

Lab 1: Introduction to R

STAT 310, Spring 2023

Contents

Installing R	2
Installing RStudio	2
RStudio Cloud	2
Getting Started with RStudio	3
Using R as a Calculator	4
Variable Assignment	5
Numeric Vectors	6
Sequences	6
Vector Functions	7
Subsetting a Vector	8
Data Frames	8
Subsetting Rows and Columns	10
Scatter Plot	11
Practice Problems	12

Installing R

To download the latest version of R go here:

<https://cloud.r-project.org/>

Click on the “Download R” link for your operating system (Windows, Mac) and follow the instructions for installation.

Installing RStudio

To download the latest version of RStudio go here:

<https://rstudio.com/products/rstudio/download/>

Click on the button to download “RStudio Desktop Open Source License” (free version). Next click on the link to download RStudio for your operating system (Windows, Mac) and follow the instructions for installation.

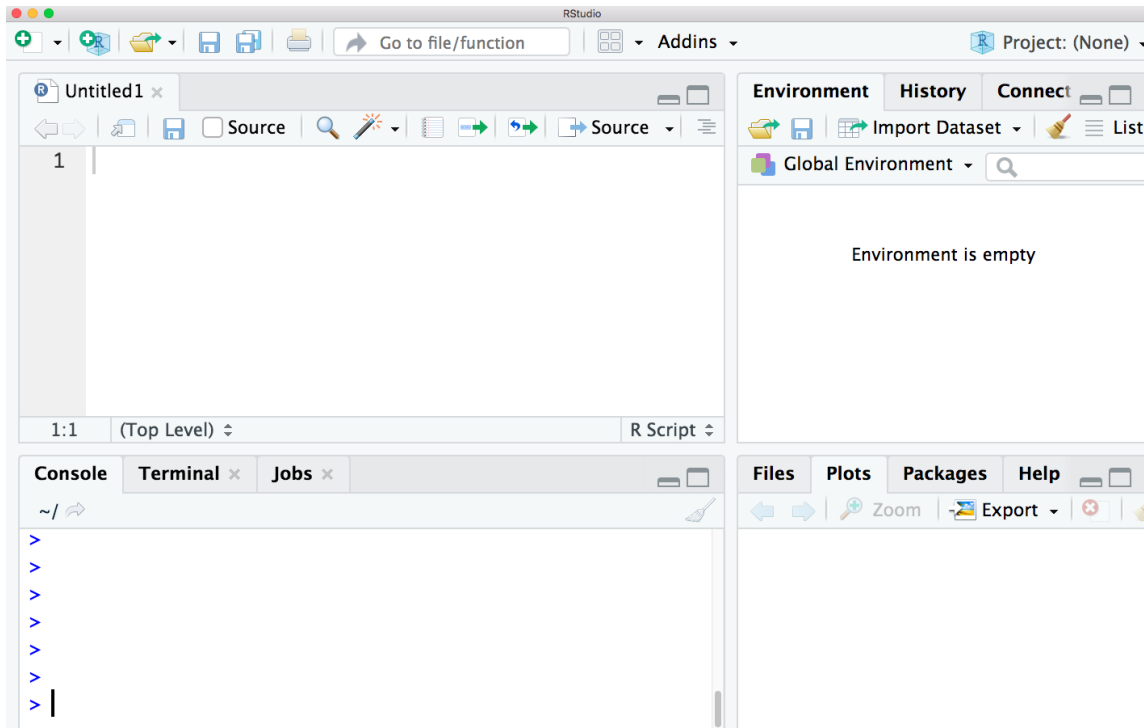
RStudio Cloud

As an alternative to the desktop version, RStudio Cloud allows you to use RStudio through your internet browser.

<https://rstudio.cloud/>

Getting Started with RStudio

Go ahead and open RStudio, what you see should look like the screenshot below. For clarification: R is the name of the programming language and RStudio is a convenient interface.



RStudio is divided into four panels:

- Top left panel: This is the code editor, this is where you write your code. To create a new R script go to File → New File → R Script. Make sure to save your R scripts so that you can access them later.
- Bottom left panel: This is the console, this is where you run your code. You can either copy and paste code from an R script to the console, or use the shortcut Ctrl+Enter (Windows) or Command+Enter (Mac) to run a line of code.
- Bottom right panel: Any plots you create will show up in this panel. This panel also contains the help menu, where you can read documentation about R functions.
- Top right panel: This contains panel the Environment and History. The Environment panel shows everything that is currently in your workspace (e.g., objects you create or data sets you have loaded for analysis). The History panel provides a history of all previous commands you have run in the console.

Using R as a Calculator

You can use R for basic calculations: addition, subtraction, multiplication, and exponentiation

```
2 + 2
```

```
## [1] 4
```

```
5 - 2
```

```
## [1] 3
```

```
2 * 3
```

```
## [1] 6
```

```
2^4
```

```
## [1] 16
```

```
2^(1/2)
```

```
## [1] 1.414214
```

```
sqrt(2)
```

```
## [1] 1.414214
```

```
(1/51 + 1/49)^(1/2)
```

```
## [1] 0.20004
```

```
7.2 + 2 * 2.1 / sqrt(101)
```

```
## [1] 7.617916
```

If you want to edit a previous command, press the up arrow key. For instance, I may want alter the last computation:

```
7.2 - 2 * 2.1 / sqrt(101)
```

```
## [1] 6.782084
```

Variable Assignment

The `<-` symbol is called the assignment operator. It assigns values to variables.

```
x <- 7 # assign 7 to the variable x
x # print value of x
```

```
## [1] 7
```

```
y <- 3
y
```

```
## [1] 3
```

```
x + y
```

```
## [1] 10
```

The `=` symbol can also be used for assignment.

```
z = -1
z
```

```
## [1] -1
```

The `#` symbol is used to write comments. Anything to the right of `#` is not evaluated in the console. Comments are useful when sharing code with others (or to help me understand my code in the future!).

Numeric Vectors

A **numeric vector** in R is just a collection of numbers. Vectors are created using `c()`, which is short for combine.

```
x <- c(1, 2, 3, 47)
y <- c(1, -1, 2, -2)
```

We can add and multiply vectors. We can also multiply or divide a vector by a scalar. Operations are performed element-wise.

```
x + y
```

```
## [1]  2  1  5 45
```

```
x * y
```

```
## [1]  1 -2  6 -94
```

```
2*x # multiplies each element by 2
```

```
## [1]  2  4  6 94
```

```
x/2 # divides each element by 2
```

```
## [1]  0.5  1.0  1.5 23.5
```

```
x^2 # squares each element of the vector
```

```
## [1]  1  4  9 2209
```

Sequences

Some ways to create vectors containing sequences of numbers:

```
x <- 1:10
x
```

```
## [1]  1  2  3  4  5  6  7  8  9 10
```

```
y <- seq(2, 20, by=2)
y
```

```
## [1]  2  4  6  8 10 12 14 16 18 20
```

```
z <- seq(0, 1, by=0.1)
z
```

```
## [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Vector Functions

R provides some easy-to-use functions for computing numerical summaries of vectors. Below are some examples using a vector containing the ages of 10 people.

```
age <- c(54, 40, 53, 26, 56, 52, 28, 39, 15, 17)
```

```
length(age) # length of vector
```

```
## [1] 10
```

```
min(age)
```

```
## [1] 15
```

```
max(age)
```

```
## [1] 56
```

```
mean(age)
```

```
## [1] 38
```

```
median(age)
```

```
## [1] 39.5
```

```
sd(age) # standard deviation
```

```
## [1] 15.70563
```

Or more conveniently, use `summary()` to compute several summary statistics at once.

```
summary(age)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      15.00   26.50   39.50   38.00   52.75   56.00
```

Note that **1st Qu.** is the first quartile (25th percentile) and **3rd Qu.** is the third quartile (75th percentile).

Documentation on these functions is provided in the help menu. To access the help menu from the console use the `help()` function. For example, enter the following command to read about the `mean` function in the help menu:

```
help(mean)
```

Subsetting a Vector

To subset specific elements of a vector use brackets `[]`. The number in the bracket is the index, that is, the position of the element within the vector.

```
age <- c(54, 40, 53, 26, 56, 52, 28, 39, 15, 17)
```

```
age[1]
```

```
## [1] 54
```

```
age[4]
```

```
## [1] 26
```

```
age[1:4]
```

```
## [1] 54 40 53 26
```

```
age[c(1, 4)]
```

```
## [1] 54 26
```

Data Frames

Data sets in R are represented as objects called **data frames**. The columns of a data frame are the variables, and the rows of a data frame are the observations.

As an example, let's start by examining a data frame called `mtcars`, which is already loaded into R. This data set comes from the 1974 *Motor Trend* magazine, and contains variables (columns) on automobile design and performance for 32 different automobile models (rows).

```
head(mtcars) # preview first several rows
```

##	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
## Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
## Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
## Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
## Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
## Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
## Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

The `head()` function provides a preview of the first several rows of the data frame. You can also type `mtcars` into the console, and it will print out the entire data set. But for larger data sets this is not advisable. To learn more about this data, use the help menu by typing `help(mtcars)` into the console.

Below are some functions that provide information on the size of a data frame:

```
nrow(mtcars) # number of rows
```

```
## [1] 32
```

```
ncol(mtcars) # number of columns
```

```
## [1] 11
```

```
dim(mtcars) # dimension
```

```
## [1] 32 11
```

We can also extract the row and column names:

```
names(mtcars) # names of the columns (variables)
```

```
## [1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"  
## [11] "carb"
```

```
row.names(mtcars) # names of the rows (car models)
```

```
## [1] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710"  
## [4] "Hornet 4 Drive" "Hornet Sportabout" "Valiant"  
## [7] "Duster 360" "Merc 240D" "Merc 230"  
## [10] "Merc 280" "Merc 280C" "Merc 450SE"  
## [13] "Merc 450SL" "Merc 450SLC" "Cadillac Fleetwood"  
## [16] "Lincoln Continental" "Chrysler Imperial" "Fiat 128"  
## [19] "Honda Civic" "Toyota Corolla" "Toyota Corona"  
## [22] "Dodge Challenger" "AMC Javelin" "Camaro Z28"  
## [25] "Pontiac Firebird" "Fiat X1-9" "Porsche 914-2"  
## [28] "Lotus Europa" "Ford Pantera L" "Ferrari Dino"  
## [31] "Maserati Bora" "Volvo 142E"
```

Subsetting Rows and Columns

To subset a specific column, or variable, use the `$` operator. For example, we can extract the `mpg` column (miles per gallon) from `mtcars`:

```
mtcars$mpg
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

Notice that when we subset a column this way, we are actually extracting a vector. Therefore, we can use some of the previously mentioned vector functions.

```
summary(mtcars$mpg)
```

```
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
##  10.40   15.43   19.20   20.09   22.80   33.90
```

Columns and rows of a data frame can also be extracted by index (position) using brackets:

```
mtcars[1, ] # extract first row
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4   21   6  160 110   3.9 2.62 16.46  0  1    4    4
```

```
mtcars[, 1] # extract first column
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

```
mtcars[2, 3] # extract element in the second row and third column
```

```
## [1] 160
```

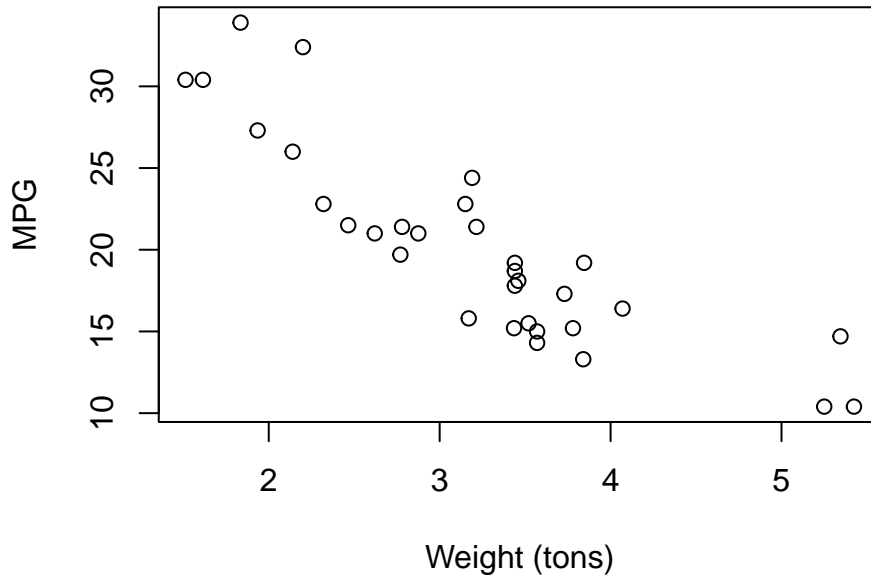
```
mtcars[2:3, ] # extract second and third row
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710   22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

Scatter Plot

To make a scatter plot in R use the `plot()` function. For example, below is scatter plot with car weight on the x -axis and mileage on the y -axis. Each point represents a different car model from the `mtcars` data frame.

```
plot(mpg ~ wt, data = mtcars, xlab = "Weight (tons)", ylab = "MPG")
```



Based on this scatter plot we can see that there is a decreasing and linear relationship between weight and mileage. That is, as weight increases the mileage decreases, on average, for car models in this data set, which makes sense.

That was a lot of information if this is your first time using R. Congratulations! Take a deep breath, you will have the rest of the semester to get comfortable with using R and RStudio. We will review this material in class, and you will see the same R commands and functions repeatedly used in different contexts.

Practice Problems

Not to be collected.

Exercise 1. Use R to find the sum of numbers from 1 to 1000.

Exercise 2. Use `seq()` to express the sequence $3, 6, 9, \dots, 60$ as a vector in R.

Exercise 3. Find the min, max, mean, and median of the car weight variable (`wt`).

Exercise 4. Make a plot of `mpg` versus `cyl`. Label the y -axis “Miles per gallon” and the x -axis “Number of cylinders”. Describe the association between the two variables.