First Steps with R and RStudio

Matt Steele

2023-10-25

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
- Postit/RStudio Youtube Page
- R for Data Science Youtube Page

Part 1: Getting Started

Why use R

- It is open-source
    - Free
    - Platform independent
    - Reproducible
    - Shareable
    - Contains add-on packages
- Created for data statistical computation and graphic export

---

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 you 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: getwd()

See the current directory you are in.

```
getwd()
```

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

```
setwd("C:/Users/Matt/Documents/GitHub/First_steps_r_fall23")
```

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
Run Code	CTRL + ENTER	CMD + RETURN
Assignment Operator	ALT + -	OPTION + -
Pipe Operator	CTRL + SHIFT + M	CMD + SHIFT + M

Commenting

Since use 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 hastag (#) 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 mpg variable

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, however, 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 function perform differnt tasks like filtering, sorting, or calculation.

```
sample(1:5000)
```

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(1:5000, size = 50, replace = TRUE)

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

sample(1:5000, 50, 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")

```
?dplyr

help("mean")
```

- 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 is does and the default/available arguments of the function.

Objects

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

Examples	
numeric value	numValue <- 400

character	chrValue <- “Hello World”
results of running function(s)	resultFunction <- mean(mtcars$mpg)
vector	vecValue <- c(1,2,3)
data frame	dfValue <- read.csv(“mtcars.csv”)

Assignment Operator

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

Keyboard Shortcut

	PC	MAC
Assignment Operator	ALT + -	CMD + -

```
a <- 35
b <- 45

a
a + b
```

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

Double or Numeric	used for numbers which can be integers (whole numbers) or real numbers (numbers with decimal points).
Character	used for text, words, and strings of characters. Enclose in double (“”) or single (’’) quotes.
Factor	used to represent categorical data with predefined levels.

Date	used for handling dates, times, and time intervals.
Boolean	used for decsion-making and represented by binary values, typically `TRUE` or `FALSE`

```
# 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
```

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

```
# if you have not yet loaded tidyverse let's go ahead and do that

library(tidyverse)
```

Read a Data Frame into R

load data from a file in your working directory using the **read_csv( )** function from tidyverse. There is also the **read.csv( )** function in base R.

read_csv(“name of file.extension)

```
fight_songs <- read_csv("fight-songs.csv")
```

Explore the Data Frame

Function: view()

view the full data frame

```
view(fight_songs)
```

Function: colnames()

view the names of columns/variables

Function: glimpse()

view the structure of data

```
glimpse(fight_songs)
```

Function: summary()

basic descriptive statistics

```r
summary(fight_songs)
```

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

individual descriptive statistics

```r
# number of times the word "fight" is used in all songs

sum(fight_songs$number_fights)

# what is the average number of times "fight" is used in all songs

mean(fight_songs$number_fights)

# what is the median number of times "fight" is used in all songs

median(fight_songs$number_fights)

# what is the standard deviation of times "fight" is used in all songs

sd(fight_songs$number_fights)
```

Extras

- Psych Package

```r
install.packages("psych")
library(psych)
```

```r
describe(fight_songs)
```

- DataExplorer package

```r
install.packages("DataExplorer")
library(DataExplorer)
```

```r
create_report(fight_songs)
```