ordinary researchers in order to perform tasks that they perform frequently. Some researchers make their packages publicly available to download and use through secure repositories like GitHub. In many cases, packages have been approved by the R Project and can be downloaded and installed directly within R.

As you can imagine, because the majority of the R packages are written by different people, most of whom do not know each other, not all of the R packages work well together. You might want to use one package and then take the output and feed it into another package, only to find out that the first package does not output in the format that the second package requires. These kinds of headaches can be common, particularly if you are using packages developed by other researchers. The packages that we will install today are packages meant to avoid those problems. We will be installing the core packages of the “tidyverse”. The tidyverse is a collection of packages that have all of the functionality needed to do basic statistical work (importing data, manipulating data, graphing data, etc.) *and* which all work together with each other. They were developed largely by a man named Hadley Wickham who has advocated for a particular way of doing statistical analysis. Most of what we will do in this class will fall inside the philosophy that he has advocated for.

To install the packages, you will first need to open R Studio. The easiest way is by clicking on the app icon. Once opened, R Studio will look something like this:



We will have a full tour of the R Studio layout later, for now I simply want to install the packages. You will see that most of the screen is taken up by a large panel on the left side. This is the console, where commands can be entered. It is here that we will install our packages. To do this, execute the following lines of code. Each line needs to be typed (or copy-pasted) into the console, then press the “enter” key on your keyboard, then wait for the package to be installed. Each line must be done one at a time.

*install.packages(“rmarkdown”)*

*install.packages(“tidyverse”)*

*install.packages(“magrittr”)*

*install.packages(“data.table”)*