

# First Steps with R and RStudio

Yiheng Wu, Jeff Werst, Matt Steele

---

## Resources

- Official Site for R and Comprehensive R Archive Network (CRAN)
  - Official Site for POSIT/RSTUDIO
  - O'Reilly Learning Platform
    - R for Data Science, 2nd Edition
    - R Programming for Statistics and Data Science
  - Posit/RStudio Youtube Page
  - R for Data Science Youtube Page
- 

## Create a File

You can create new R-focused files that allow you to document your code and the outputs of your code.

- **R Script:** create code based text file that allows you to save and execute code at your discretion

*File > New File > R Script*

- **R Markdown or Quarto Document:** create literate code based files that allow you to write plain text language combined with your R code. This file outputs results from data analysis (including plots and tables) along with written commentary into a nicely formatted and reproducible document

*File > New File > R Markdown*

*File > New File > Quarto Document*

---

## RStudio

RStudio is an Integrated Development Environment (IDE) that allows you to save your code, store your variables and environments and view outputs.

### Source Pane

this pane is opened when you create or open a markdown or script file.

- This area is where you can create code in script or markdown files

---

### Console Pane

This is where you interact with the R. The results of your commands are displayed in this pane.

- Useful for testing code and exploring data

---

### Environment Pane

view functions, objects, and data sets that are stored here

- Your environment can be saved and accessed at any point
  - Save your environment to your working directory
- 

### **Misc Pane**

view files, plots, packages, and get help

---

### **Set Your Preferences**

*Tools > Global Options*

Some suggested Preferences to set:

- Code > Editing > Use Native Pipe Operator
  - Code > Editing > Soft wrap source R files
- 

### **Working Directory**

The working directory in R is the folder where you are working. Hence, it's the place (the environment) where you have to store your files of your project in order to load them or where your R objects will be saved.

---

### **Function: setwd()**

You can set your working directory in RStudio by going to

*Session > Set Working Directory > Choose Directory*

- Choose the folder you want to use for your current project

*HINT: After you set your working directory, save the path by copy and pasting the file path from the console area into the source area using the `setwd()` function*

**Function: `getwd()`**

See the current directory you are in.

```
getwd()
```

---

## Part 2: Tips Before You Get Started

---

### Keyboard Shortcuts

Since R is heavily focused on using code to execute commands it is useful to become familiar with keyboard shortcuts

*Tools > Keyboard Shortcuts Help*

**Popular Shortcuts:**

	PC	MAC
<b>Run Code</b>	CTRL + ENTER	CMD + RETURN
<b>Assignment Operator</b>	ALT + -	OPTION + -
<b>Pipe Operator</b>	CTRL + SHIFT + M	CMD + SHIFT + M

---

## Commenting

Since you will be performing several operations in a single document and even in a particular code chunk, it becomes important to document what processes you were performing or make notes to use for yourself or others about your intentions.

You can do this in R using **\*\*comments\***. Entering a hashtag (#) into your code will comment anything that comes after for one single line. Entering 3 single quotations ('''') will comment out multiple lines.

```
# get the mean of the miles per gallon variable for all cars in the dataset mtcars  
mean(mtcars$mpg)
```

---

## Part 3: Coding Basics

---

### Built-in Functions and Arguments

#### Functions:

Like most computer software, R allows you to run commands. Commands in R are referred to as functions. There are several built-in functions in base R and when you install and call a new package, you will have access to more functions that you can use.

These built-in functions are ready-made tools like you would have a toolbox, and like how these different tools perform various tasks like measuring, cutting, or fixing things, different functions perform different tasks like filtering, sorting, or calculation.

```
sample(1:20)
```

#### Arguments:

Arguments are the values and parameters that are acted on by the function. They are the information that you give to a function to tell it what to do.

```
sample(1:20, size = 10, replace = TRUE)
```

*# arguments in a function have an order. Entering information in the order will allow you to skip the argument name*

```
sample(1:20, 10, TRUE)
```

---

## Documentation

- Base R Function Documentation
- For every package you want to use in R, there will be a documentation website or document. Explore these documents to see the available functions and their default arguments
- Get help with a package, function, or argument within R. Using `(?)` or `help("name of function")`

```
?sample
```

```
help("sample")
```

- The **TAB** will auto-suggest options for you in R, when typing a name of a function. Scrolling over the name of the of the function will give you an overview of what it does and the default/available arguments of the function.

---

## Objects

Objects allow you to store and work on data (numbers, words, tables, and more).

Table 2: **Examples**

---

numeric value

numValue <- 400

character  
results of running function(s)  
vector  
data frame

```
chrValue <- "Hello World"  
resultFunction <- mean(mtcars$mpg)  
vecValue <- c(1,2,3)  
dfValue <- read.csv("mtcars.csv")
```

---

## Assignment Operator

The assignment operator (<-) allows you to create an object.

Table 3: **Keyboard Shortcut**

	PC	MAC
Assignment Operator	ALT + -	CMD + -

```
a <- 35  
b <- 45  
  
a  
a + b  
  
sample_result <- sample(1:10)  
  
sample_result
```

---

## Naming Conventions

- Use descriptive and meaningful names that indicate the purpose of the object
- Use lowercase letters.
- Use underscores to separate words (e.g., my\_variable\_name).

- Avoid using reserved words or functions (e.g., “if,” “else,” “for,” “function”).

---

## Data Types

Categories for different kinds of information

<b>Double or Numeric</b>	used for numbers which can be integers (whole numbers) or real numbers (numbers with decimal points).
<b>Character</b>	used for text, words, and strings of characters. Enclose in double (“”) or single (‘’) quotes.
<b>Factor</b>	used to represent categorical data with predefined levels.
<b>Date</b>	used for handling dates, times, and time intervals.
<b>Boolean</b>	used for decision-making and represented by binary values, typically <b>TRUE</b> or <b>FALSE</b>

```
# double
str(400)

# integer
str(400L)

# character
str("Hello World")

# factor
str(as.factor(mtcars$cyl))

# boolean
```



```
str(TRUE)
```

---

## Packages

R packages are like toolkits or collections of pre-built functions, data sets, and tools that extend the capabilities of the R programming language.

---

### Install Packages

You must install a package before you can load it. *But you only need to install it one time.*

```
install.packages("tidyverse")
```

### Load Packages

For every new session, you must load it to use the package's functions.

```
library(tidyverse)
```

---

## Part 4: Entering Data

---

## Vectors

ordered collections of data items of the same type

*vectorSyntax* <- c(object1, object2, object3)

```
vec_one <- c(1,2,3)
```

```
vec_two <- c(4:6)
```

```
vec_two
```

```
vec_one[1]
```

```
vec_one[1]+10
```

```
vec_three <- c(1,2,3,'hello','world')
```

```
vec_three[1]
```

---

## Data Frame

two-dimensional sequence of data variables (columns) and observations (rows). While each variable in a data frame typically contains data of the same type, different variable can contain different data types.

*dataFrameSyntax* <- data.frame(column1, column2, column3)

```
# create vectors
```

```
title <- c("Star Wars", "The Empire Strikes Back", "Return of the Jedi")
```

```
year <- c(1977, 1980, 1983)
```

```
length.min <- c(121, 124, 133)
```

```
box.office.mil <- c(787, 534, 572)
```

```
# combine these vectors with the data.frame() function
```

```
starWars.data <- data.frame(title, year, length.min, box.office.mil)
```

```
starWars.data
```

---

## Part 5: Working with Data Frames

---

### Subsetting Variables

allows you to select and work with specific variables (columns) from a data frame.

*object\_name\$variable\_name*

```
starWars.data$year
```

---

### Export the Data Frame

Once you are done entering your data, you can export it to your working directory. The function without built-in arguments is **write.table( )** but if are saving it as a csv, you are better using **write.csv( )**.

*write.csv(object, "name of file.extension")*

```
write.csv(starWars.data, "starwars.csv")
```

---

## Part 6: Explore Data

---

## Explore the Data Frame

---

### Function: `view()`

view the full data frame

```
view(starWars.data)
```

### Function: `mean()`, `median()`, `sum()`

individual descriptive statistics

```
# the total box office revenue  
sum(starWars.data$box.office.mil)  
  
#the mean box office revenue  
mean(starWars.data$box.office.mil)  
  
#what is the median box office revenue  
median(starWars.data$box.office.mil)  
  
# what is the standard deviation of the box office revenue  
sd(starWars.data$box.office.mil)
```

---