

# An Unfortunately Brief Introduction to R

Marshall A. Taylor, Ph.D.

2020 HSI Learning Resilience Conference  
4:15p to 5p (EST)  
November 3, 2020

# What is R?

- R is a programming language optimized for statistical computing.
- It is a contemporary implementation of the S statistical programming language.
- It is “open source,” which means that it is free, publicly available, and transparent in the sense that all underlying code can be viewed and adapted by end-users. Think of it as “statistical programming for the people, by the people.”

# What is R?

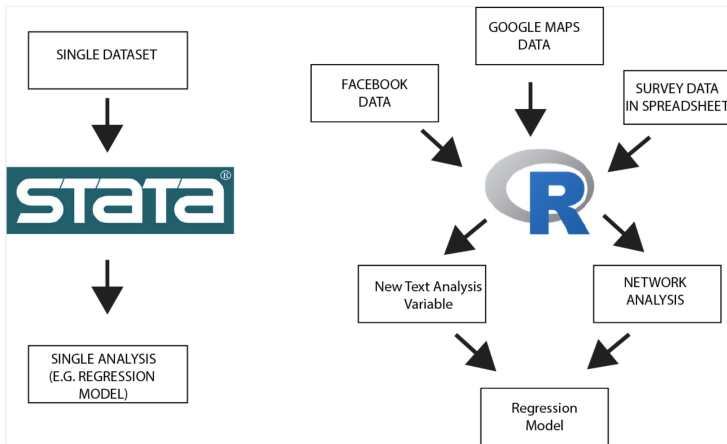
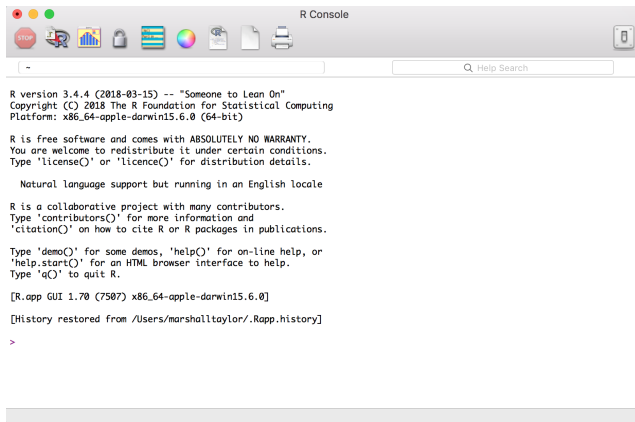


Figure 1: R is an “object-oriented” language

*Note:* Visualization from Bail (2016).

# What is R?



The image shows a screenshot of the R Console window on a macOS system. The window has a title bar with standard macOS window controls (red, yellow, green buttons) and a title "R Console". Below the title bar is a menu bar with various icons (stop, search, bar chart, lock, grid, color wheel, document, printer) and a search field labeled "Help Search". The main content area displays the R startup message, which includes the version number (3.4.4), copyright information (© 2018 The R Foundation), platform details (x86\_64-apple-darwin15.6.0), and a welcome message. It also lists several useful commands like 'license()', 'demo()', 'help()', and 'q()'. The prompt is a red greater-than sign (>).

```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.70 (7507) x86_64-apple-darwin15.6.0]
[History restored from /Users/marshalltaylor/.Rapp.history]

>
```

Figure 2: A Look at R

Not much to look at, right?

# Introducing RStudio

- RStudio is an **IDE** (**I**ntegrated **D**evelopment **E**nvironment).
- IDE's essentially exist to make programming in particular languages a bit easier by “integrating” various aspects of the programming process into a single interface.
- Once you have installed R and RStudio, you should never have to open R again; just open RStudio. RStudio works with R “behind the scenes,” so to speak.

# Introducing RStudio

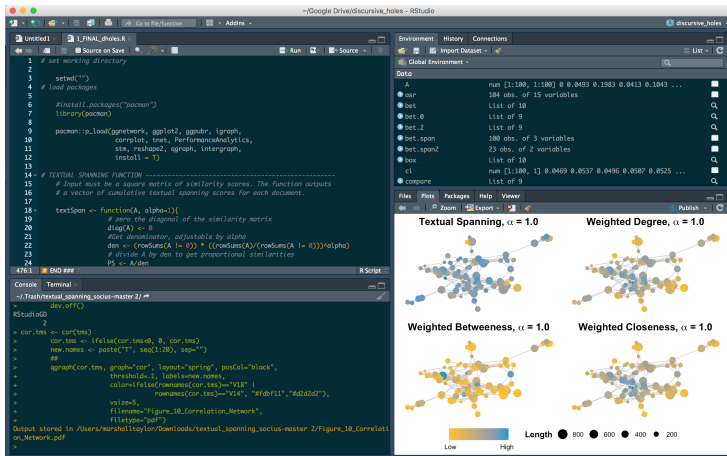


Figure 3: A Look at RStudio

# Getting Started

## Open RStudio.

You should see this (though yours will be white, unless you changed your color palette like I did on my machine).

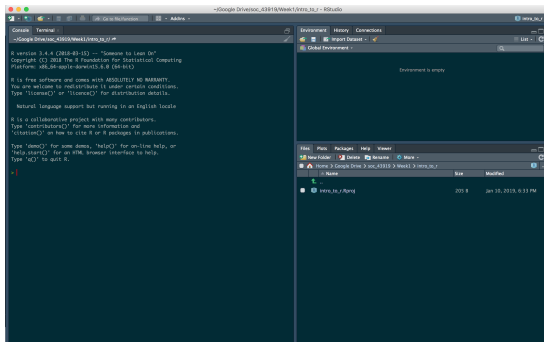


Figure 4: A Fresh RStudio Environment

# Getting Started

Now click the little double window icon to the right of the “Console” panel.

This will separate the left panel in two. On the top is the **editor** where you will do your coding. On the bottom is the **R console**, where your output will be printed.

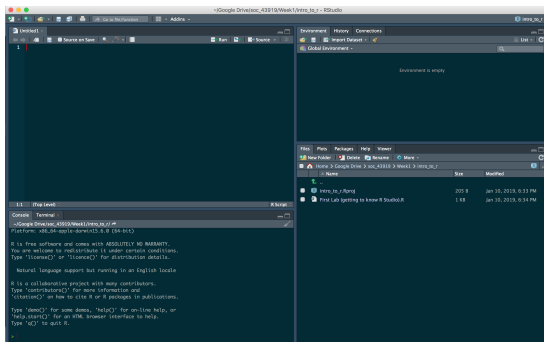


Figure 5: RStudio with a new editor panel open



# Anatomy of the RStudio Environment

## The Editor

This where you'll be doing your coding.

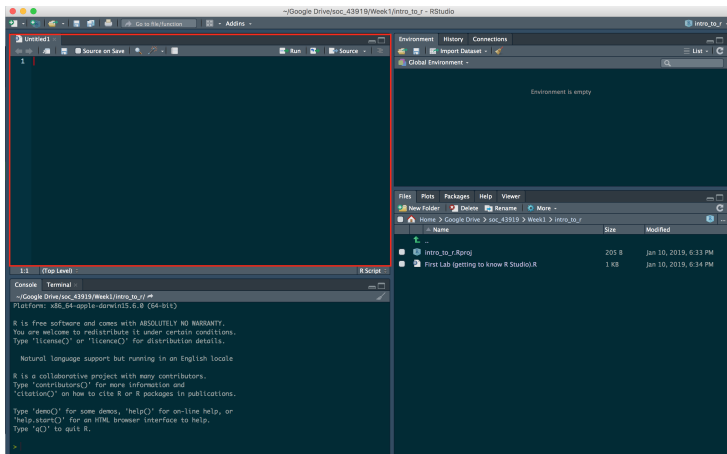


Figure 6: The Editor

# Anatomy of the RStudio Environment

## The Console

This where the output from your coding will be printed.

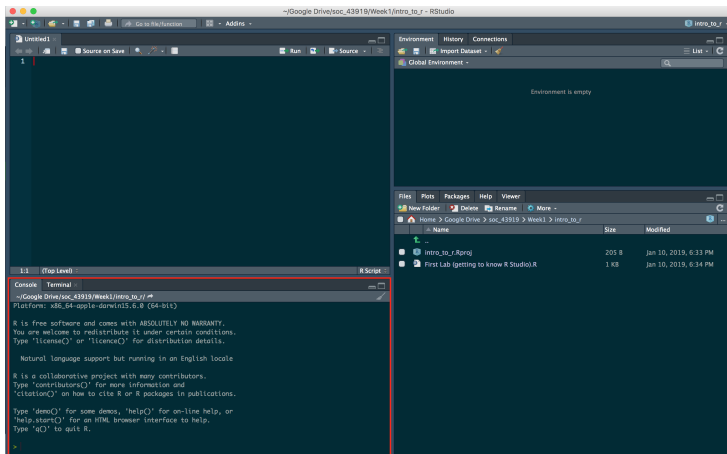


Figure 7: The Console

Let's give it a try. First, name your editor file (also called a **script** with your first initial and last name followed by “\_Rintro”)—e.g., for me, MTaylor\_Rintro.R.

Then try multiplying 5 by 5 from your script. Simply type `5 * 5` on line #1, and press either `Ctrl+Enter` (Windows) or `Cmd+Enter` (Mac) like this:

```
5 * 5 # Use hashtags to make notes to yourself. The
      # hashtag tells R not to try to run anything
      # on the same line.
```

And the answer will be printed in the console panel:

```
[1] 25
```

# Anatomy of the RStudio Environment

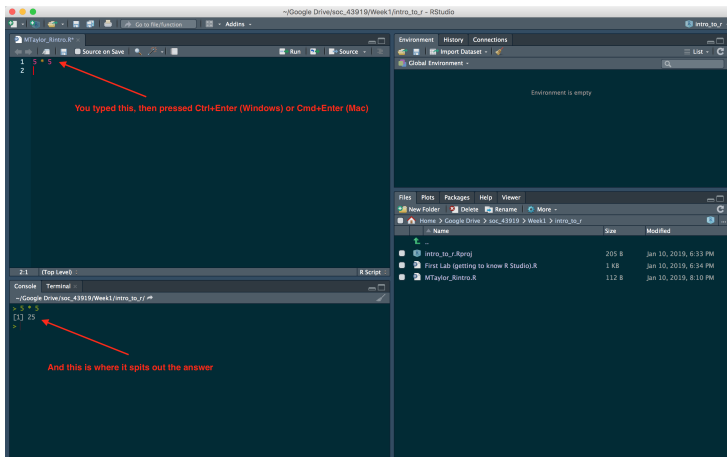


Figure 8: Your First Executed Command!

# Anatomy of the RStudio Environment

Recall that R is an object-oriented language. When you create an object, it will show in the “Environment” panel to the top right. These objects can anything from **datasets**, **vectors of words or numbers**, **matrices**, single numbers, words, lists, and so on.

Creating objects follows this structure:

**object\_name** <- **command to define the object**

Let's create a few objects of different types.

```
# A single number
object_1 <- 5 * 5

#A single word (called "strings" in programming
  #parlance)
object_2 <- "text"

#A "vector" of numbers
object_3 <- c(1, 2, 3, 4, 5)

#A vector of words
object_4 <- c("latte", "mocha", "espresso",
  "americano", "cappuccino")
```

# Anatomy of the RStudio Environment

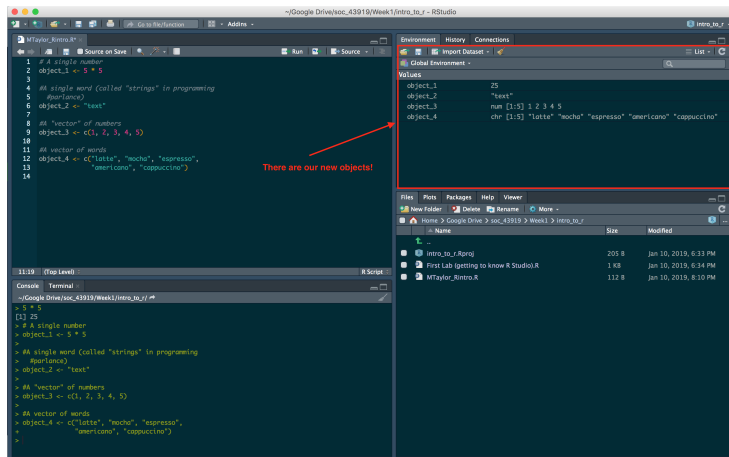


Figure 9: Creating your first objects



# Anatomy of the RStudio Environment

We can then “re-invoke” each object without having to re-issue the command.

We can either (a) highlight the object name in the code and run it, or (b) re-type the object name and run that line.

We can then do all sorts of things with these objects that would be much more difficult if we tried to do them on a cell-by-cell basis. For example, try running this:

```
#Multiply all items in a numeric object by a constant
object_5 <- object_3 * 5

#Bind two objects together
object_6 <- cbind(object_4, object_5)

#Change column names
colnames(object_6) <- c("cafe_drinks", "number")

#Convert one type of object into a different type
object_6 <- as.data.frame(object_6)

#Convert particular columns in the object from strings
#to numeric vectors or vice versa
object_6$number <- as.numeric(object_6$number)

#Define new columns/variables within an object
object_6$product <- object_6$number * 10

#Create a new object that is the transpose of object_6
object_7 <- t(object_6)
```

Note that matrices and data frames—two types of objects you will be working with a lot—will show up under the “Data” viewer. You can open them up in a new tab by clicking on the little spreadsheet icon to the right of the data viewer.

# Speaking of Matrices and Data Frames...

You will be working frequently with **matrices** and **data frames** in R. Matrices are two-dimensional data structures (i.e., the data are arrayed along two features, rows and columns) where each entry must be of the same type (Wickham 2014:13). For example, rows may be individuals, columns may be movie genres, and cell entries may be the number of movies within each genre that each person owns:

	Comedy	Horror	Action	Drama
Person 1	4	0	0	2
Person 2	1	8	2	1
Person 3	0	1	1	1
Person 4	7	4	3	3
Person 5	1	0	0	0

Table 1: A  $4 \times 5$  Matrix

# Speaking of Matrices and Data Frames...

Data frames are also two-dimensional objects, but the contents can be “heterogenous”—that is, they can be of different types (Wickham 2014:13). For example, a data frame can contain both numeric vectors (like how many movies of a particular genre a person owns) and string vectors (such as a person’s favorite genre). In R, we’ll usually treat string vectors as **factors**—which are essentially pre-defined categories.

	Comedy	Horror	Action	Drama	Favorite
Person 1	4	0	0	2	Comedy
Person 2	1	8	2	1	Horror
Person 3	0	1	1	1	Action
Person 4	7	4	3	3	Comedy
Person 5	1	0	0	0	None

Table 2: A Data Frame of Different Types

# Speaking of Matrices and Data Frames...

Since data frames can house different types of data, they are what we would usually call **datasets** in other statistical software programs: e.g., Stata, SPSS, or SAS. Typically, you will want your two-dimensional objects to be data frames. If you have a two-dimensional object that is in matrix form, you can easily convert it to a data frame using the `as.data.frame()` function (shown in an earlier slide).

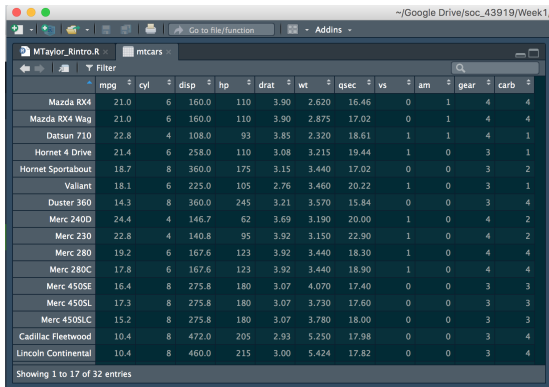
Let's take a look at a “real” data frame. R comes with a handful of data frames ready to go, mostly for teaching and practice. One consists of data from the 1974 *Motor Trend* magazine, which provides various statistics across 32 vehicles produced in 1973 and 1974. Let's load it using the `data()` function and take a look at the first 10 rows using the `head()` function:

```
#Load mtcars data
data(mtcars)

#Take a look at the data
head(mtcars, 10)
```

# The Data Viewer

Now click on the spreadsheet icon associated with the mtcars object over in the global environment pane. You should see something like this pop up as a new tab in the editor pane:



The screenshot shows the RStudio Data Viewer interface. At the top, there's a window title bar with standard macOS window controls and a path: ~/Google Drive/soc\_43919/Week1/. Below the title bar is a toolbar with icons for file operations and a 'Go to File/function' button. The main area is divided into two panes: 'MTaylor\_Rintro.R' and 'mtcars'. The 'mtcars' pane is active and displays a table of car data. The table has 11 columns: mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, and carb. The data is sorted by mpg in descending order. The first 17 rows are visible, and a status bar at the bottom indicates 'Showing 1 to 17 of 32 entries'.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4

Figure 10: Looking at the Data using the Data Viewer

R runs on what we call “packages.” These are user-written programs—or sets of programs “packaged” together—that you install and load to carry out various functions.

For example, maybe we want to get some descriptive (or summary) statistics to get a sense of how the mpg (miles per gallon) variable in the mtcars data frame is distributed. There are various functions we can use, but let's use the describe() function from the psych package (Revelle 2018):

```
#Install the package  
install.packages("psych")
```

```
#Load the package  
library(psych)
```

```
#Use the describe() that comes with the package  
describe(mtcars$mpg)
```



And here's what the `describe()` function should print to the console:

```
> describe(mtcars$mpg)
  vars  n  mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1     1 32 20.09 6.03   19.2   19.7 5.41 10.4 33.9  23.5 0.61    -0.37 1.07
```

And if you need to know more about a package or a specific function within that package:

```
#Want to know what other things we can do with the psych package?
?psych #or
help("psych")

#Or a specific function?
?describe #or
help("describe")
```

**Never** just hop into a work session without setting your working directory. This is especially important for those of you working from ND campus computers.

You can check your current directory with the `getwd()` function:

```
#What's my current directory?  
getwd()
```

So if I was working from my desktop, I would get this printed to the console:

```
[1] "/Users/marshalltaylor/Desktop"
```

Then use the `setwd()` function to set your new directory:

```
#Reset the working directory  
setwd("/Users/marshalltaylor/Google_Drive/soc_43919/Week1/intro_to_r")
```

And verify that everything is correct by re-issuing the `getwd()` function:

```
[1] "/Users/marshalltaylor/Google_Drive/soc_43919/Week1/intro_to_r"
```

This is a very important step and should be the first thing you do before working on anything in this class. **If you are working from a campus computer and do not set your directory to your Google Drive, then your files will be stored locally on that particular computer and will be wiped clean after you log off!**

Finally, always be sure to save your work.

Save your list of R objects (all the objects listed in your “Global Environment” to the upper right of the RStudio screen) using the `save.image()` function:

```
#Save all of the objects I created today  
save.image("MTaylor_Rintro.RData")
```

You can also save your objects using the floppy disk icon where the objects are listed.

And save your R script—where you’ve been writing all of your code—using either the floppy disk icon about the editor or keyboard shortcuts (Ctrl+S for Windows or Cmd+Enter for Mac).

There are great online resources for learning to navigate R and RStudio. The [readings for this week](#) are some great places to start.

Also try Googling your questions. I can almost guarantee most of your questions have answers, most likely on [Stack Overflow](#).

And here's a [link](#) to an RStudio “cheatsheet.”

- Bail, Christopher A. 2016. "Computational Sociology." Class #1 Slides. Retrieved January 10, 2019 ([link](#)).
- Revelle, William. 2018. "psych: Procedures for Psychological, Psychometric, and Personality Research." R package version 1.8.10.
- Wickham, Hadley. 2014. *Advanced R*. Boca Raton, FL: CRC Press.