

A First Look at R

Rob McCulloch

June 6, 2020

What is R?

R is a free, open source, program for doing data analysis.

Currently, R and python are the dominant computing environments for data science.

Both R and python are *scripting languages*. You can interactively type in commands at the “command line” and you can maintain a script of commands which can be run in chunks or all at one time.

Installing R and Rstudio

First install R and then Rstudio.

To install R go to <https://www.r-project.org/>

then:

CRAN/pick a mirror/download install (mac, or windows, or linux).

For Rstudio, got to <https://rstudio.com/>

then:

Download/rstudio desktop free.

This YouTube video walks you through it:

<https://www.youtube.com/watch?v=orjLGFmx6l4>

This site <https://swirlstats.com/students.html> also tells you how to install R and Rstudio and is a very nice site for learning R. This site teaches you more about R than you need for Business stats.

First R

On Windows or the Mac you should be able to fire up Rstudio by double clicking the icon on your desktop. The figure below shows you what you what looks like Rstudio on a mac. It should look very similar on Windows.

There are four panes. When you first go into Rstudio the top left pane may not be there. Use /file/new file/R script' to get an (empty) R script in the top left pane.

You type commands into the R script (top-left pane, the “code editor pane”). When you execute the commands you see the results in the console tab of the bottom left pane (the console pane).

The top right pane is the “workspace pane”. The environment tab allows you to see what data and variables you have in your R session. The bottom right pane is the “Notebook pane”.

The files tab of the notebook pane allows you see the files in your working directory. Other useful tabs of the notebook pane allow you to manage your plots, manage your R packages, and get help.

By moving your cursor to the boundaries between the panes you can adjust the size of the panes. By typing CNRL/shift ({0,1,2,3,4}) you can see just one pane (1-4) or all (0).

There are also convenient ways to enlarge or make small each panel by clicking near the top right of each of the four panels.

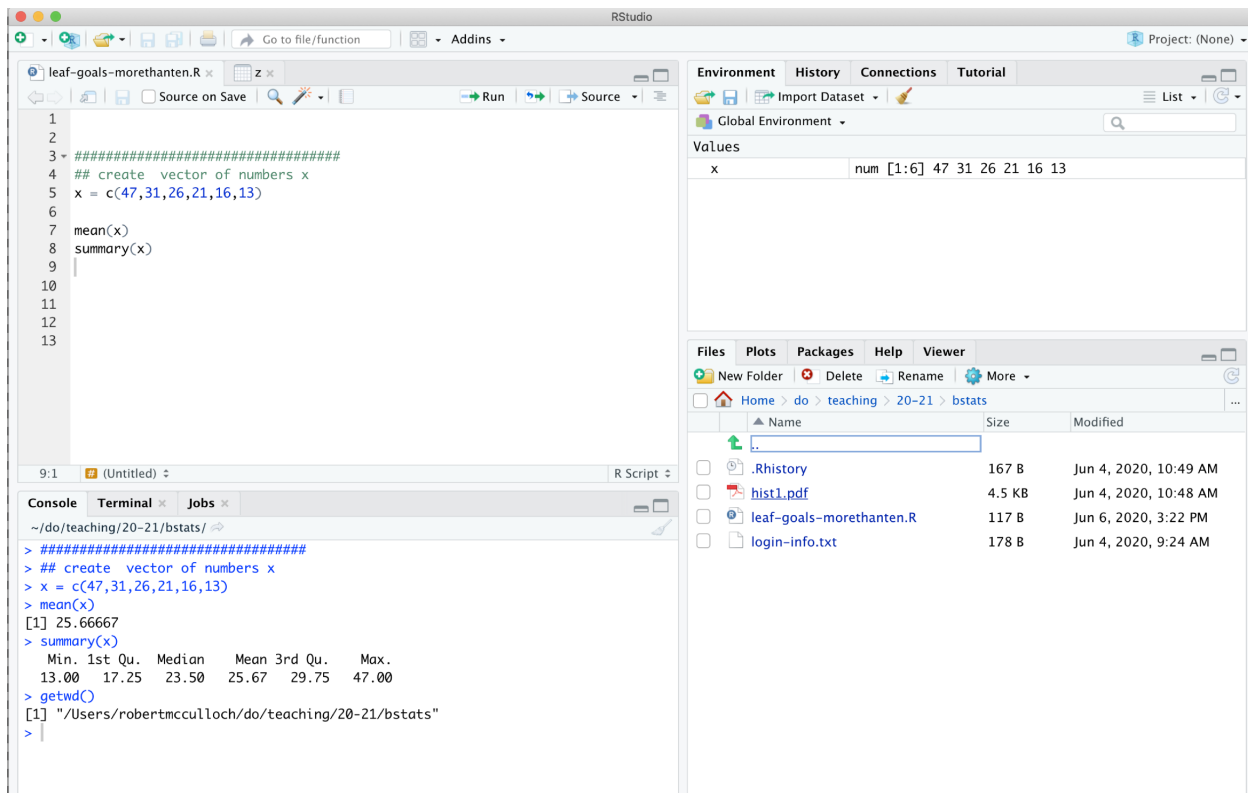


Figure 1: Rstudio on the mac

Simple Calculations in R

The simplest way to use R is to type commands into the console.

```
2+3
## [1] 5
```

The “2+3” is what you type into the console (and then hit return) and below is the result R prints out.

You can do just about any kind of numerical calculation simply in R:

```
2^2
## [1] 4
2**3
## [1] 8
```

```
2+3*5
## [1] 17
10/2
## [1] 5
```

Variables and Functions in R

Just about everything you do in R involves storing some kind of information in a variable and doing something with that information using a function. Let's see a simple example.

```
x=2
```

We have created the variable with name `x`.

The information "2" is stored and we can access this information by the name `x`. For example we can print out `x` just by typing its name.

```
x
## [1] 2
```

We can then do things with the information in `x`:

```
x^2
## [1] 4
```

We can apply *functions* to the information in variables:

```
y = x^2
z = sqrt(y)
z
## [1] 2
```

`sqrt` is the function which calculates the square root. We have stored the results from calling the function `sqrt` with *argument* `y` in the variable `z`.

Now we have three variables in our R session. To see what variables are around use

```
ls()
## [1] "x" "y" "z"
```

The *arguments* to a function are the variables or quantities you put in. Sometimes we will have lots of arguments. Above, the function `ls` has no arguments.

Here we use the `sum` function with two arguments.

```
sum(x,y)
## [1] 6
```

To remove of variable from your workspace, use the `rm` function:

```
ls()
## [1] "x" "y" "z"
rm(z)
ls()
## [1] "x" "y"
```

You can get rid of all the variables using the top-right pane. You just click on the little brush beside the drop down menu labeled “Import Dataset”.

Note that R will often talk about “objects”. For example, when you click the “little brush” just mentioned, you get a message asking you if you want to clear all objects. For us, an object and a variable will almost always mean the same thing.

Note the R is *case sensitive*. `x` is not the same as `X`.

```
xvar = 5
Xvar = 20
ls()
## [1] "x" "xvar" "Xvar" "y"
```

Help in R

If you want help for the function `sum`,

```
help(sum)
```

Help results will show up under the Help tab in the lower-right pane.

Under the Help tab of the lower-right pane, try clicking on the “little house” (Show R help). This will get you to several links to R help. The R-studio help is generally pretty good. Check out the Cheat sheets, they are great.

Note that *googling* works amazing well for R. Anytime you don’t know how to do something, just google it. For example, try googling, “how do i list variables in R”.

Getting out of R

type `q()`, or `/R studio/` quit R studio.

Working With a Vector of Data, a Simple R script

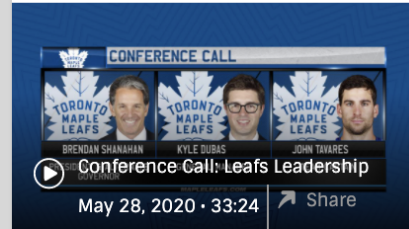
Let’s get some data in R and have a look at it.

Here is data on scoring for the Toronto Maple Leafs.







Skaters

Player	GP	G	A	P	+/-	PIM	PPG	PPP	SHG	SHP	GWG	OTG
 Auston Matthews	70	47	33	80	19	8	12	25	0	0	5	0
 William Nylander	68	31	28	59	-2	12	9	17	0	0	7	1
 John Tavares	63	26	34	60	-7	24	7	21	0	0	4	3
 Zach Hyman	51	21	16	37	13	23	3	6	1	1	1	0
 Mitchell Marner	59	16	51	67	6	16	6	24	0	1	2	1
 Kasperii Kapanen	69	13	23	36	0	22	0	3	2	2	2	1
 Alexander Kerfoot	65	9	19	28	-2	32	1	3	0	0	1	0
 Jason Spezza	58	9	16	25	-3	18	2	7	0	0	0	0
 Ilya Mikheyev	39	8	15	23	7	4	0	0	1	1	1	0
 Andreas Johnsson	43	8	13	21	0	14	4	5	0	0	1	0
 Pierre Engvall	48	8	7	15	4	6	0	0	1	2	3	0
 Frederik Gauthier	61	7	5	12	-4	10	0	0	0	0	0	0
 Jake Muzzin	53	6	17	23	12	40	0	0	0	1	1	0
 Tyson Barrie	70	5	34	39	-7	16	1	12	0	0	0	0
 Travis Dermott	56	4	7	11	14	37	0	0	0	0	1	0

Leafs TV



Leaderboard

	GOALS Auston Matthews #34 · TOR · Center	47 GOALS
	ASSISTS Mitchell Marner #16 · TOR · Right Wing	51 ASSISTS
	POINTS Auston Matthews #34 · TOR · Center	80 POINTS
	PPG Auston Matthews #34 · TOR · Center	12 PPG
	WINS Frederik Andersen #31 · TOR · Goalie	29 WINS
	SV% Frederik Andersen #31 · TOR · Goalie	.909 SV%

As a simple example, let's look at the goals scored for those players who got more than 10 goals.

We can create a variable which stores all the information.

```
x=c(47,31,26,21,16,13)
x
## [1] 47 31 26 21 16 13
```

Already, typing this into the console becomes cumbersome.

Create an R script “/file/new file/ Rscript” and the type

```
# simple R script to look at leafs scoring
x=c(47,31,26,21,16,13)
```

into the script window.

Click on the line `x=c(47,31,26,21,16,13)` and then click on the **run** button (top right of the code editor pane).

You will then see the results of running this command in the console!!

Click on the console window and just enter `x`, it will print out. Notice how `x` shows up in the Environment tab of the Workspace pane (top right).

Add the following lines to your R script. You can execute each line one at a time by repeatedly clicking **run** or select a bunch of lines to be run together and then clicking **run**.

```
# simple R script to look at leafs scoring
x=c(47,31,26,21,16,13)
mean(x)
sd(x)
summary(x)
```

We are in business, this is how you use R!!

Here is what your script should produce.

```
# simple R script to look at leafs scoring
x=c(47,31,26,21,16,13)
mean(x)
## [1] 25.66667
sd(x)
## [1] 12.32342
summary(x)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      13.00  17.25   23.50   25.67  29.75   47.00
```

If you want to get a spreadsheet type view of the variable `x`, you can use the R function `View`. Try

```
> View(x)
```

Now we want to save our R script.

But first we have to understand how to set your *working directory*.

Working Directory

R has the notion of a “working directory”.

Note the “folder” and “directory” mean the same thing.

The working directory is the place R will look for files if you want to read something (e.g. data) in or write something out (e.g. and R script) to save it.

To set your working directory, go to `/session/set working directory`.

Save your simple script and check that it shows up in the files tab of the bottom right pane.

Warning, the assignment operator in R

Note that when I want to assign a value to the variable `x` I use:

```
x=2
x
## [1] 2
```

You will also see an alternative way to make assignments:

```
y <- 5
y
## [1] 5
```

R purists prefer the “<-” notation, but I like simple old “=”.

Names for variables

In our goals scored example, I used the name `x` to refer to the vector of values recording the number of goals scored by the top Leaf scorers.

This is terrible !!!

In general you want to choose a name that means something. For example

```
leafsGS = c(47,31,26,21,16,13)
summary(leafsGS)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  13.00  17.25   23.50   25.67   29.75   47.00
```

Choosing good names for you variables is one of the most important and difficult parts of the process.

In statistics, we often want to talk about general ideas using some notion of a variable. The names `x` and `y` get used a lot. But in your actual application, you should use more meaningful names.

Arguments to Functions

The basic way R works is to have data contained in vectors or data frames, or some other data structure and then apply functions to the data.

The “arguments” to the function is what goes in. For example:

```
seq(1,10)
## [1] 1 2 3 4 5 6 7 8 9 10
```

Here the function `seq` is taking two arguments, the first gives the start and the second gives the end for a sequence of integers.

There are a few ways to provide arguments to functions. The basic way is to give the arguments in a specific order. Above, the function `seq` knows that the first argument is the start and the second argument is the end.

The other basic way to provide arguments is by their name:

```
x = seq(from=1,to=10)
x
## [1] 1 2 3 4 5 6 7 8 9 10
```

If I give the arguments by name, the order does not matter:

```
x = seq(to=10,from=1)
x
## [1] 1 2 3 4 5 6 7 8 9 10
```

Functions can have a lot of arguments and it often helpful to call by name so you know what you are talking about.

For example:

```
x = seq(from=1,to=10,length.out = 20)
x
## [1] 1.000000 1.473684 1.947368 2.421053 2.894737 3.368421 3.842105
## [8] 4.315789 4.789474 5.263158 5.736842 6.210526 6.684211 7.157895
## [15] 7.631579 8.105263 8.578947 9.052632 9.526316 10.000000
```

You can figure out what happened just from the names of the arguments.

Try `> help(seq)` and see how the arguments are discussed.

A Simple Plot

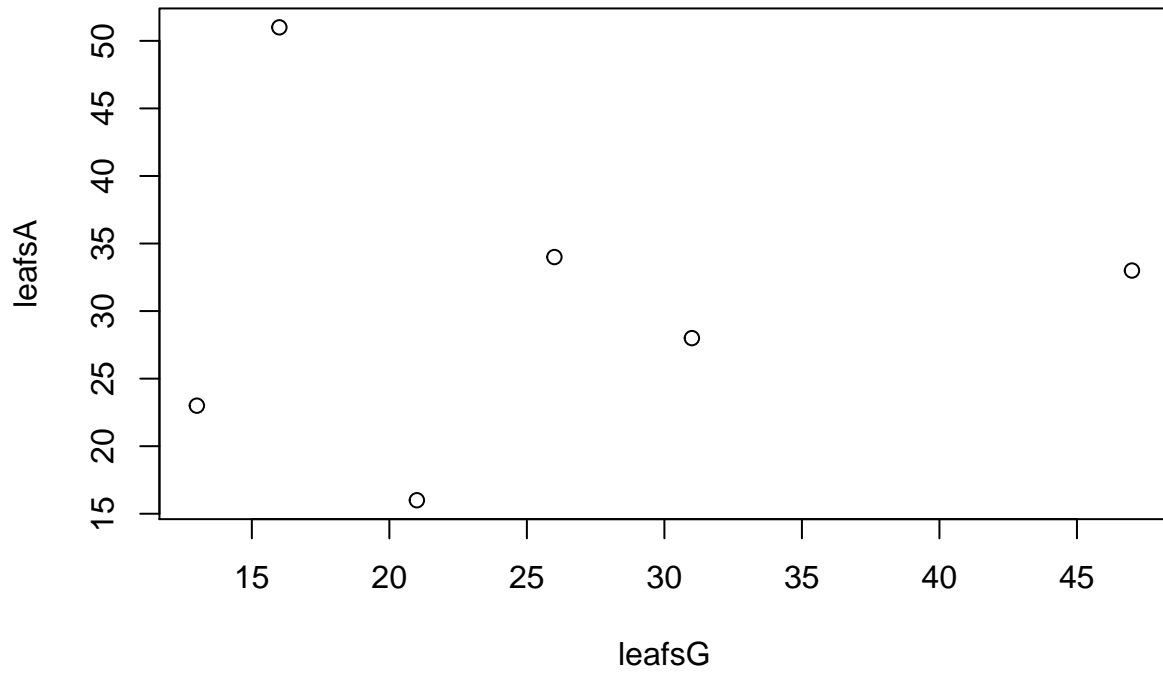
Let's do a simple scatter plot (x versus y) in R and learn about some of the optional arguments to the `plot` function.

```
# lets get the Leaf players goals and assists
leafsG = c(47,31,26,21,16,13) # goals
leafsA = c(33,28,34,16,51,23) # assists

#summarize
cat("Goals\n") # cat will print to screen, \n is a newline
## Goals
summary(leafsG)
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  13.00  17.25   23.50   25.67  29.75   47.00
cat("Assists")
## Assists
summary(leafsA)
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  16.00  24.25   30.50   30.83  33.75   51.00
```

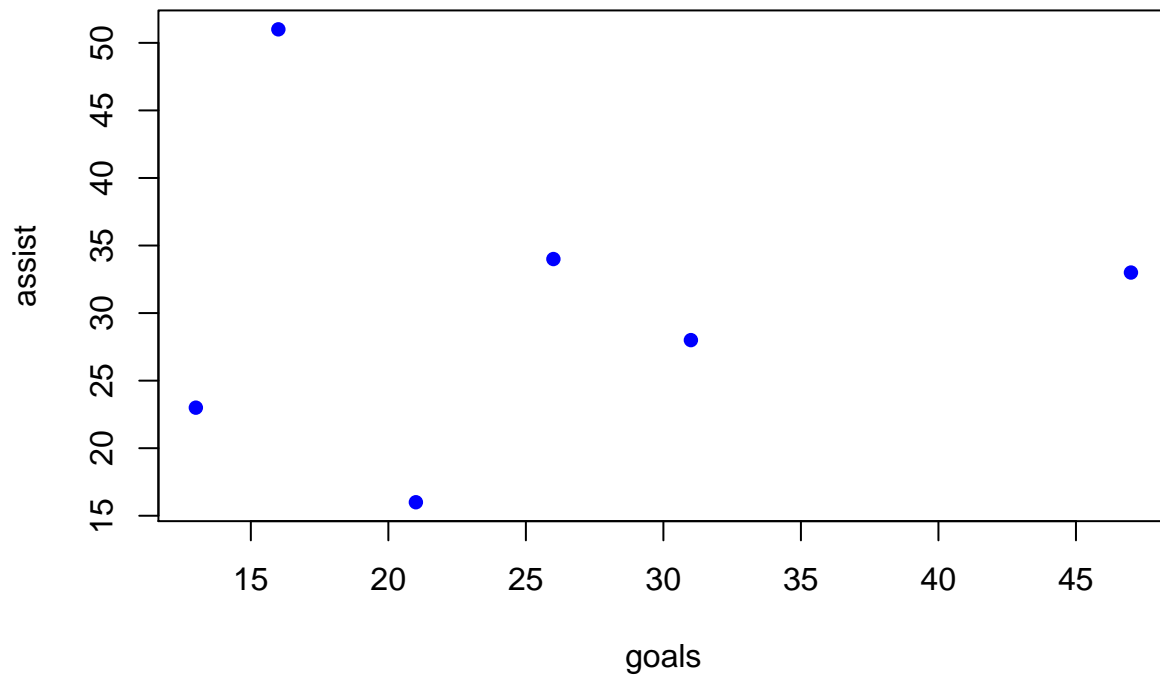
In order to visualize the Leaf players let's plot `leafG` versus `leafA`.

```
plot(leafsG,leafsA)
```

Two players really stick out from the 6.
Matthews as a very large number of goals and Marner has a very large number of assists.
Additional arguments to the `plot` function allow us to control the appearance of the plot.

```
plot(leafsG,leafsA, xlab="goals",ylab="assist",col="blue",pch=16)
```



- xlab, ylab : labels for axes
- col: color of plotted point
- pch: plot symbol

Try googling “plot symbols in R”.

There are lots more optional arguments to `plot`. Try typing “`help(plot)`” in the console.