How-to-R

Mir Henglin

Written for beginners.

As always, in further refinement and development.

What is this Document?

This is a (hopefully) helpful document written to help introduce you to some of the basic functionality of programming in R. It is written in **RMarkdown** using RStudio. This type of document facilitates **literate programming** where code and text are mingled to create cohesive stories and documentation. It is fairly sparse and short, providing a quick introduction to help familiarize the reader so that they can begin their own learning and research. Other helpful links for learning R include tryR, which will allow you to learn by typing in R commands online, The R Inferno, which will introduce the reader to some quirks of the R language, Stack Overflow, which serves as an online community help board for R, Advanced R, for users wanting to refine their code and learn more about the R language itself, and Impatient R, which is a short overall introduction to the language and includes links to other very helpful resources like quick-R and twoTorials. My choice of next step for a beginner would be R for Data Analysis by Hadley Wickham, which also introduces the reader to many important packages such as dplyr.

That being said, lets get to it.

What is R?

R is a *programming language*. Programming languages are interfaces that allow one to write out commands in human-understandable text that is later translated into the 1s and 0s that a computer understands and can act upon. R is special among programming languages in that it is focused on statistical analyses. It has many built in functions to run and manage every step of a data analysis process. Programming can be intimidating at first to those used to menus and buttons, but once you learn it offers immense power and flexibility. Below, I hope to guide the reader through installing and setting up R, as well as teach them how to make some simple graphs.

Installing R

There are two steps that I would recommend to creating an environment that makes it easy and productive to code in R. The first is to (obviously) install R. The second is to install RStudio. RStudio is an IDE, which is a program that interfaces with the R language and provides many useful tools and interfaces for the programmer. One of the best things about R is how powerful and comprehensive RStudio is, and how it facilitates easy use. RStudio also brings back many buttons and menus that might make working with R less intimidating.

Note, example images below images won't be 100% consistent with one another, but will show the main thrust of each point.

RStudio

When you open RStudio, you should see something like this:

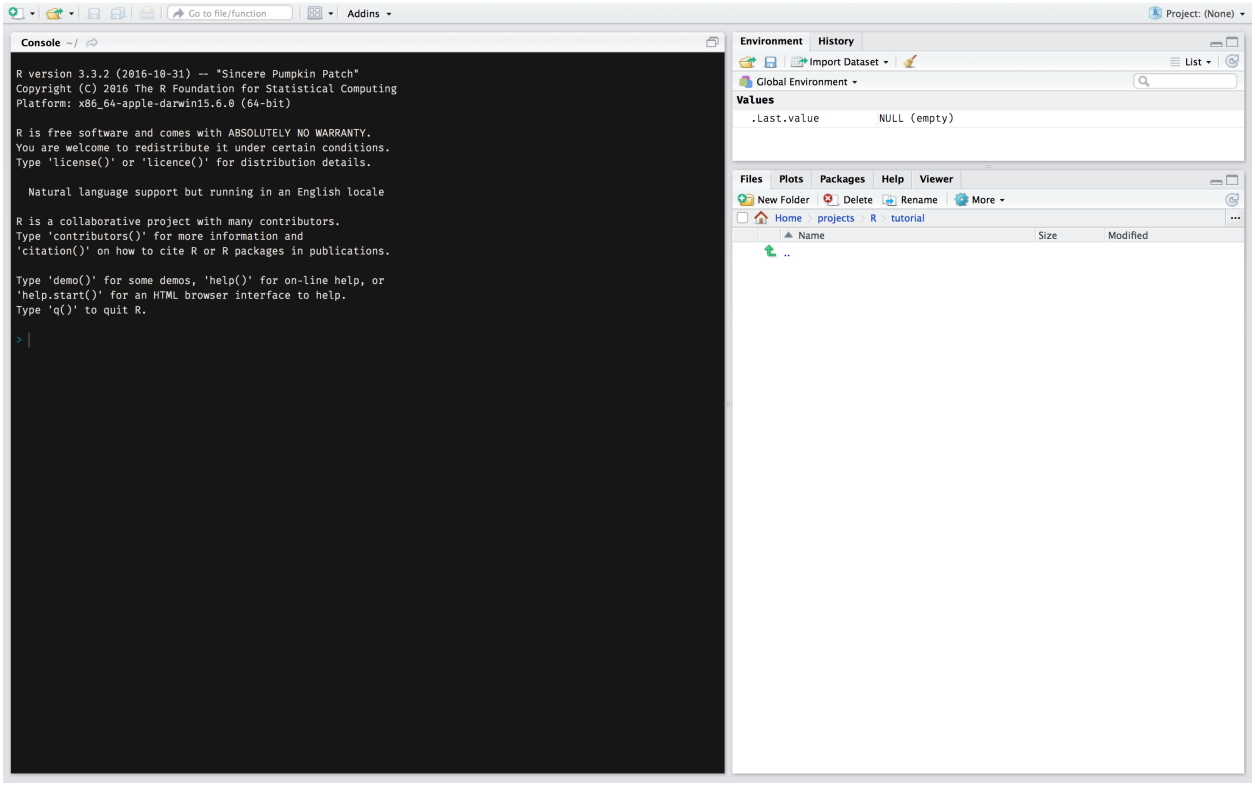


Figure 1: RStudio fresh open.

Note that some appearance and color options have been modified. To edit these settings, go to **Tools > Global Options > Appearence** and edit the settings there.

Panes

The pane on the left is the *Console* it is where commands to R are run and output displayed. The panes on the right carry additional information to help the programmer as they work. The top right pane shows the History and Environment tabs. The *Environment* tab helps you to keep track of all the objects and variables that exist in the current R working session, while the *History* tab keeps track of all commands submitted to the console.

The bottom right pane contains the File, Plots, Packages, Help, and Viewer tabs. The *Files* tab shows your folder directory, allowing you to navigate folders without leaving the programming environment The *Plots* tab serves a similar function, capturing plotting commands so that the programmer can view plots without leaving the programming environment. The *Packages* tab allows you to see what packages are available to load and to install new packages from the internet. Packages are basically bundles of code and data that others have written to share with the public. Installing and loading packages gives you access to the code within the package. The *Help* tab shows the help file for functions. To bring up the help file for a function, use a question mark in front of the function you need help with `?function-name`. The *Viewer* pane is for viewing web-like content without leaving the programming environment.

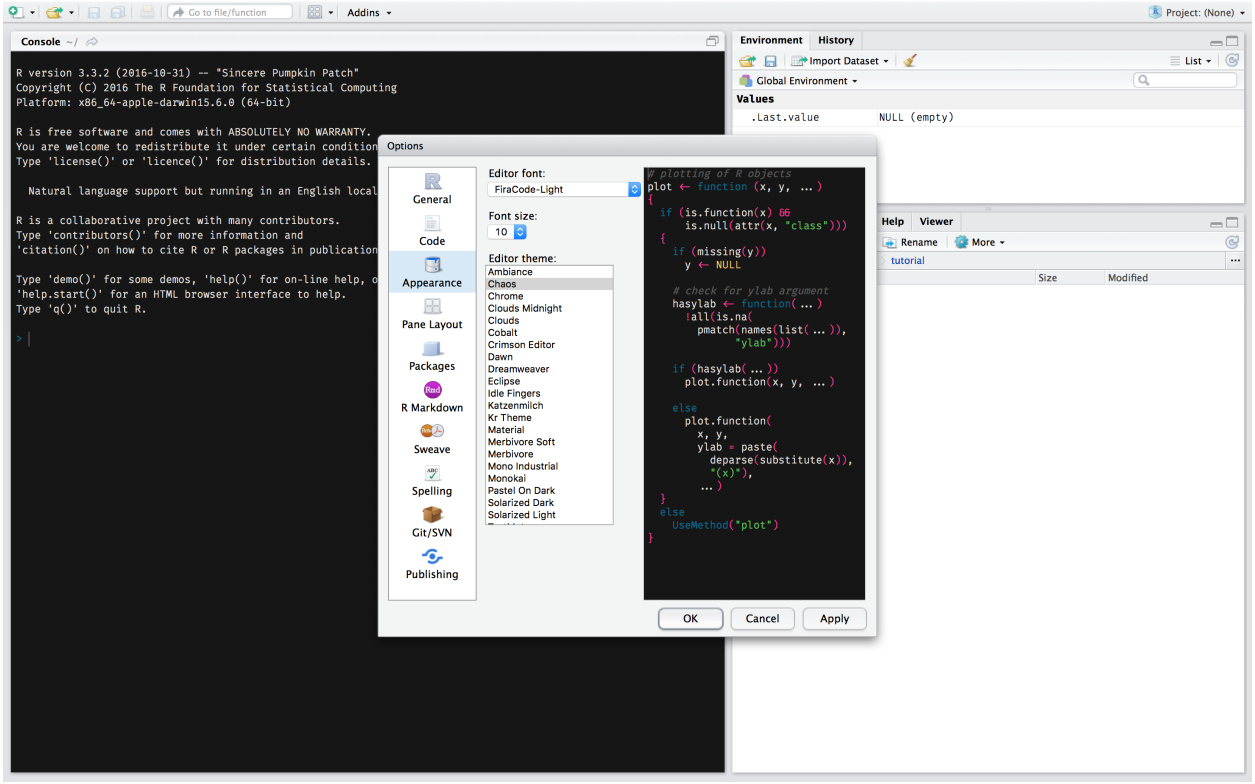


Figure 2: RStudio appearence settings.

First commands

Lets send a few commands to the console. After typing each command, we press **enter** to submit it.

We see that after each command, the results appears below it. Our first command was to type `1 + 1`. After pressing enter, the results is evaluated to `2`. The command below that, `rnorm` is our first **function**. Functions are operations whose results depend on their **arguments**. The function `rnorm` generates random draws from a normal distribution. We pass it the argument `10` to indicate that we want 10 random draws. By default, the draws will be from a standard normal.

Getting Help

To learn more about the rnorm function, we can type in a `?` before the name of the function. Lets learn more about the rnorm function.

We see on the right that the help pane now displays the documentation for the rnorm function. We can see all the arguments that we can pass, and how each of them affect the output of the function. Lets say that instead of 10 draws from a standard normal, we want 10 draws from a normal distribution with mean 10 and standard deviation 5. The way to do that would be.

```
rnorm(10,
      mean = 10,
      sd = 5)
```

```
##  [1]  1.989292 13.021285 19.147881 11.822084 12.510936  9.659172 16.506183
##  [8] 14.377859  3.395184 12.854929
```

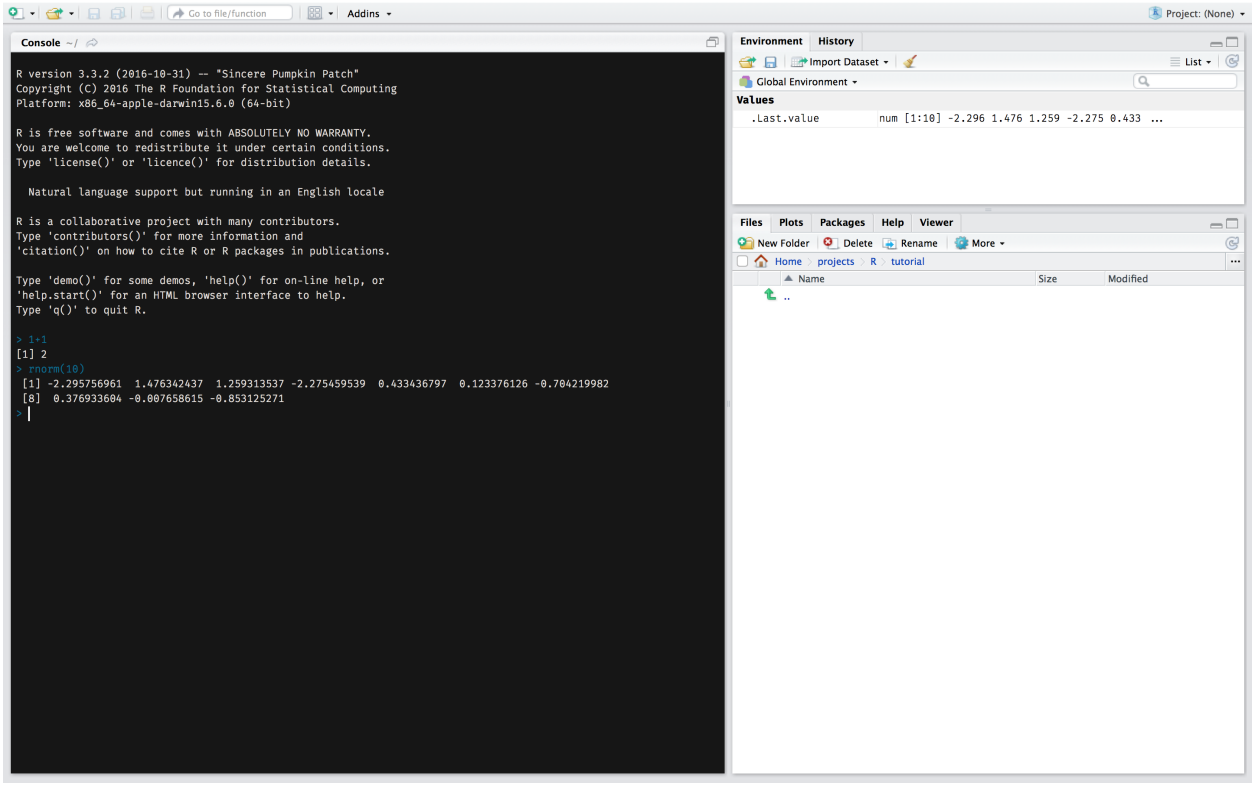


Figure 3: First Commands.

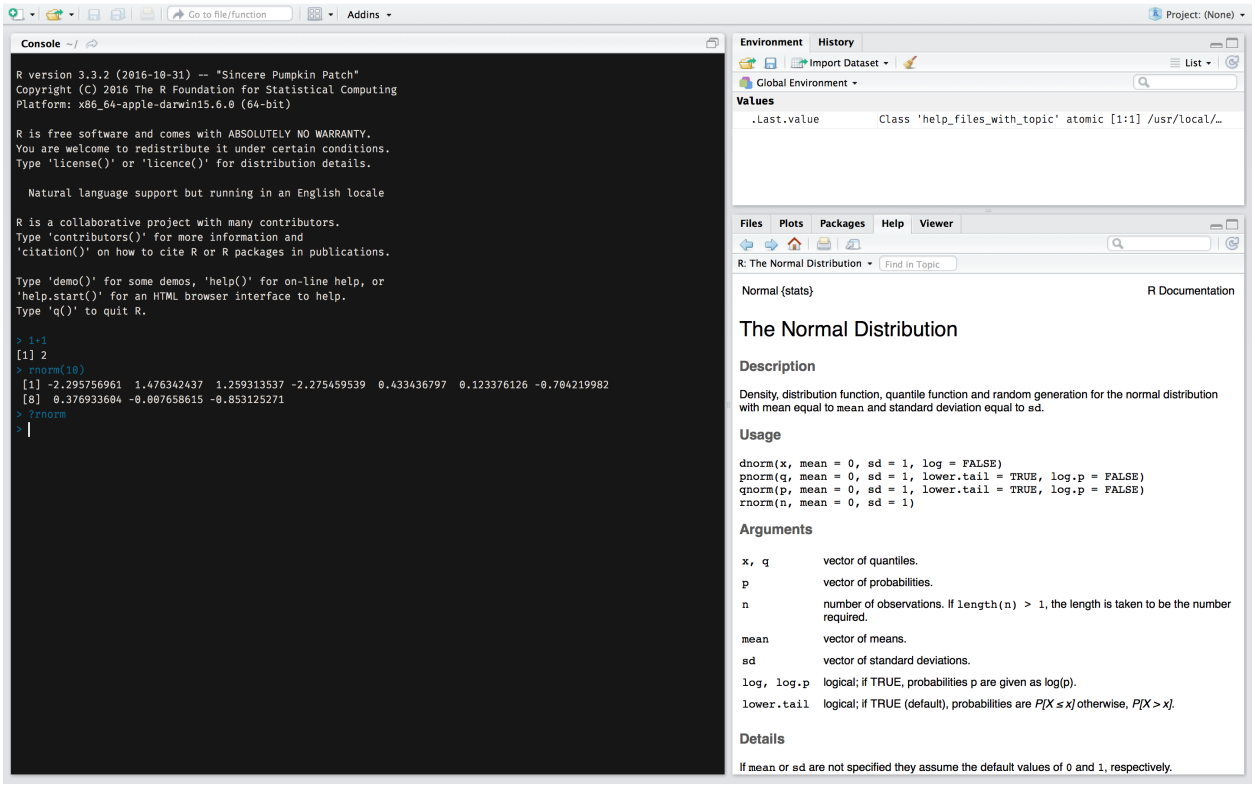


Figure 4: rnorm help.

You may notice that instead of attaching a screenshot, I have evaluated this code inline. I will occasionally do this instead of attaching a screen shot if the focus is mostly on code and output, instead of something happening in the RStudio program window itself. (The ability to include code and text in a nicely formatted, easily readable way is part of the literate programming paradigm mentioned earlier.)

It is also possible to browse the documentation by **Clicking the 'Home' Icon in the Help pane > Click 'Packages'** which will show descriptions of all installed packages. These links can be clicked to find the documentation for the functions contained in each package. You can also use the built in help search-engine by clicking **Search Engine & Keywords** under the home menu. Help can also be found online, often by Goggling a query of the form *R 'function-name' 'package' 'short-description-of-problem'*.

Installing and Loading Packages

Probably the best thing about R is its rich and extensive ecosystem of packages. Packages are user developed functions that are meant to be shared and distributed among all R users. These packages often target a specific problem, analysis method, or workflow. There is an active community of R developers releasing and updating packages constantly. If you have a need, there is probably a package that has been developed to help make it easy.

Packages can be installed from the packages pane. Click *install* to bring up a menu. Type the package name into the search bar, and click install. *readr* is a package for importing and exporting data by Haley Wickham. While installing a package, some diagnostic text should appear in the console, and a confirmation message should appear at the end to let you know that installation was successful. The package can either be loaded by clicking the check box next to its name in the package menu, or by typing `library(package-name)` into the console.

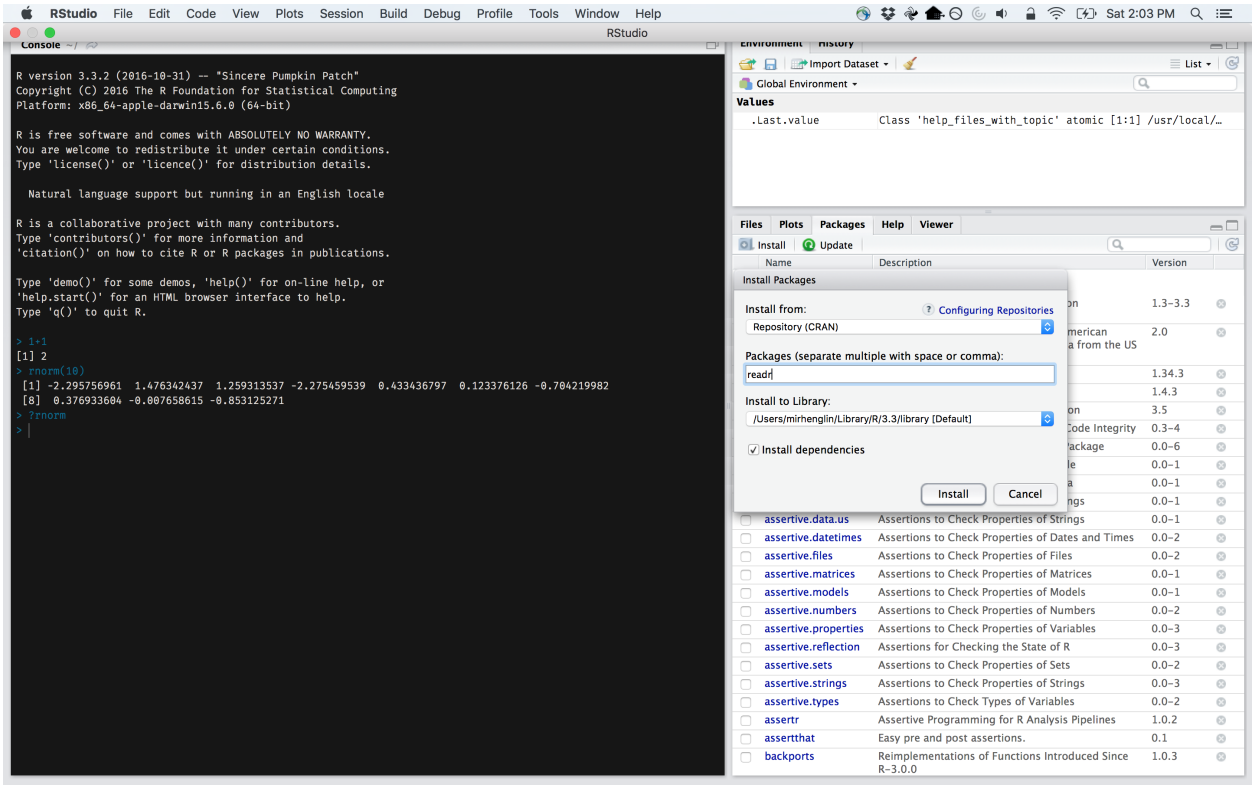


Figure 5: installing readr.

```
library(readr)
```

Loading data

The first step to any data analysis is to get the dataset loaded into R. It is almost always useful to to set the working directory near to where the dataset is stored. When the RStudio looks for files to load or browse, it starts in the working directory. You can set your working directory by clicking **Session > Set Working Directory > Choose Directory**. You will notice that this generates code in the console. It is possible to type in that same code and get the same results without clicking any buttons. Many buttons are simply point-and-click interfaces to R commands. This may also be auto managed through the use of **Projects**.

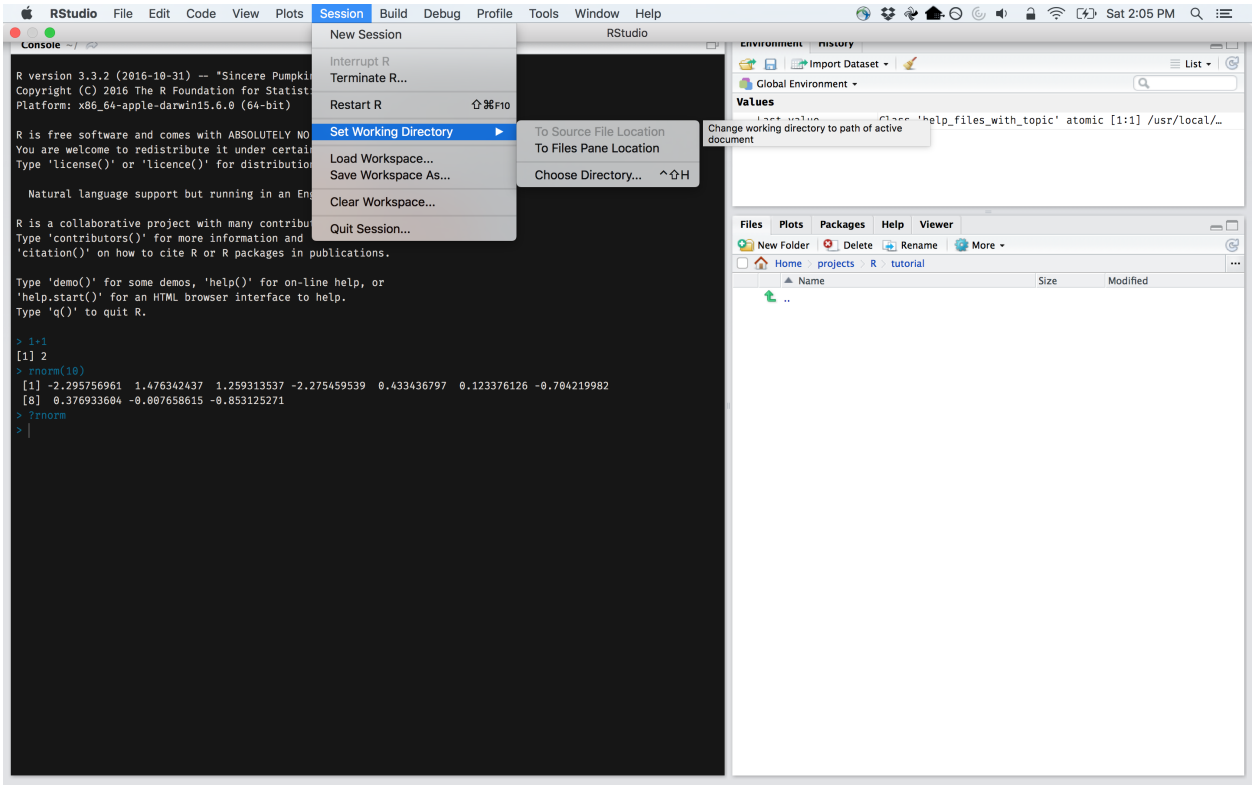


Figure 6: set wd.

Lets say that we have set the working directory to a folder that contains some data. We can see in the folder pane that there is a dataset located directly in the directory, and a folder with come data in it as well.

We can load the data by first loading the `readr` library, and using it to read in our data. We see that if we try to use the `read_csv` function before loading the library that an error occurs. Because we have changed our working directory, we don't need to type in the full path of the dataset (though that can work.) We need only type in the name of the file.

It order to play with the dataset, we need to store it as a variable. in this case, we will use the `<-` symbol to assign. The value on the right of the arrow is given the name on the left of the arrow. We can then print the value of that variable by typing its name into the console.

If you want to view the data in an excel-like formula, one can use the `View` Function.

All of this can also be done from a point and click interface by clicking on the graph icon next to the data file. Accessing data stored in a folder is done by prepending the folder name before the dataset.

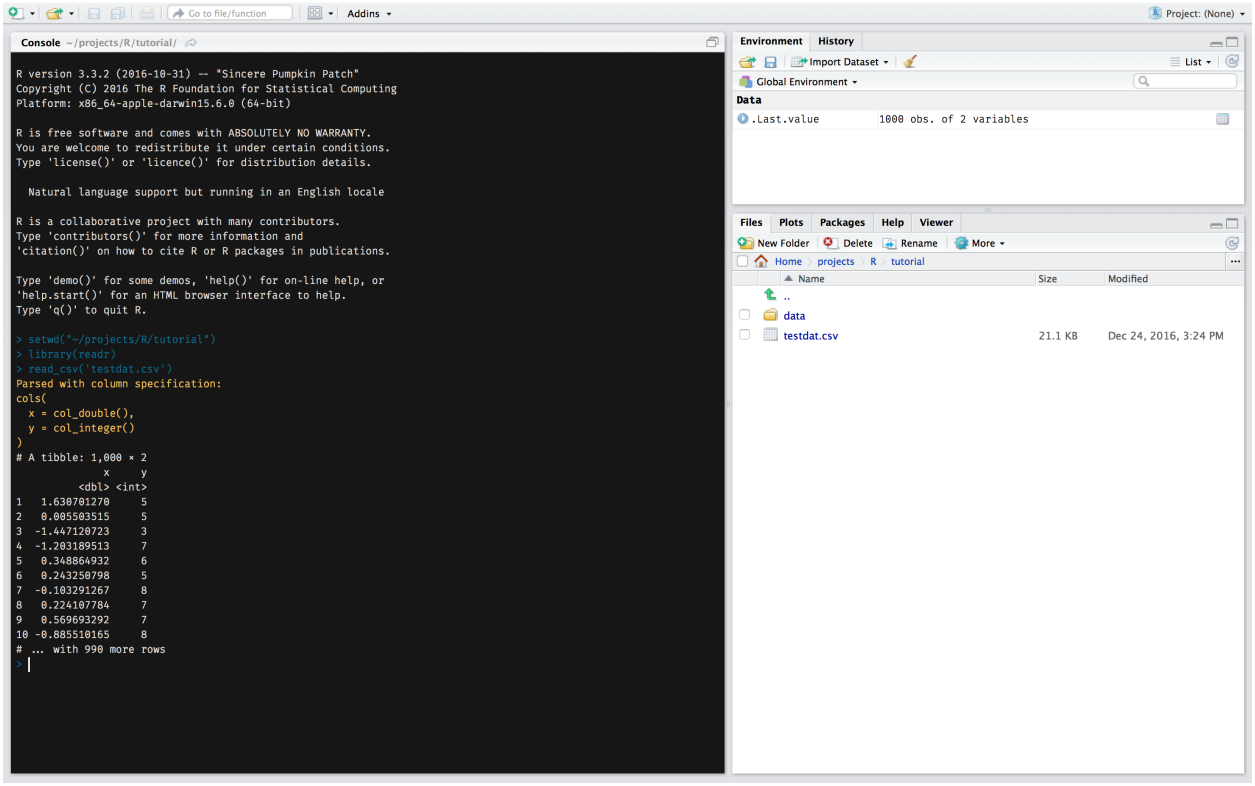


Figure 7: import.

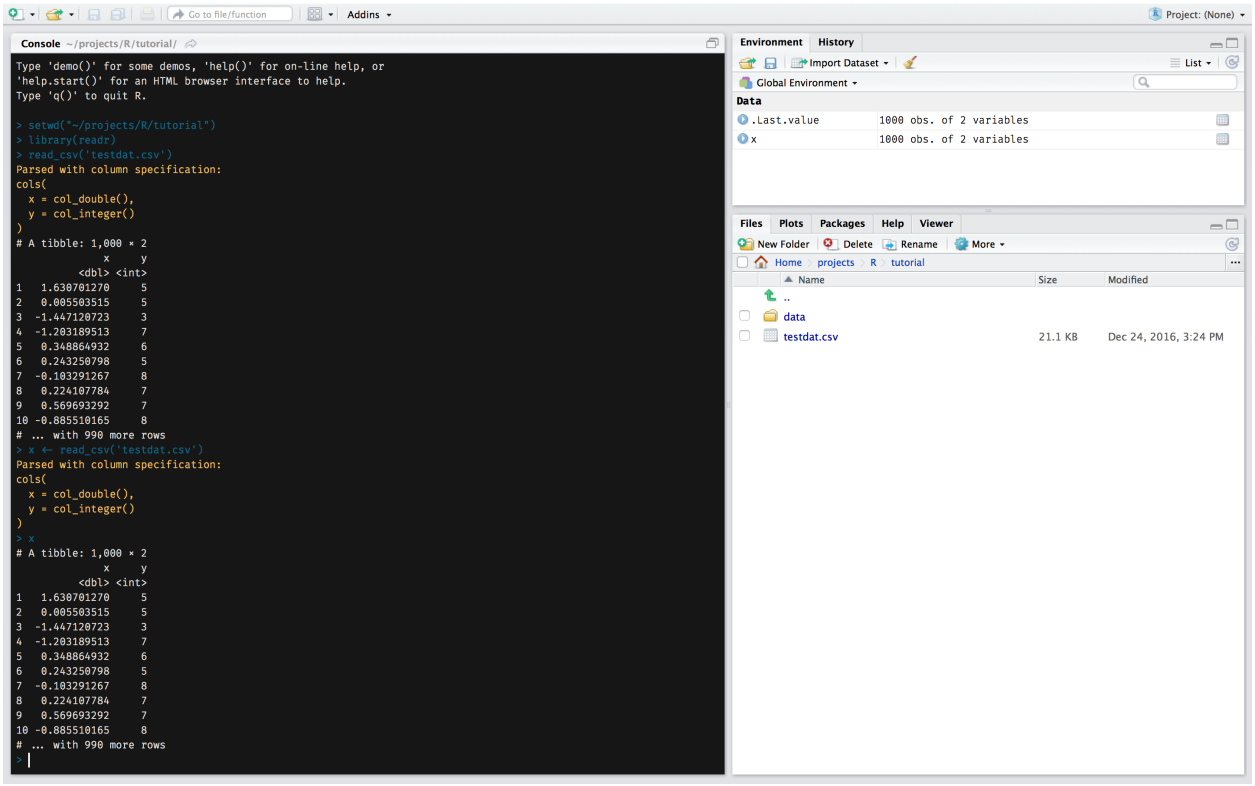


Figure 8: variable.

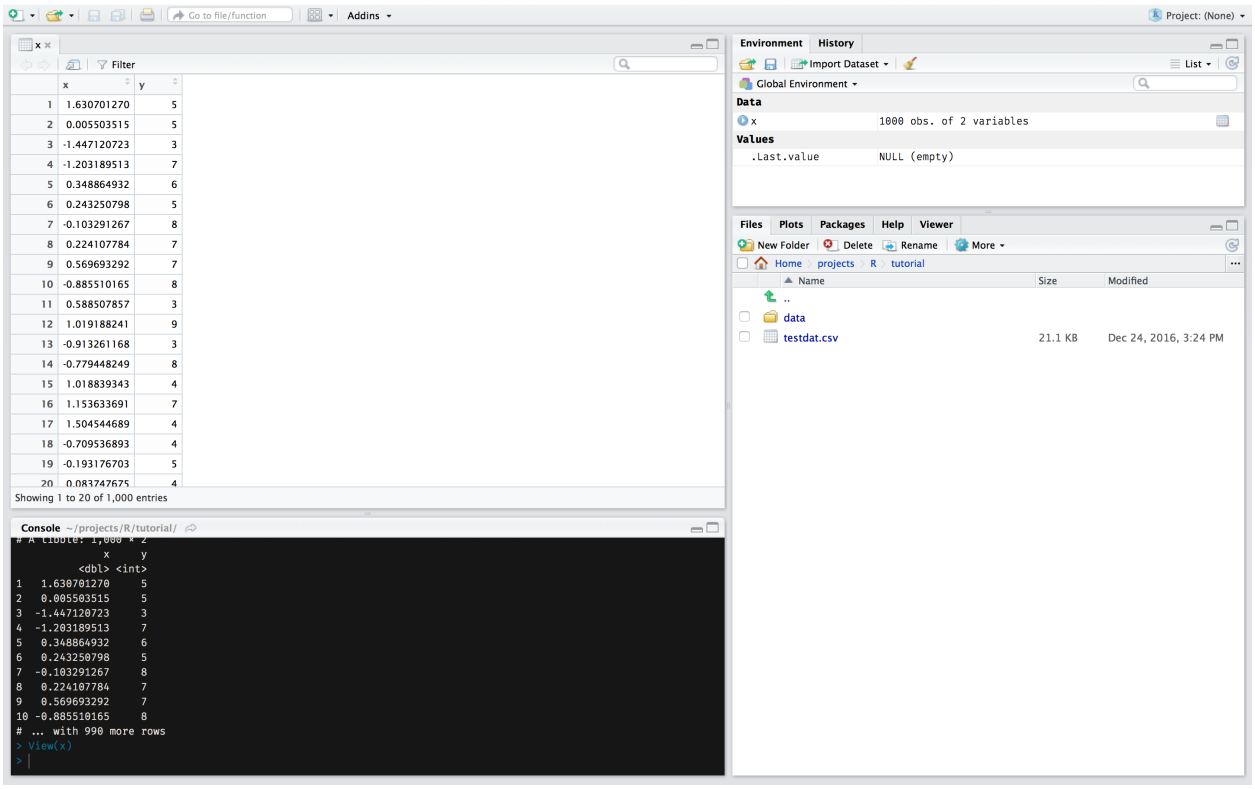
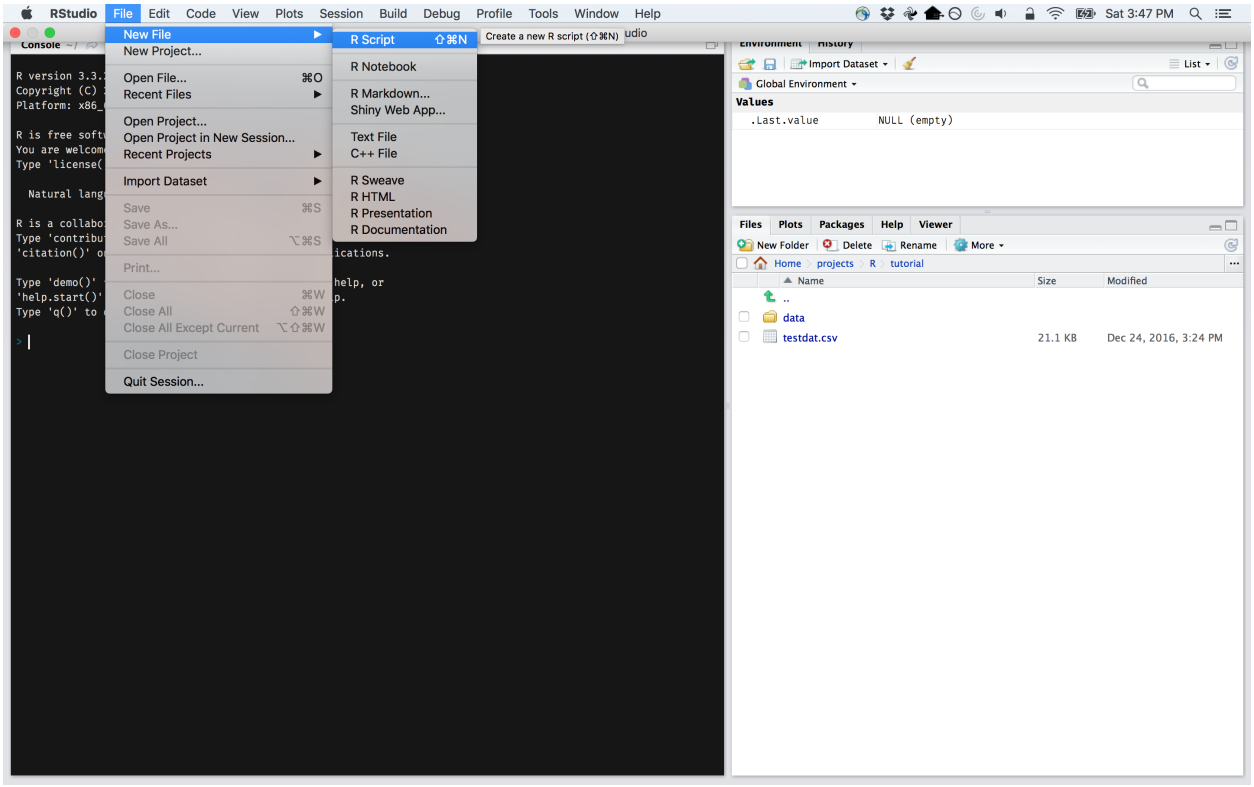


Figure 9: viewing data.

```
read_csv('data/testdat.csv')
```

Creating a file

Typing commands into the console is not the best way to code if you want to preserve, develop, and share the commands being typed in. That is because once they are typed in, they are gone. Instead, we will create an R text-file that we type our commands into, before sending that code to the console. They can be saved, moved, and organized similar to Word files. To create a new file, click **File > New File > R Script**



The Script should now appear in the top left corner, and the console pane is shifted to the bottom left.

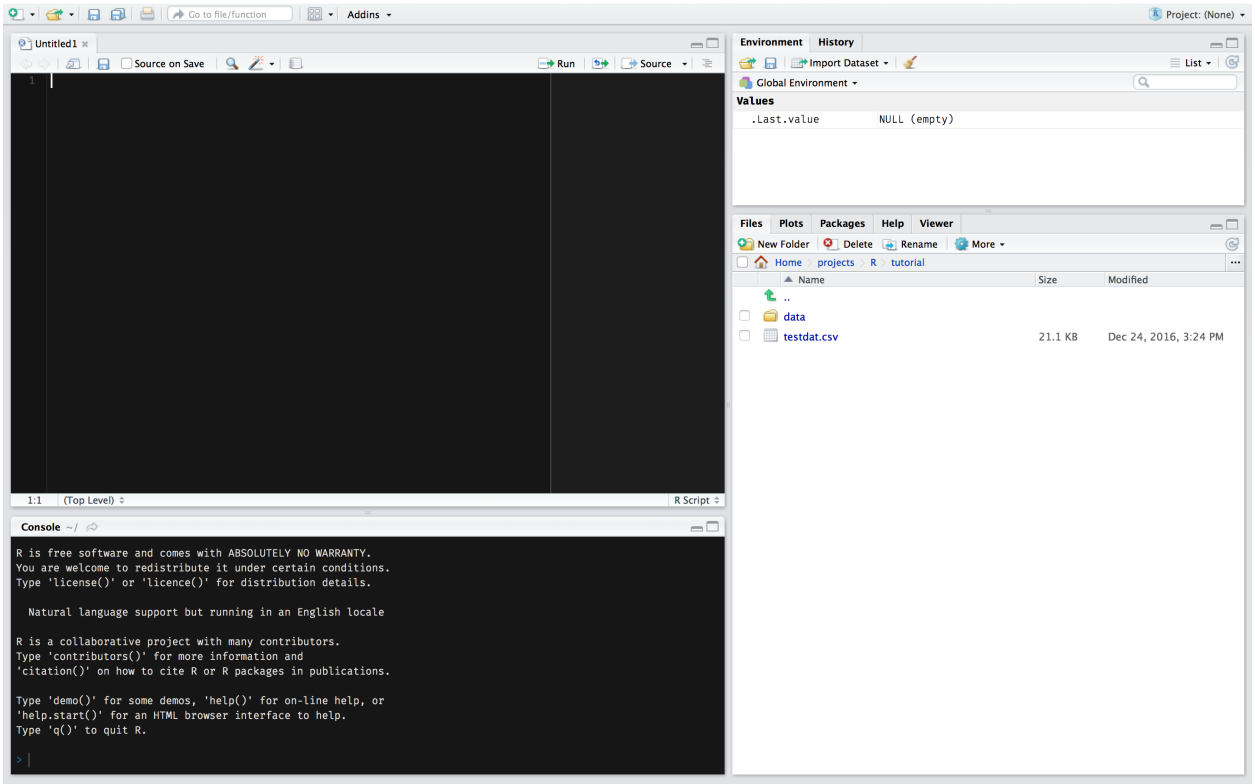
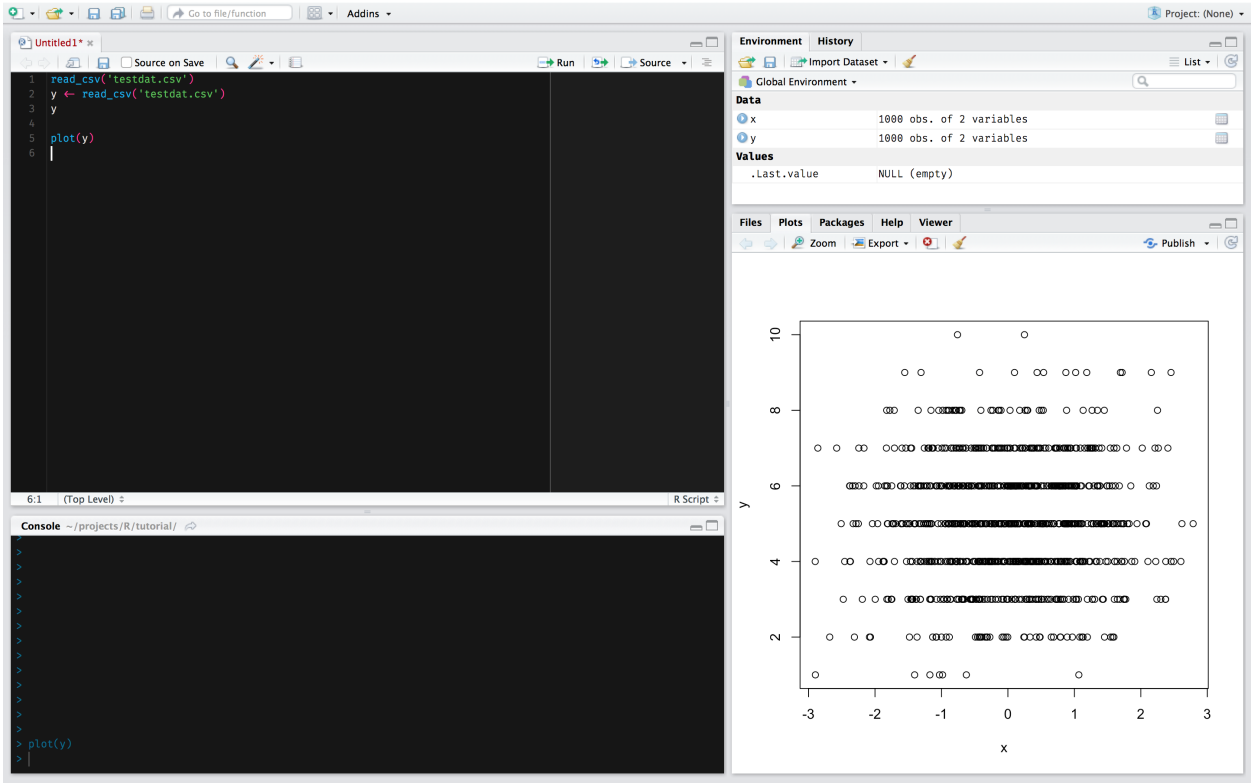


Figure 10: After creating new file.

Basic plot

We will be using the script to do some basic plotting. Lets assign our dataset to a variable y and plot it.

```
library(readr)
y <- read_csv('data/testdat.csv')
plot(y)
```



We see that the plot has appeared in the plot tab in the bottom right. We were able to plot the dataset by simply calling `plot` on it because there were only two columns of numbers. We can also plot two arbitrary sequences of numbers by passing in two equal length series of numbers.

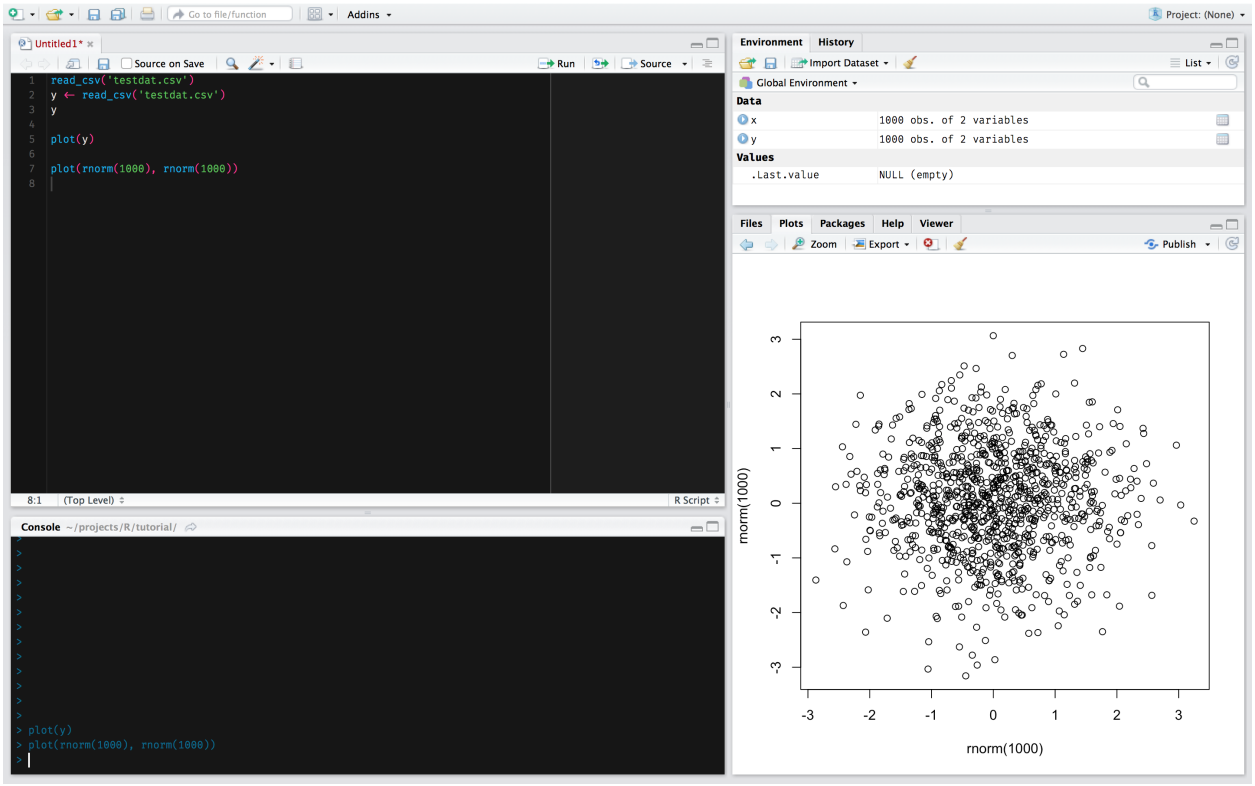


Figure 11: rnorm plot.