Tutorial on R and R Studio (Part I)

Le Wang

Roadmap

1. Why do we need computer programs?
2. What is `R`?
3. Why `R`? Three useful features
4. Why R studio?
5. A Short Tour of R Studio
6. Script file

Why do we need computer programs?

Let's look at a dataset. see if you can discern any pattern

```
##  [1]  5 20  4 48 79 60  6 42 91  4 26 75 46 18 39 81  2 92 61 69 18 62  4
## [24]  4 11 57 16 61 13 92 59 51 19 17 39 88 71 68 40  5 87 75 98 10 79 98
## [47] 82 53 90  5
```

Imagine that you have millions or even billions of these numbers and more dimensions.

Why do we need computer programs?

To emphasize the important role that quantitative social science research can play in today's data-rich society. To make contributions to this society through data-driven discovery, we must learn how to analyze data, interpret the resulting empirical evidence, and communicate our findings to others.

To start our journey, we presented a brief introduction to `R`, which is a powerful programming language for data analysis.

What is R?

R is a **programming language** used for statistical analysis and graphics. It is based on S-plus, which itself was based on S, a programming language originally developed by AT&T.

If you are interested in the history (and future) of R, please refer to an article **R : Past and Future History** by Ross Ihaka.

Why R?

Three useful features:

1. Object-Oriented Programming
2. Functional Programming
3. Polymorphic

R: Object-Oriented Programming

Object-oriented programming in R is a style of programming that is based on defining classes, and creating and manipulating objects of those classes.

Objects are instances of classes, which also determine their types.

Unlike many other statistical softwares such as SAS and SPSS, R will not spit out a mountain of output on the screen.

Instead, R returns an **object** containing all the results. You, as an user, have the flexibility to choose which result to be extracted or reported.

R: Functional Programming

Advanced R by Wickham here

R, at its heart, is a functional programming (FP) language. This means that it provides many tools for the creation and manipulation of functions.

In particular, R has what's known as first class functions (treated as objects). You can do anything with functions that you can do with vectors:

Vectoerized Programming

This feature allows us to write faster yet more compact code. For example, a common theme in R programming is **avoidance of explicit iteration**. Unlike many other statistical softwares, explicit loops are discouraged.

Instead, R provides some functions that could allow us to express iterative behavior implicitly.

R: Polymorphic

R is also *polymorphic*, which means that a single function can be applied to different types of inputs (much more user friendly).

Such a function is called a *generic function* (If you are a C++ programmer, you have seen a similar concept in *virtual functions*).

R: Polymorphic (Example)

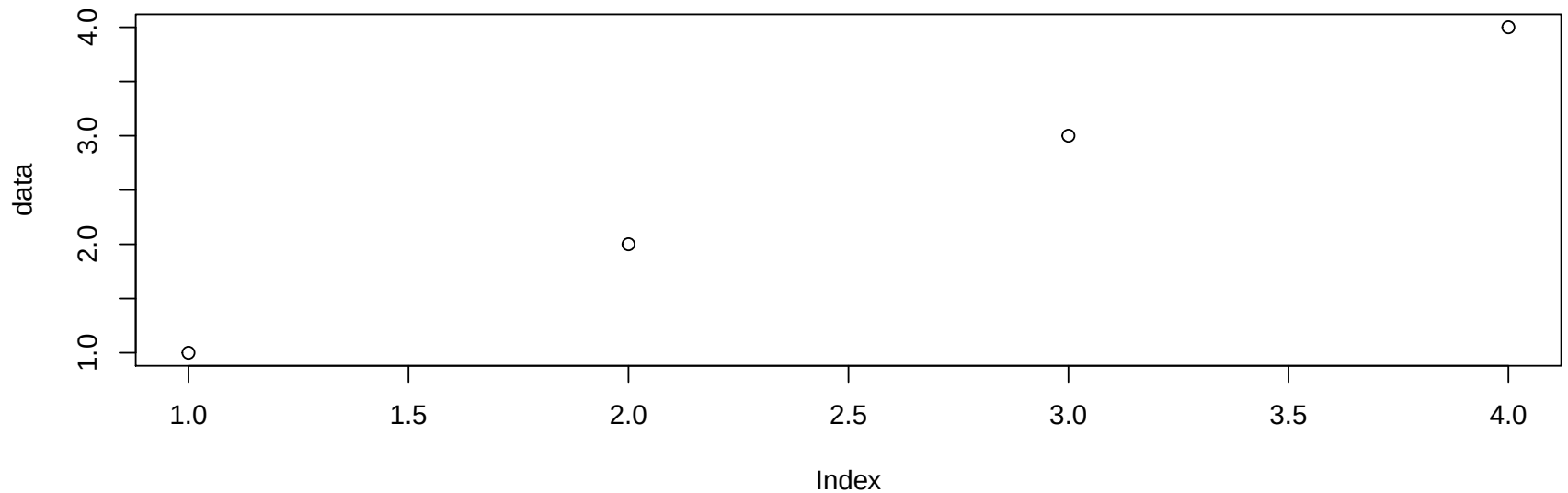
Lets look at one example `plot()`

1. Plot a vector of numbers
2. Plot some model results

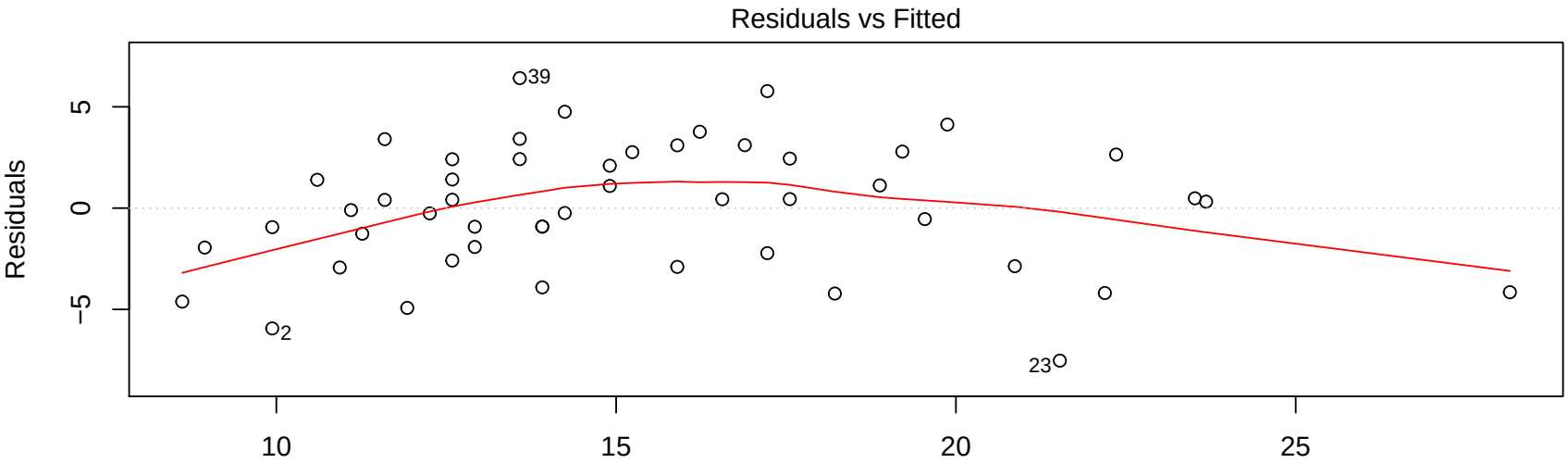
No matter which purpose, we use the same function.

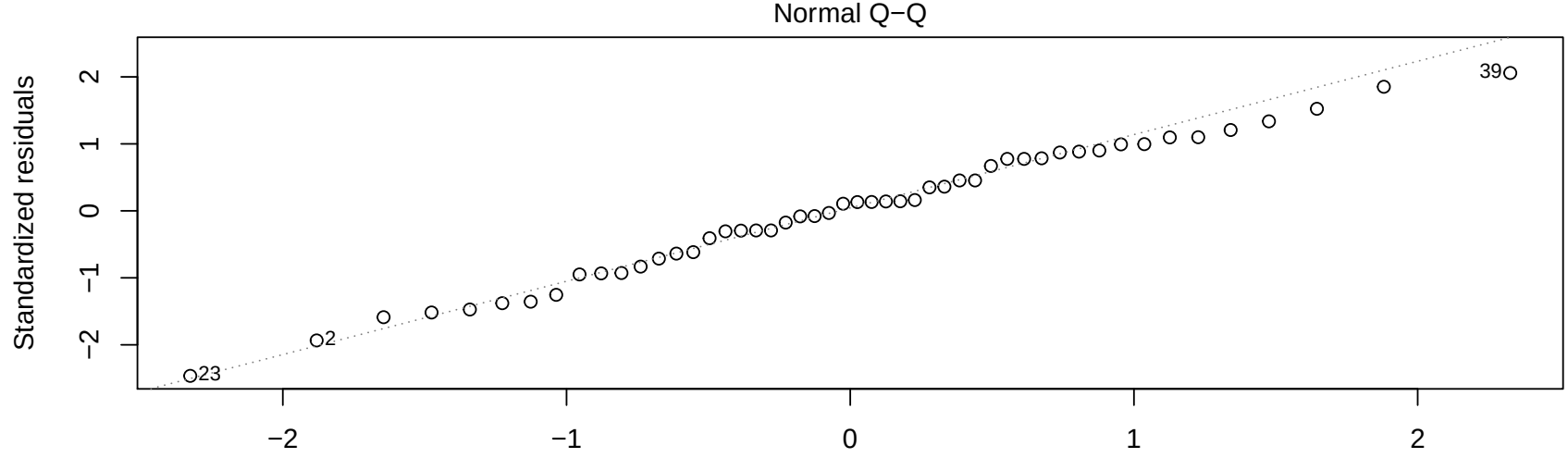
R: Polymorphic (Example)

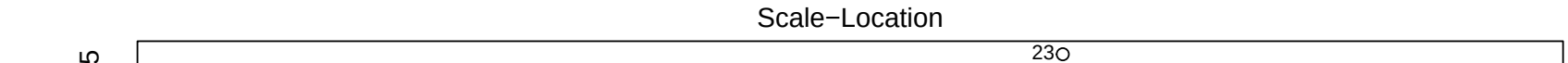
```
data<-c(1,2,3,4)
plot(data)
```



```r
# Regression Analysis
results<-lm(speed ~ dist,data=cars)
plot(results)
```







Why R Studio?

`R` Interface is **ugly**!

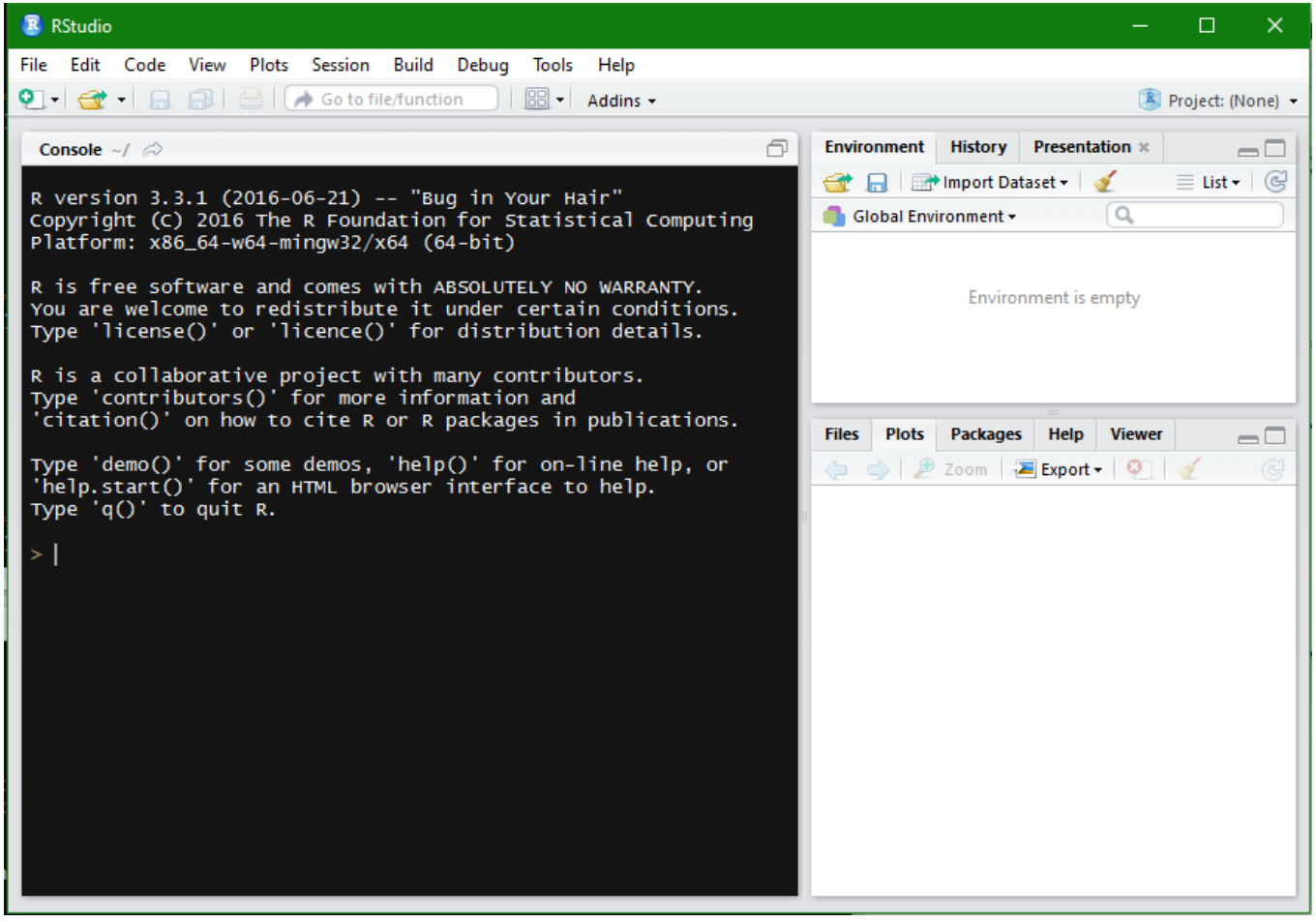
Many students in this class are much more familiar with Windows operation system and have never been exposed programming before, so we will use R studio, one of the free GUIs that have been developed for `R`.

R studio should really be considered as *integrated development environments* (IDEs), since it is aimed more toward programming.

R Studio: A short tour

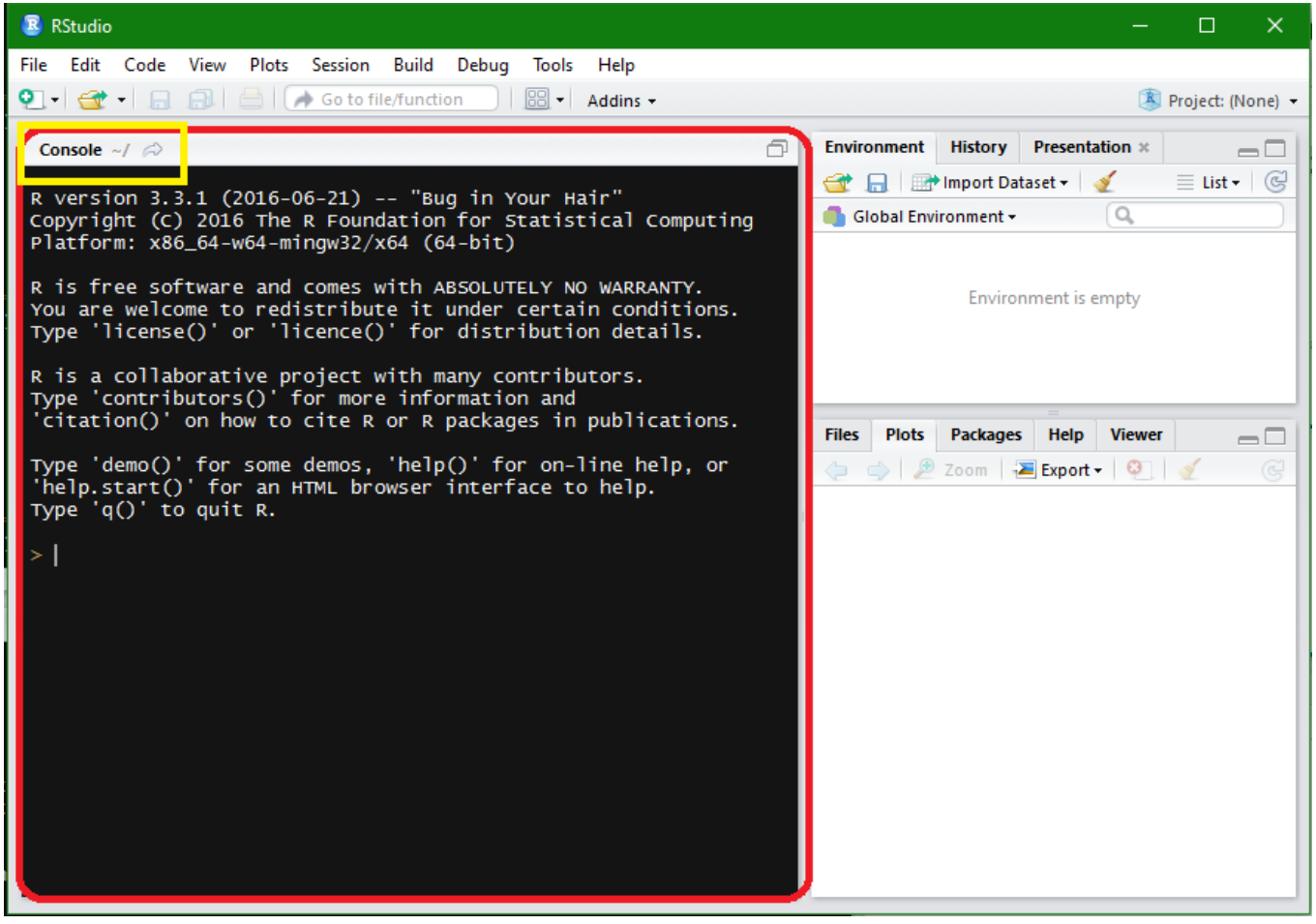
Initial Start

When you first (like very first time) open R studio you will see three panels.



R Studio: A short tour

Console



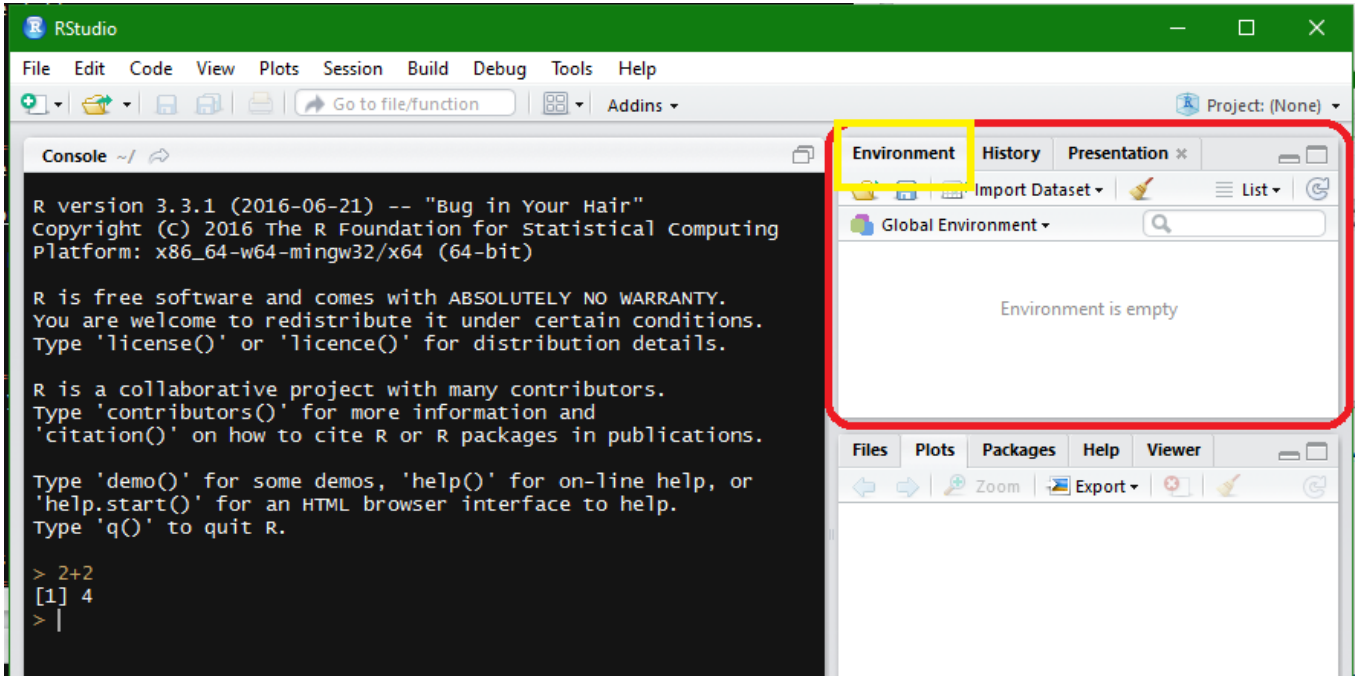
Console

1. Everytime you launch RStudio, it will have the same text at the top of the console telling you the version of R that you're running.
2. Below that information is the prompt, $>$. As its name suggests, this prompt is really a request, a request for a command.
3. Initially, interacting with R is all about typing commands and interpreting the output.
4. These commands and their syntax have evolved over decades (literally) and now provide what many users feel is a fairly natural way to access data and organize, describe, and invoke statistical computations.

The console is where you type commands and have them immediately performed.

R Studio: A short tour

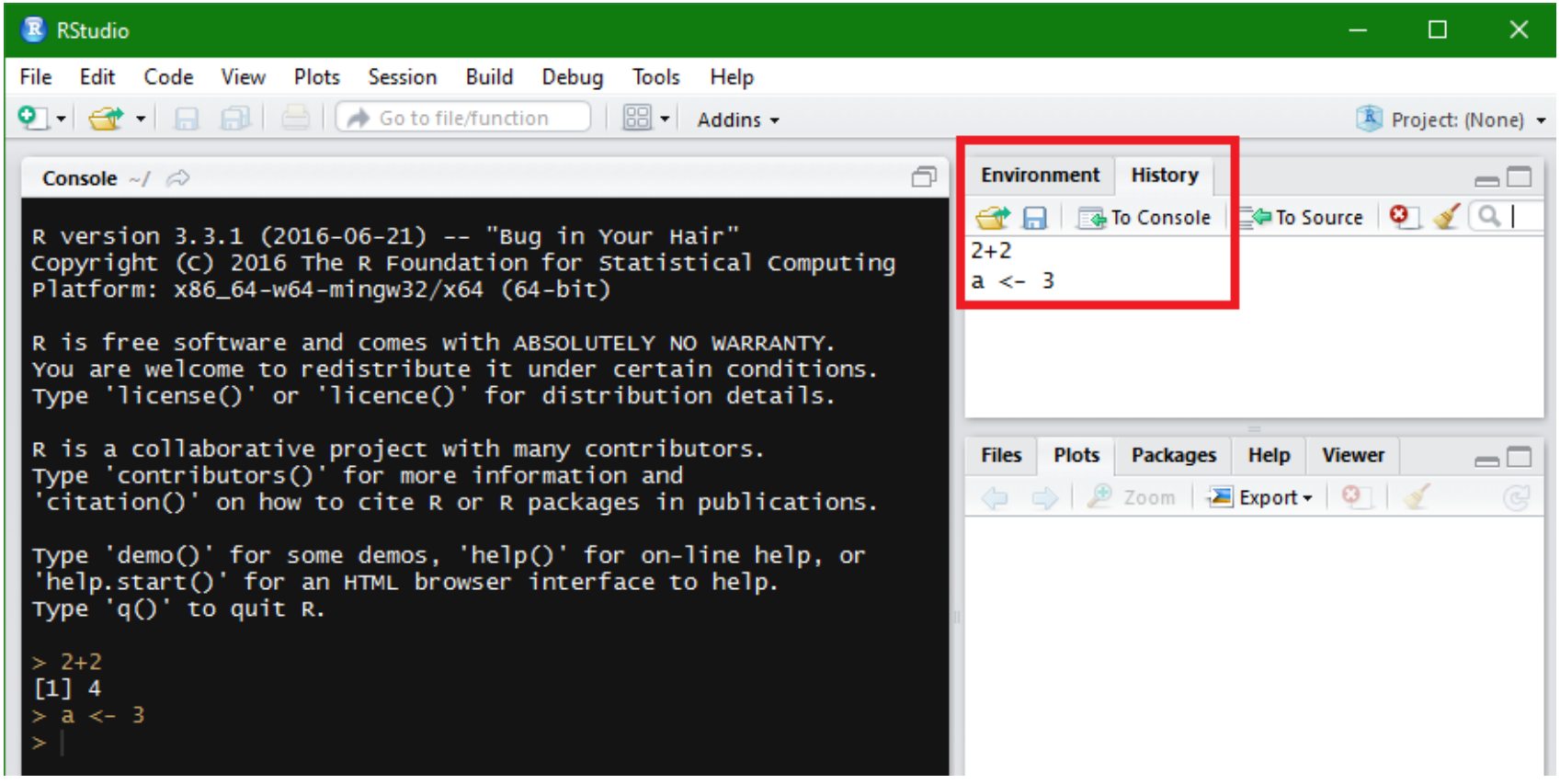
Environment The panel in the upper right contains your workspace (aka Environment)



1. This shows you a list of objects/variables that R has saved.
2. For example here a value of 3 has been assigned to the object a.

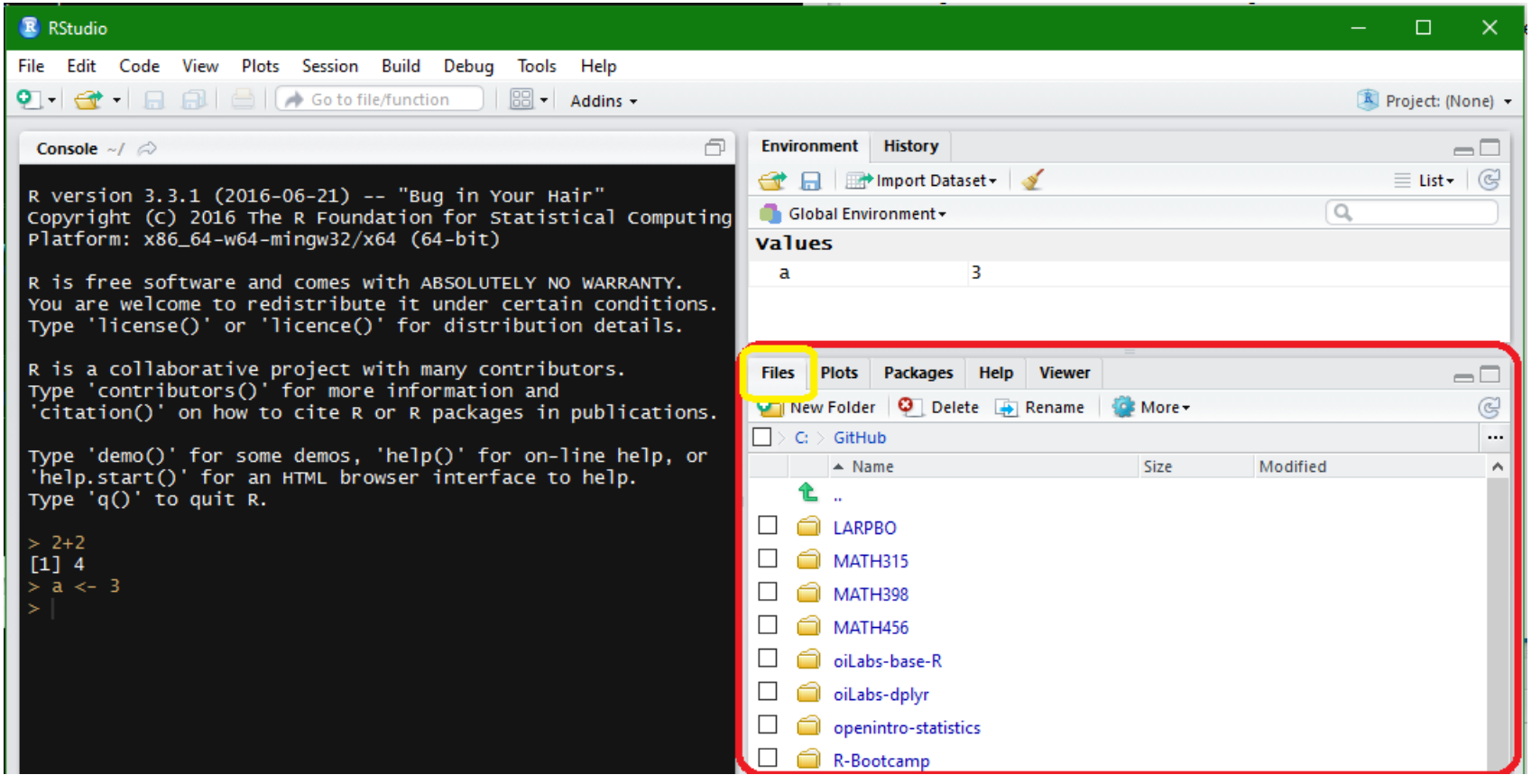
R Studio: A short tour

History Up here there is an additional tab to see the history of the commands that you've previously entered.



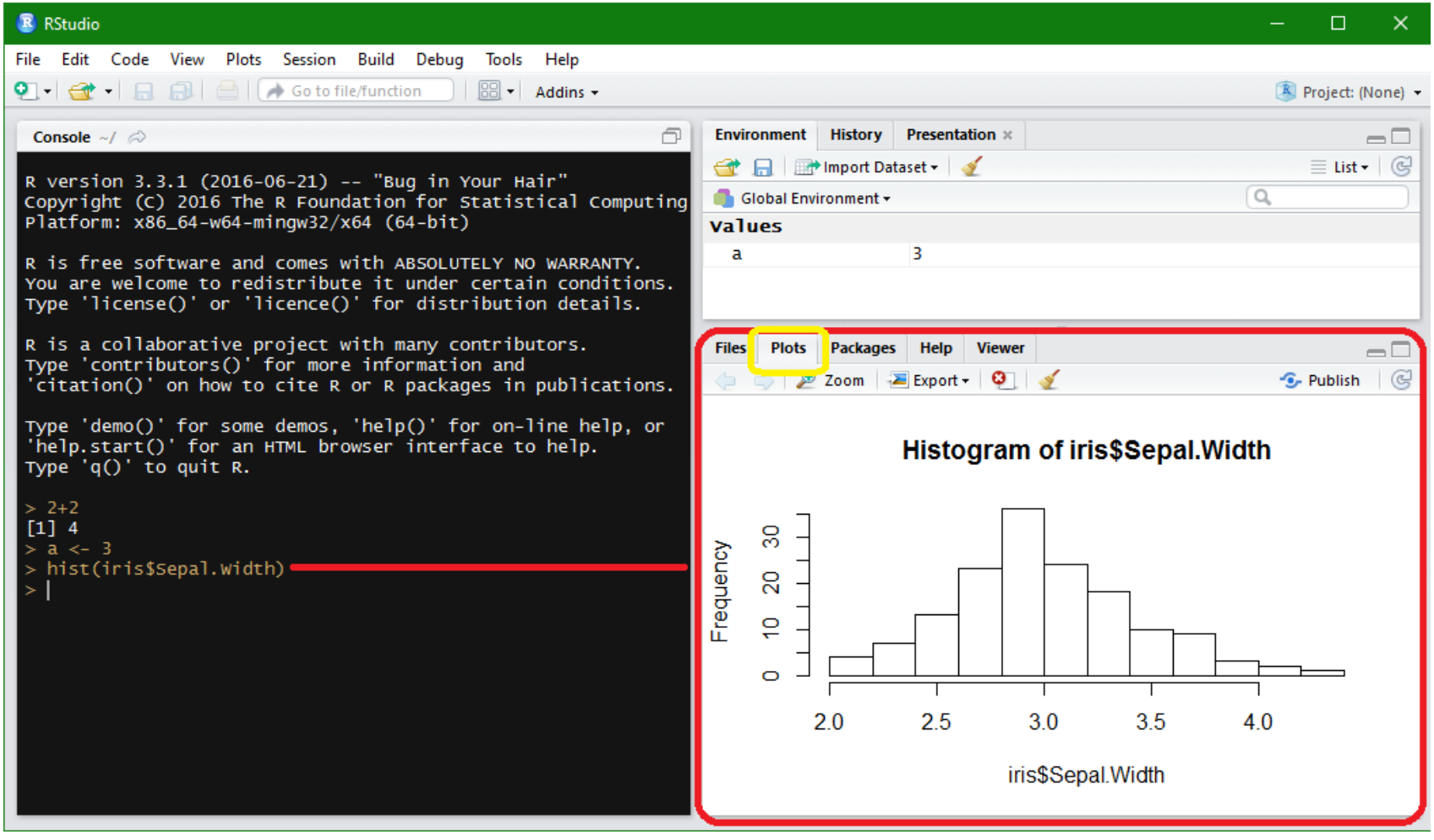
R Studio: A short tour

Files The files tab allows you to open code/script files within R studio.



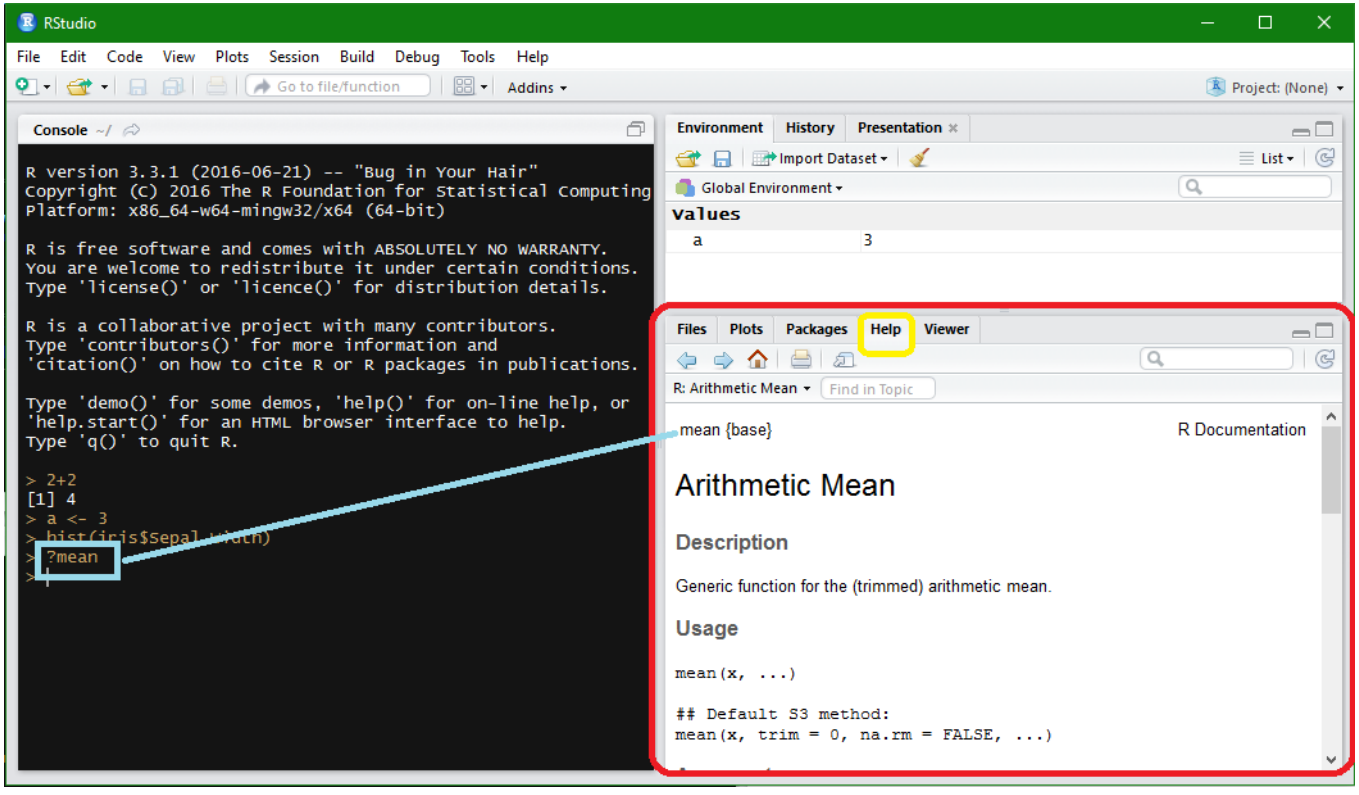
R Studio: A short tour

Plots Any plots that you generate will show up in the panel in the lower right corner.



R Studio: A short tour

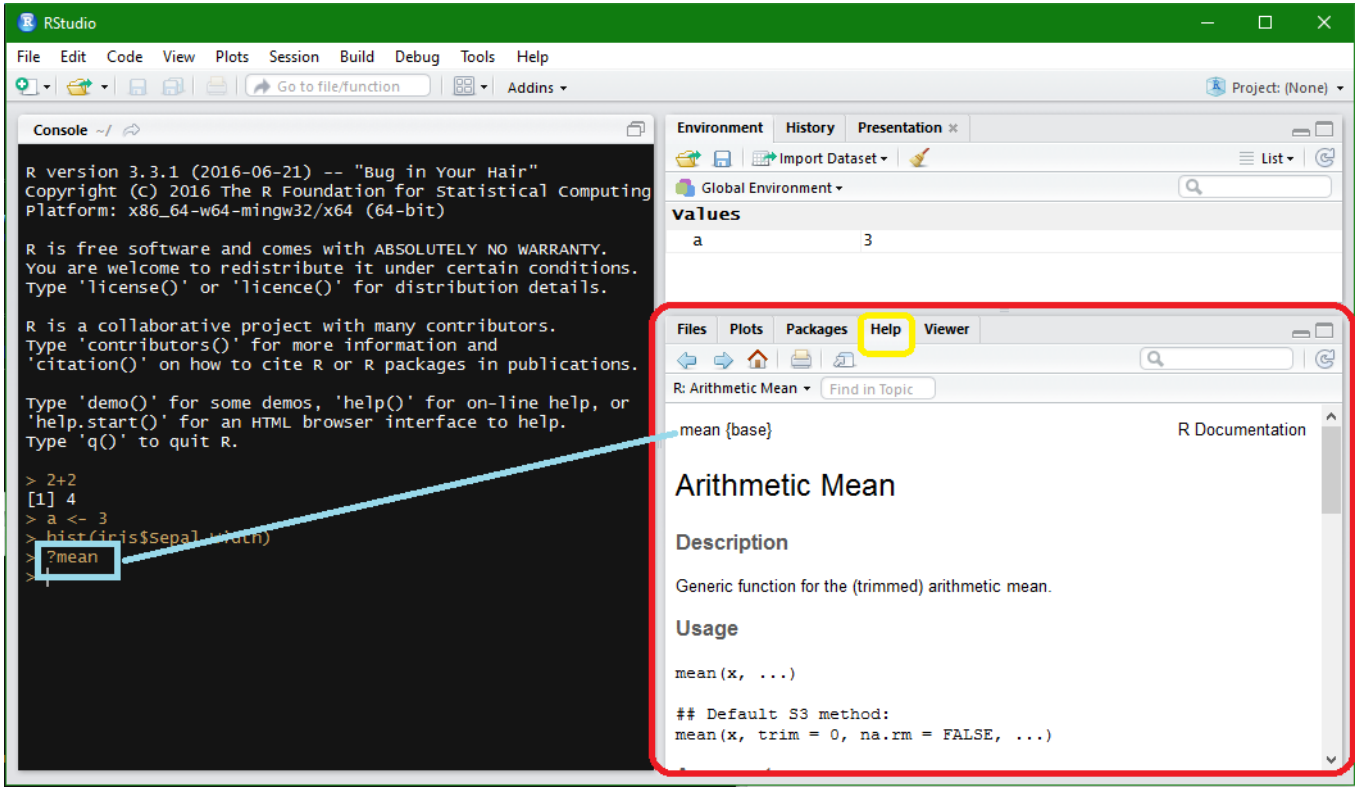
Help To check the syntax of any function in R, type ? in front of the function name to pull up the help file.



For example here I typed ?mean to get the help file for the mean function. Admittedly these aren't the most helpful of files at times.

R Studio: A short tour

Script File The top left is your editor window, where you write code or script, the console is now at the bottom.**I usually change it**



R Script

Most of R users typically submit commands to R by typing either in console or editor panel, rather than clicking a mouse in a Graphical User Interface (GUI).

Script is nothing but **a collection of commands**

R Script and Reproducible Research

There are at least two advantages of doing so.

1. This allows us to run a bunch of results altogether by putting a collection of commands in a file (i.e., "script""").
2. It is also a lot more transparent and straightforward to **share** and **replicate** what you have done.

In our class, you will do this!

R Script

Quiz 1: Create a script file

1. Open the program RStudio and go to File > New > R Script. This will open a blank text document.
2. In the document,

Highlight both lines of code and click the button marked "Run." If everything is working correctly, the console should display TRUE.

Or, pressing CONTROL-ENTER or COMMAND-RETURN depending on whether you're running Mac OSX, Linux or Windows.

3. go to "File > Save As", and choose a file name.