R for Data Analysis

Trevor French

8/16/2022

Table of contents

Part I

Introduction

> "There is synthesis when, in combining therein judgments that are made known to us from simpler relations, one deduces judgments from them relative to more complicated relations. There is analysis when from a complicated truth one deduces more simple truths." -André-Marie Ampère (Hofmann 1996)

Everyone is a data analyst. The purpose of this book is to inspire and enable anyone who reads it to reconsider the methods they currently employ to analyse data. This is not to suggest that the methodologies outlined will be useful or sufficient for everyone who reads it. Some analyses can be performed quickly without the need for additional computation while others will require advanced analytics techniques not outlined in this book; however, the aspiration is that all will be equipped with novel tools and ideas for approaching data analysis.

Prerequisites

No prior knowledge is required to begin this book. The content will start at the very beginning by showing you how to set up your R environment and the basics of programming in R. By the end of the book, you will be able to perform intermediate analytics techniques such as linear regresion and automatic report generation.

You will need an environment which you use to run your code. It is recommended that you download R and R Studio locally for this requirement. This book will walk you through how to do that as well as offer alternatives if that is not an option for you.

Structure of the Book

- Part I (Fundamentals) will introduce you to the basics of programming in the context of R.
- Part II (Data Acquisition) will teach you how to create, import, and access data.
- Part III (Data Preparation) will show you how to begin preparing your data for analysis.
- Part IV (Developing Insights) goes through the process of searching for and extracting insights from your data.
- Part V (Reporting) demonstrates how to wrap your analysis up by developing and automating reports.

Each part will be concluded with practical exercises for you to test your skills.

1 What is R?

> R was a programming language that was designed specifically for the needs of statistics and data analysis. -Hadley Wickham (Hermans 2021)

R is a statistical programming language used commonly for data analysis amongst other disciplines. It was built by Ross Ihaka and Robert Gentleman at the University of Auckland and was first released in 1993.

1.1 History

Robert Gentleman and Ross Ihaka "both had an interest in statistical computing and saw a common need for a better software environment in [their] Macintosh teaching laboratory. [They] saw no suitable commercial environment and [they] began to experiment to see what might be involved in developing one [them]selves." (Ihaka 1998)

While R was officially first released in 1993, it wasn't until 1995 that Ross Ihaka and Robert Gentlemann were convinced by Martin Mächler to release the source code freely (Ihaka 1998).

1.2 Resources

You can learn more about R at https://www.r-project.org/

Read Ross Ihaka's account of R's origination here https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf

"What is R?"" by Microsoft https://mran.microsoft.com/documents/what-is-r

R manuals by the R Development Core Team https://cran.r-project.org/manuals.html

R-bloggers https://www.r-bloggers.com/

R User Groups https://www.meetup.com/pro/r-user-groups/

R Studio Community https://community.rstudio.com/

The R Journal https://journal.r-project.org/

Microsoft R Application Network https://mran.microsoft.com/

2 What is Data Analysis?

> I mean my definition is data science is like data analysis by programming. Which of course begs the question of what data analysis is, and so I think of data analysis as really any activity where the input is data and the output is understanding or knowledge or insights. So I think of that pretty broadly. And then to do data science you're not doing it by pointing and clicking. You're doing it by writing some code in a programming language. -Hadley Wickham (Eremenko 2020)

Data analysis at it's most simple form is the process of searching for meaning in data with the ultimate goal being to draw insight from that meaning.

2.1 The Process of Data Analysis

The process of data analysis can be generally described in six steps:

1. **Gathering Requirements** - Before one embarks on an analysis, it's important to make sure the requirements are understood. Requirements include the questions which your stakeholders are hoping to answer as well as the technical requirements of how you are going to perform your analysis.
2. **Data Acquisition** - As you might imagine, you must acquire your data before conducting an analysis. This may be done through the methods such as manual creation of datasets, importing pre-constructed data, or leveraging APIs.
3. **Data Preparation** - Most data will not be received in the precise format you need to begin your analysis. The process of data preparation is where you will structure and add features to your data.
4. **Developing Insights** - Once your data is prepared, you can now begin to make sense of your data and develop insights about it's meaning.
5. **Reporting** - Finally, it's important to report on your data in such a way that the information is able to be digested by the people who need to see it when they need to see it.

Other sources may include additional steps such as "acting on the analysis". While this is a critical step for organizations to capture the full value of their data, I would argue that it occurs outside of the *analysis* process.

This book will focus on the technical skills required to conduct an analysis. Because of this, we will be covering steps two through five.

2.2 Resources

Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data https://onlinelibrary.wiley.com/doi/book/10.1002/9781119183686

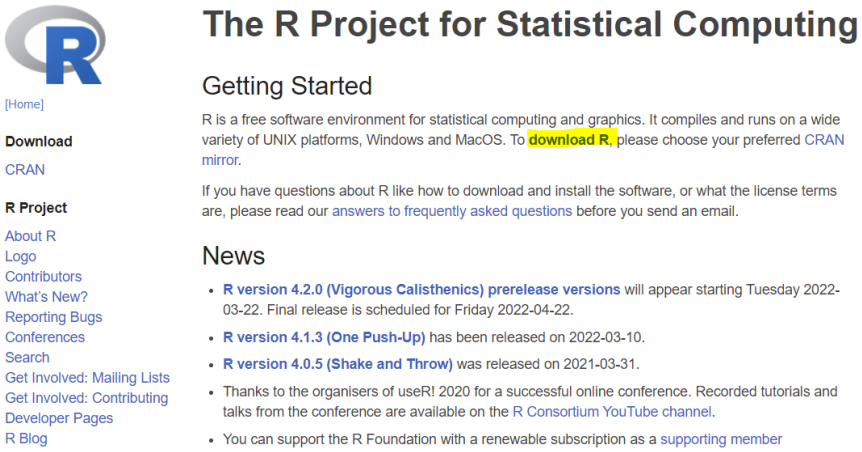
Managing the Analytics Life Cycle for Decisions at Scale https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/manage-analytical-life-cycle-continuous-innovation-106179.pdf

3 Setup

3.1 Install R

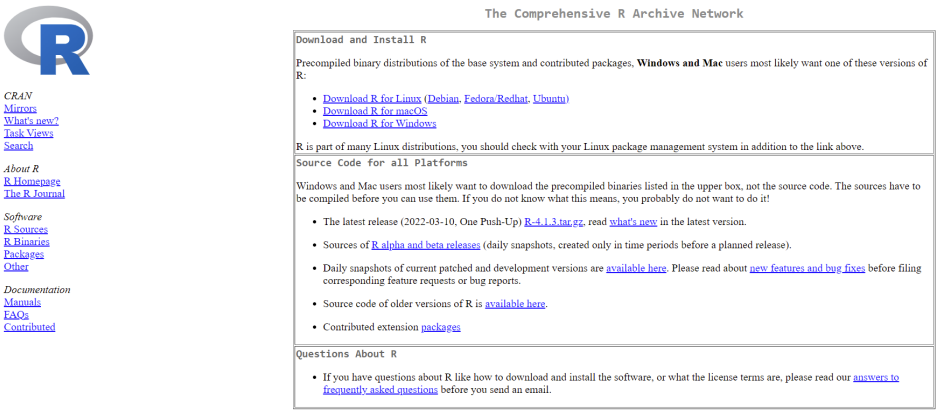
Before you do anything, you'll need to download R. This download will allow your computer to interpret the R code you write later on.

1. Download R From R: The R Project for Statistical Computing
2. Select "download R"

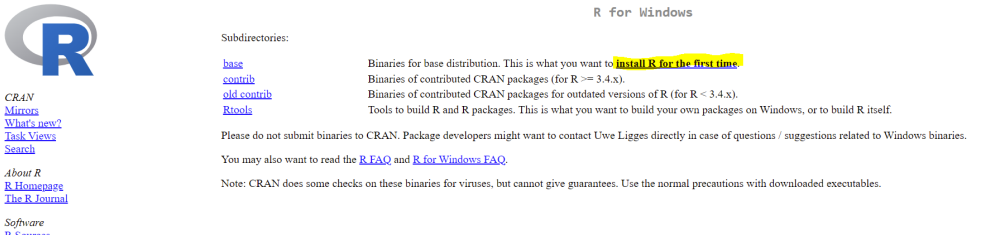


3. Choose any link but preferably the one closest to your physical location

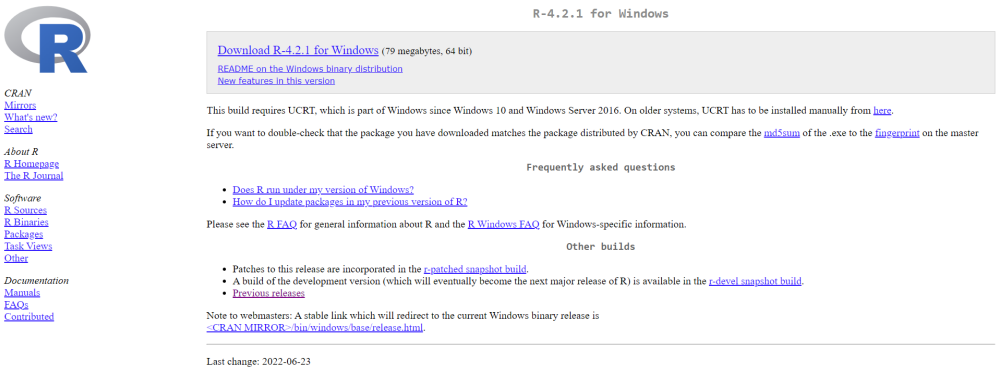
4. Choose your operating system



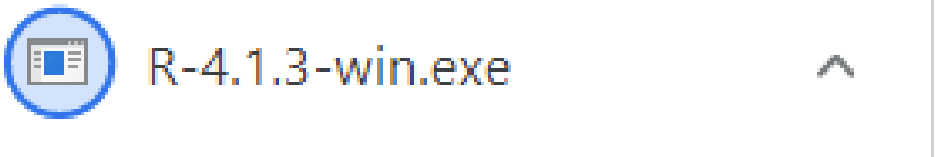
5. Press “Install R for the first time”



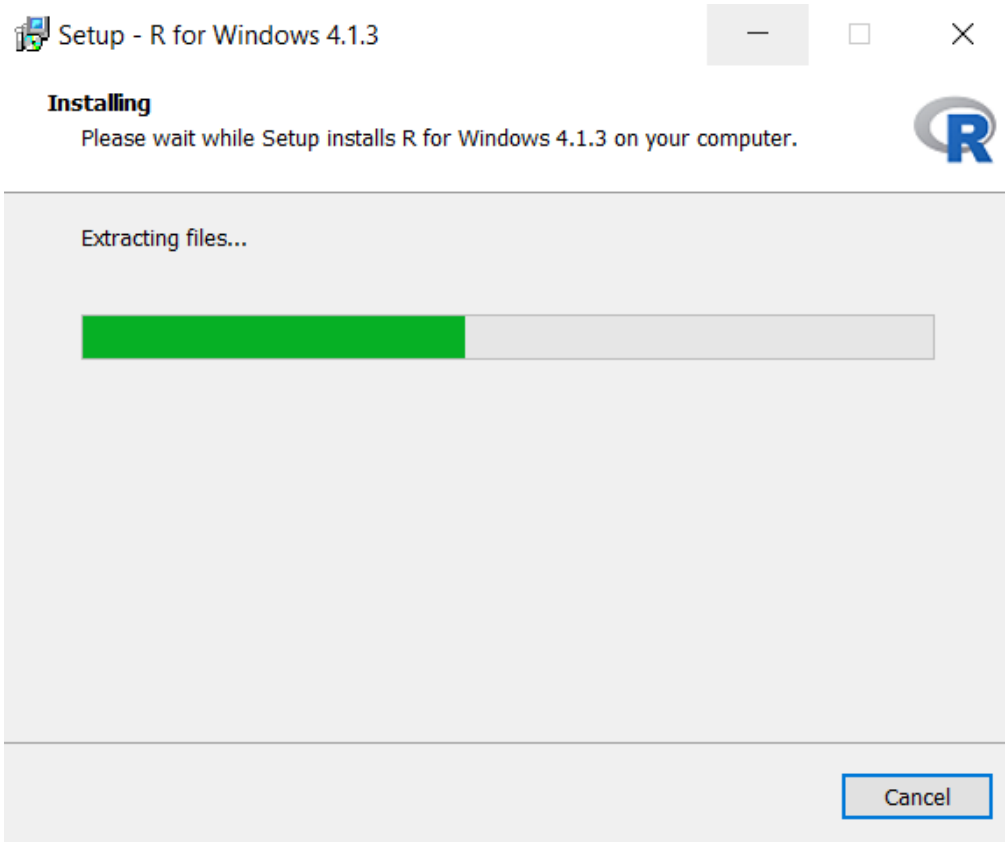
6. Press “download”



7. Open installer



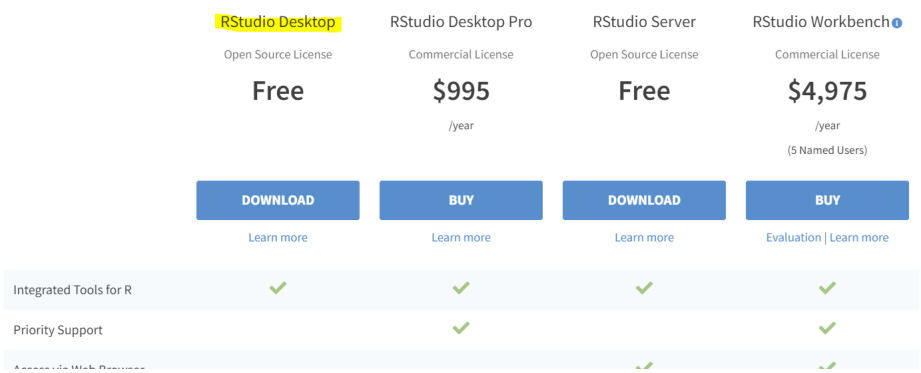
8. Follow the prompts and leave all options set as their default values



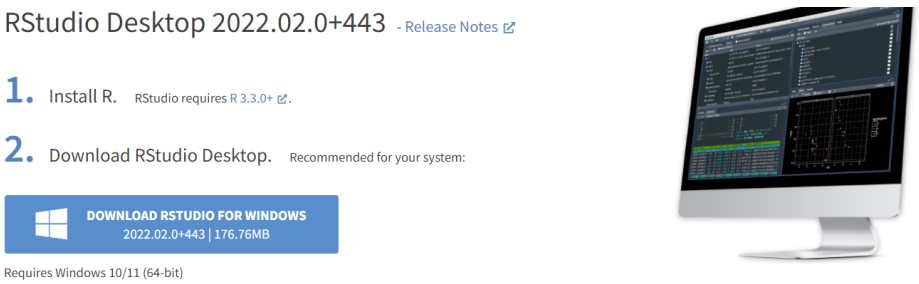
3.2 Install R Studio

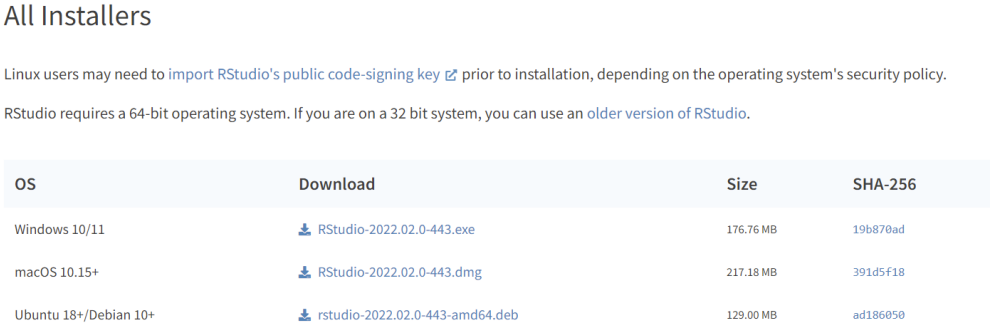
After you install R, you'll need an environment to write and run your code in. Most people use a program called "R Studio" for this. To download R Studio follow the steps listed below:

1. Navigate to the R Studio download site: Download the RStudio IDE
2. Press the "download" button under RStudio Desktop



3. Choose the download option for your operating system



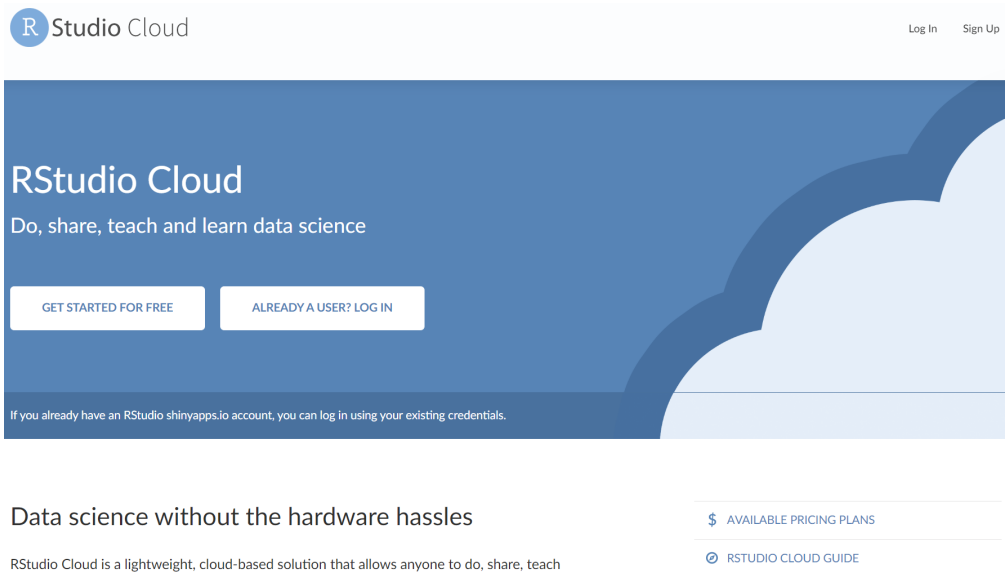


4. Open the installer and accept all defaults

3.3 Alternatives

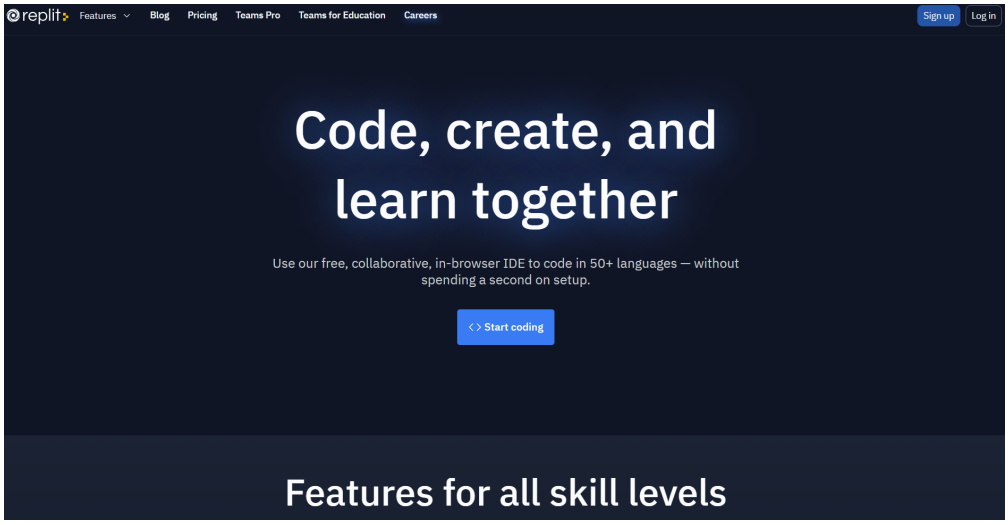
3.3.1 R Studio Cloud

R Studio Cloud offers users a way to replicate the full R Studio experience without having to download or set anything up on your personal computer. You can sign up for a free account here:



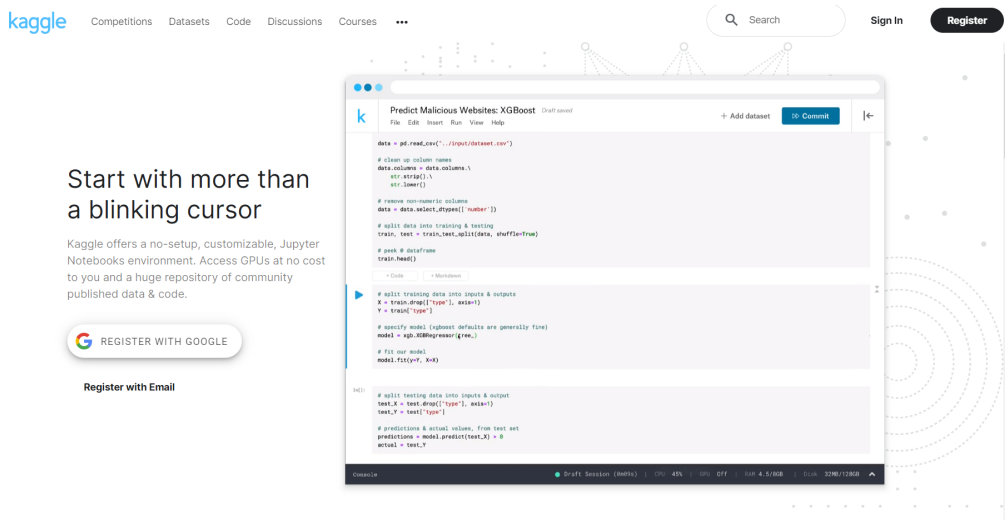
3.3.2 Replit

Replit allows users to code in 50+ languages in the browser. While you won't be able to follow along with the R Studio specific examples, you will be able to run R code. You can sign up for a free account here:



3.3.3 Kaggle

Kaggle is one of the most popular sites for data analysts to compete in data competitions, find data, and discuss data topics. They also have a feature that allows you to write and run R (and Python) code. You can sign up for a free account here:



Part II

Part I: Fundamentals

This section will walk you through the fundamentals of R including:

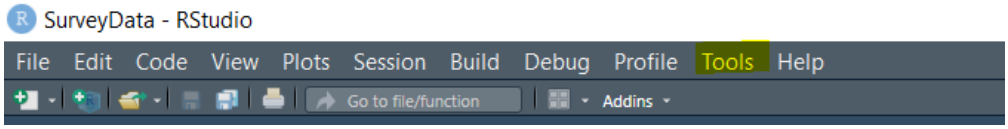
- **Getting Familiar with R Studio**- learn about the four window panes in the R Studio IDE and what you can do in each of them.
- **Programming Basics**- pick up a few of the fundamental principals of programming and the syntax that is unique to the R language.
- **Data Types**- learn about the data types that exist in R.
- **Data Structures**- learn about the structures in which you can store your data in R. 1.2 Structure of the Book Part I (Fundamentals) will introduce you to the basics of programming in the context of R. Part II (Data Acquisition) will teach you how to create, import, and access data. Part III (Data Preparation) will show you how to begin preparing your data for analysis. Part IV (Developing Insights) goes through the process of searching for and extracting insights from your data. Part V (Reporting) demonstrates how to wrap your analysis up by developing and automating reports. Each part will be concluded with practical exercises for you to test your skills.

4 Getting Familiar with R Studio

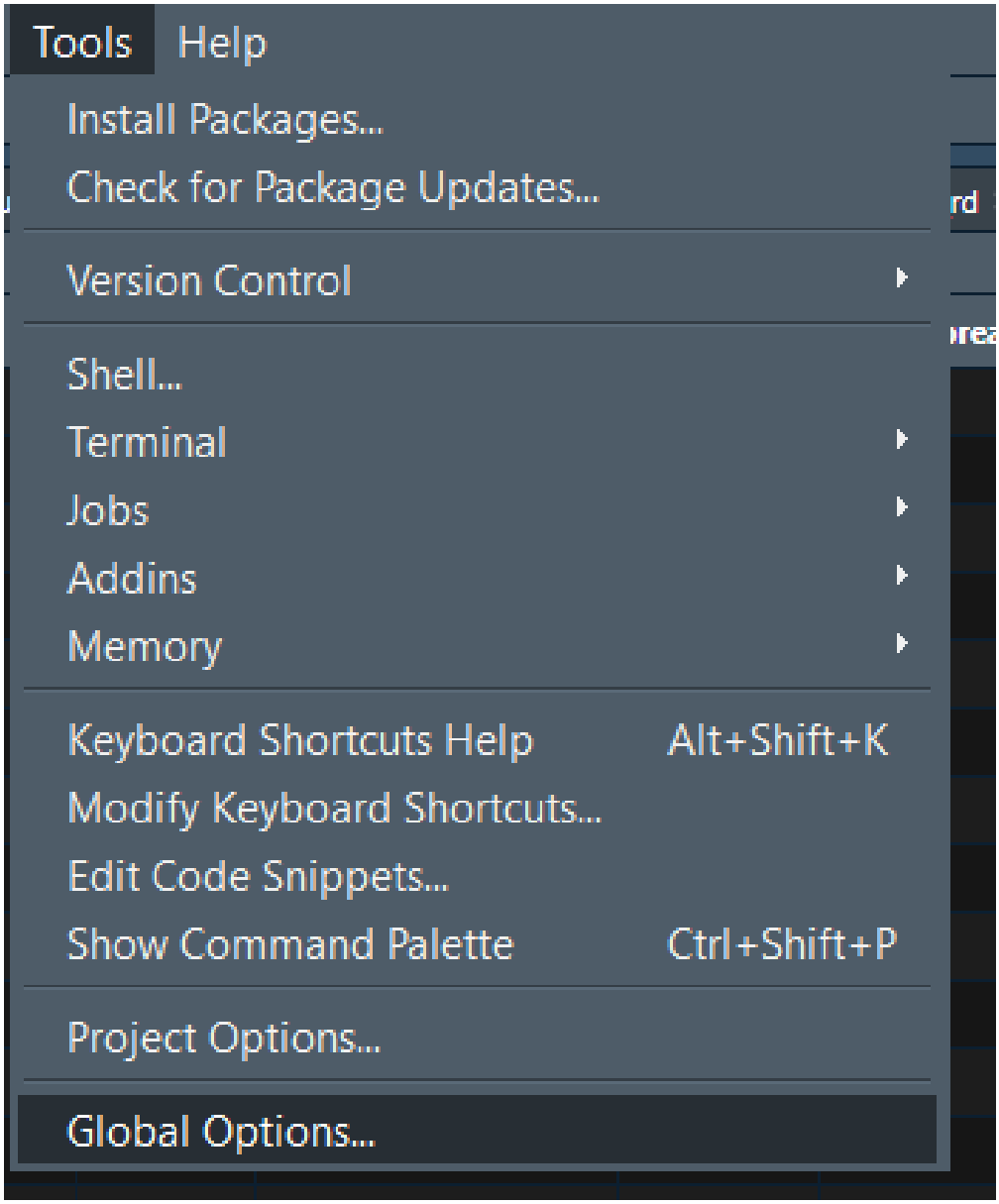
4.1 Customization

You are able to customize how your version of R Studio looks by following these steps:

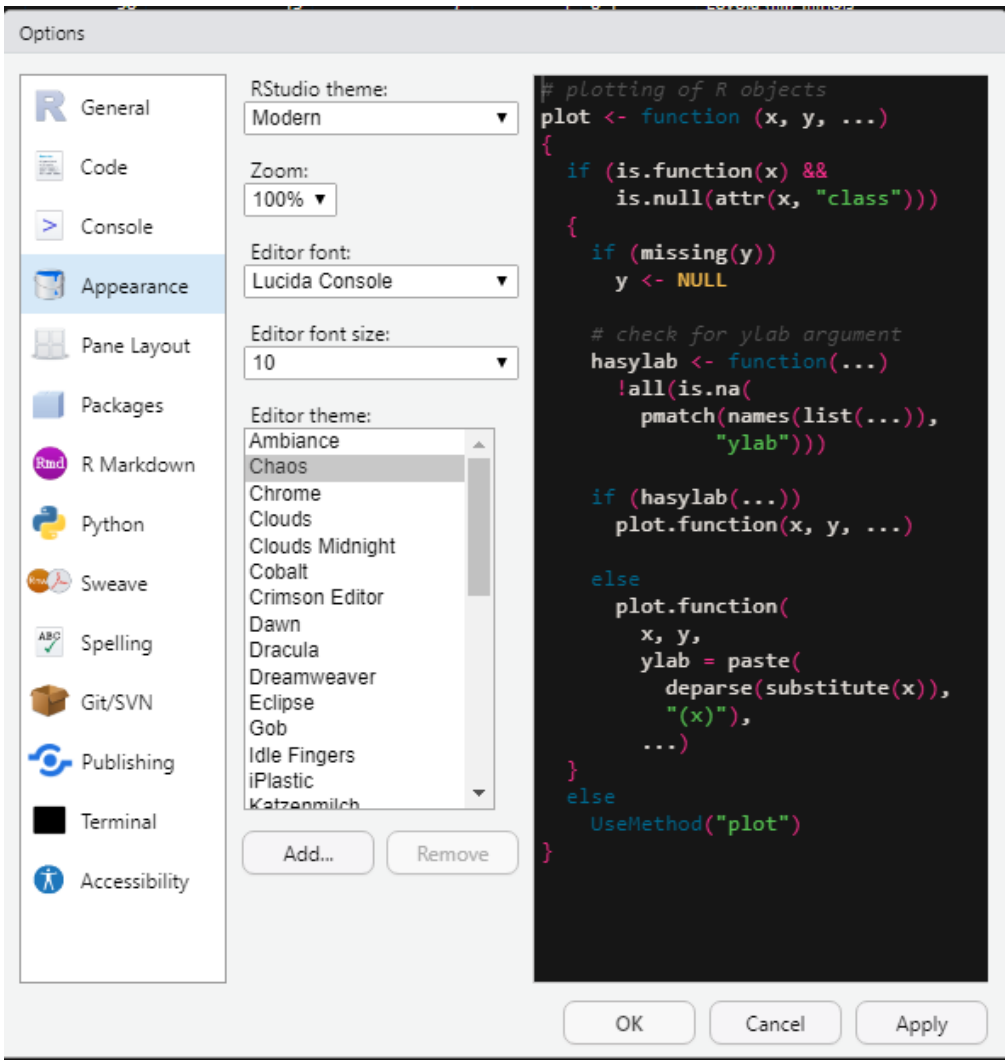
1. Open R Studio and choose 'tools' from the toolbar



2. Choose 'Global Options'



3. Choose 'Appearance' and select your favorite theme from the 'Editor Theme' section

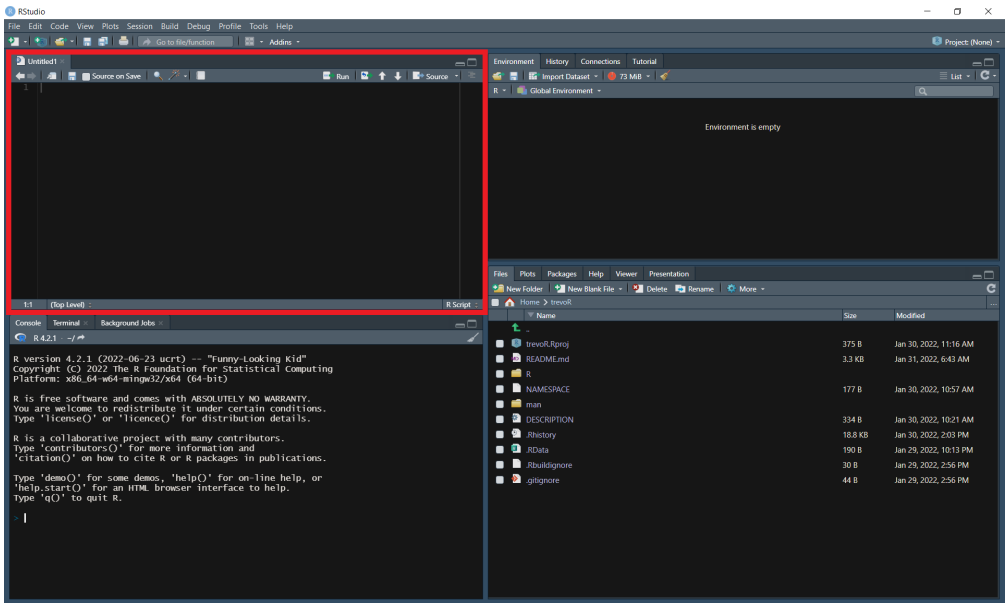


4. Press 'Apply'

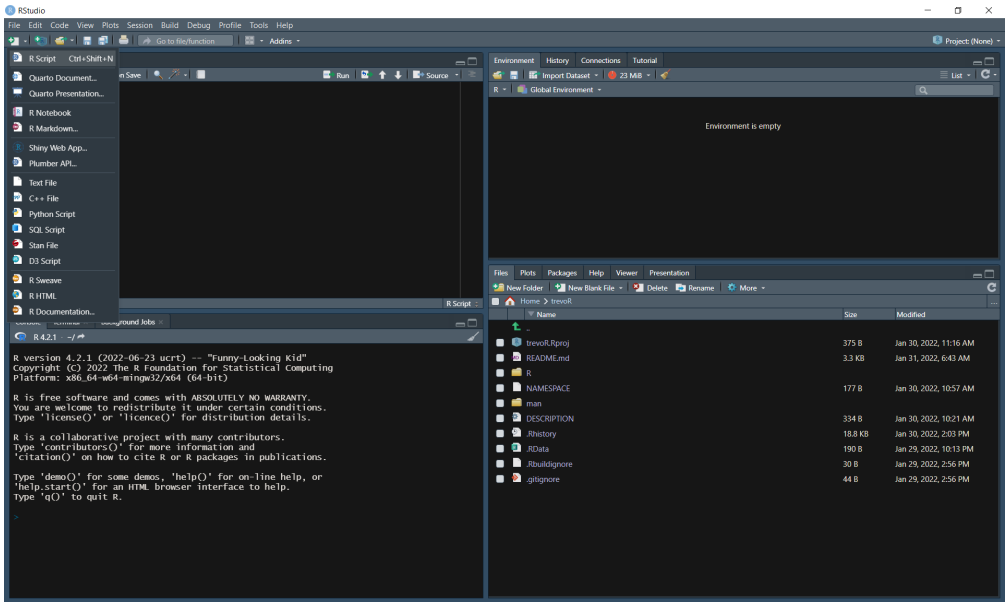
There are other customization options avaialable as well. Feel free to explore the "Global Options" section to make your version of R Studio your own.

4.2 Source Pane

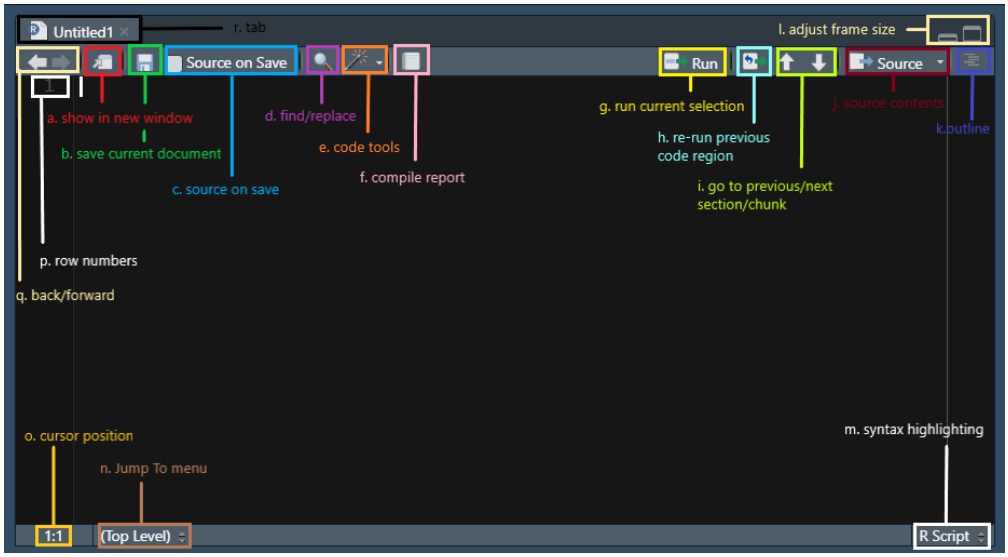
The source pane is the top left pane in R Studio. This is where you will write and edit your code.



If you don't see the source pane, you may need to create a new R Script by pressing "Ctrl + Shift + N" ("Cmd + Shift + N" on Mac) or by selecting "R Script" from the "New File" dropdown in the top left corner.



Each element of the source pane is outlined below.



a. **Show in New Window**- This allows you to pop the source pane into a new window by itself.
b. **Save Current Document**- This saves the file contained in the tab you currently have active.
c. **Source on Save**- Automatically sources your file every time you hit save. "Sourcing" is similar to "Running" in the sense that both will execute your code; however, sourcing will execute your saved file rather than sticking lines of code into the console.
d. **Find/Replace**- this feature allows you to find and replace specified text, similar to find and replace features in other tools such as Excel.
e. **Code Tools**- This brings up a menu of options which help you to code more efficiently. Some of these tools include formatting your code and help with function definitions.
f. **Compile Report**- This allows you to compile a report directly from an R script without needing to use additional frameworks such as R Markdown.
g. **Run Current Selection**- This allows you to highlight a portion of your code and run only that portion.
h. **Re-run Previous Code Region**- This option will execute the last section of code that you ran.
i. **Go to Previous/Next Section/Chunk**- These up and down arrows allow you to navigate through sections of your code without needing to scroll.
j. **Source Contents**- This option will save your active document if it isn't already saved and then source the file.
k. Outline- Pressing this option will pop open an outline of your current file.
l. **Adjust Frame Size**- These two options will adjust the size of the source pane inside of R Studio.
m. **Syntax Highlighting**- This allows you to adjust the syntax highlighting of your active document to match the highlighting of other file types.
n. **"Jump To" Menu**- This menu allows you to quickly jump to different sections of your

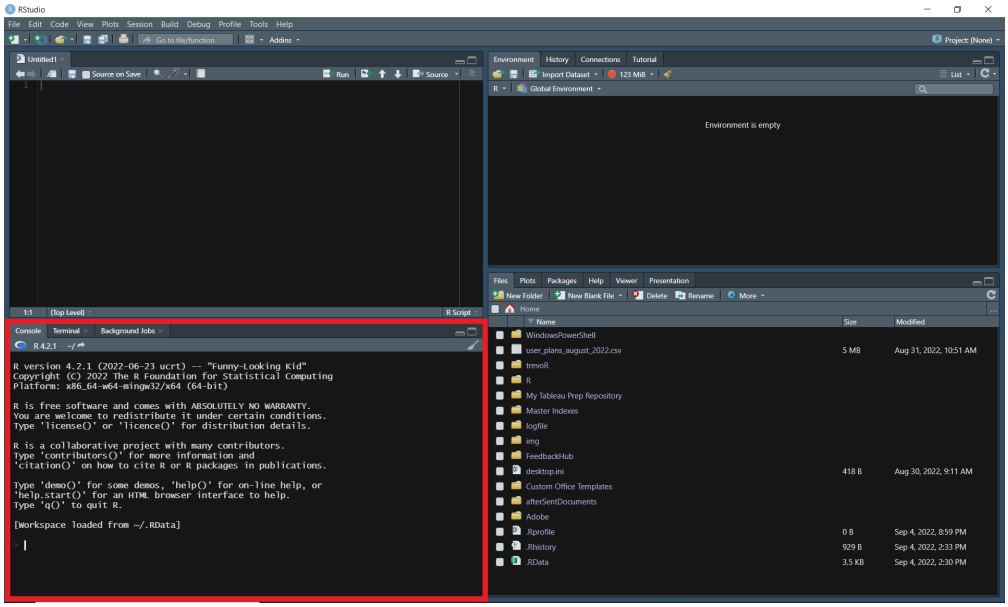
code.

o. **Cursor Position**- This displays your current cursor position by row and column.
p. **Row Numbers**- The lef-thand side of your document will display the row number for each line of your code.
q. **Back/Forward**- These arrows are navigation tools that will allow you to redo/undo the following actions: opening a document (or switching tabs), going to a function definition, jumping to a line, and jumping to a function using the function menu (Paulson 2022).
r. **Tab**- This is a tab in the traditional sense, meaning you are able to have a collection of documents open displayed as tabs. These tabs will have the title of your document and often an icon of some sort to demonstrate the file type.

4.3 Console

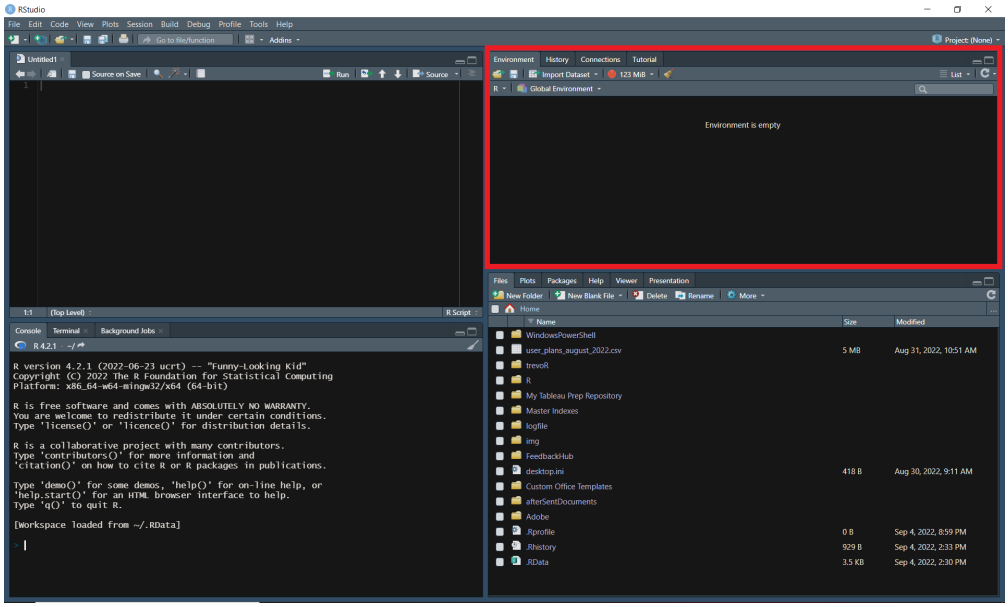
The console pane is the bottom left pane in R Studio. This pane has three tabs: "Console", "Terminal", and "Background Jobs".

- The "Console" tab is where you will be able to run R code directly without writing a script (this will be covered in the next chapter).
- The "Terminal" tab is the same terminal you have on your computer. This can be adjusted in the global options.
- The "Background Jobs" tab is where you can start and manage processes that need to run behind the scenes.



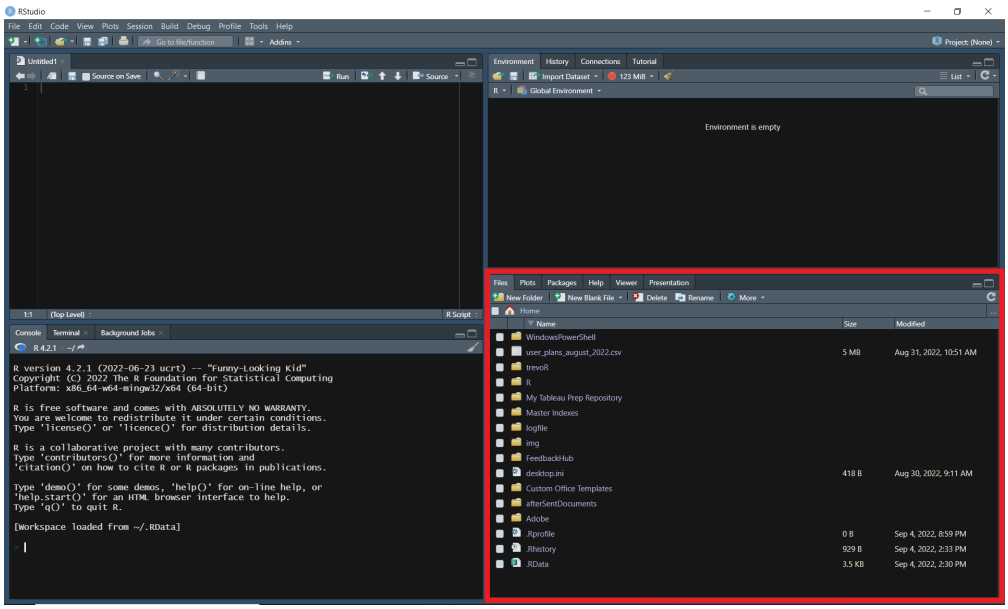
4.4 Environment

The environment pane is the top right pane in R Studio. This is where you will manage all things related to your development environment.



4.5 Files

The files pane is the bottom right pane in R Studio.



4.6 Resources

This is where the difference between sourcing and running is.

https://support.rstudio.com/hc/en-us/articles/200484448-Editing-and-Executing-Code

https://support.rstudio.com/hc/en-us/articles/200484568-Code-Folding-and-Sections-in-the-RStudio-IDE

()[https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts-in-the-RStudio-IDE]

Back and forward

https://support.rstudio.com/hc/en-us/articles/200710523-Navigating-Code-in-the-RStudio-IDE

5 Programming Basics

5.1 Executing Code

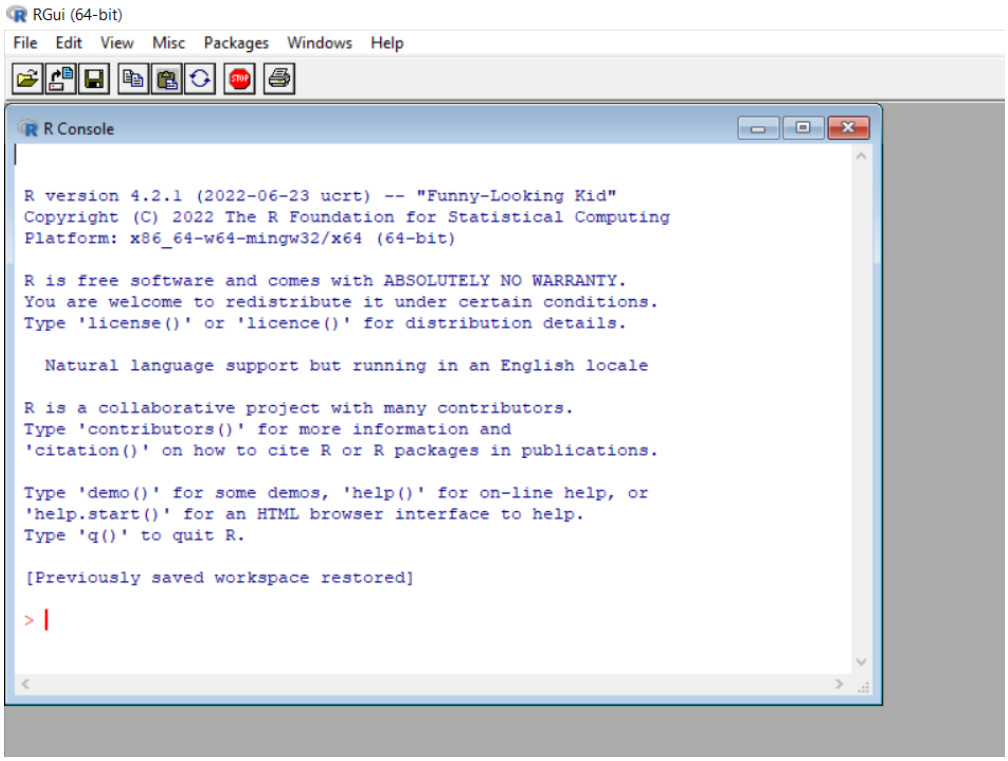
When working in most programming languages, you will generally have the option to execute code one of two ways:

- in the console
- in a script

5.1.1 Console

The first way to run code is directly in the console. If you're working in R Studio, you will access the console through the "console" pane.

Alternatively, if you downloaded R to your personal computer, you will likely be able to search your machine for an app named "RGui" and access the console this way as well.



In the following example, the text "print(3+2)"" is typed into the console. The user then presses enter and sees the result: "[1] 5".

```
print(3+2)
```

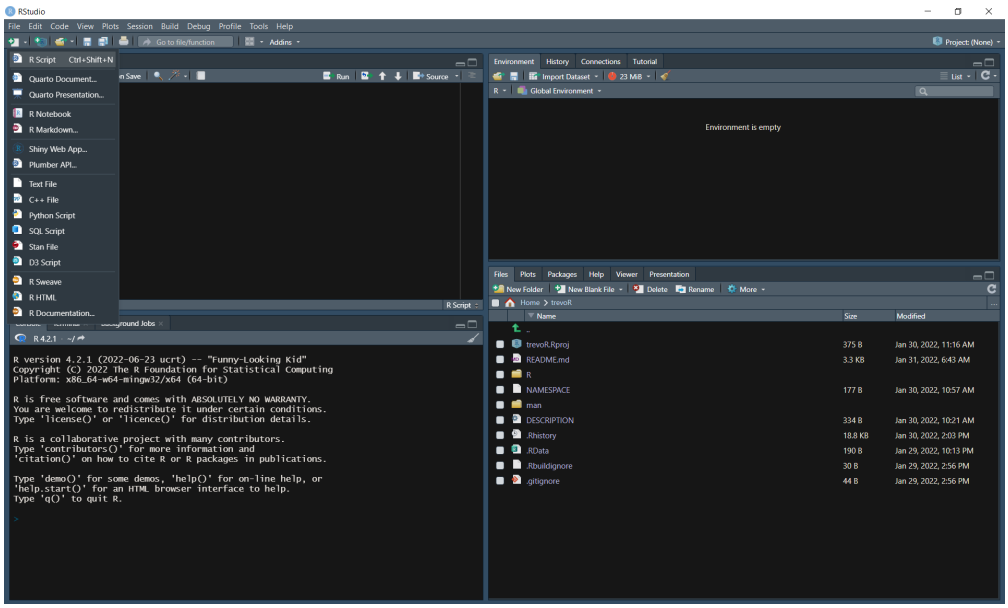
```
[1] 5
```

You may be wondering what "[1]" represents. This is simply a line number in the console and can be ignored for most practical purposes. Additionally, most of the examples in this book will be structured in this way: formatted code immediately followed by the code output.

5.1.2 Script

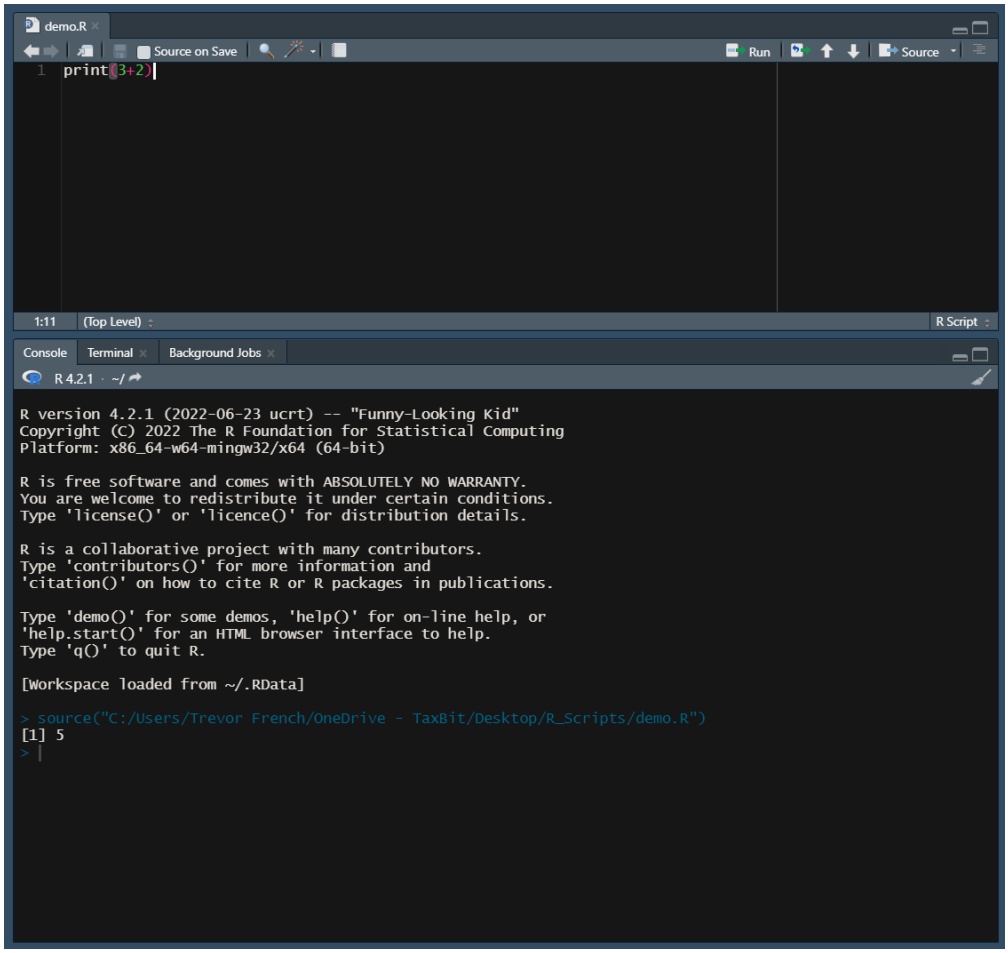
You likely will be using scripts most of the time when working in R. A script is just a file that allows you to type out longer sequences of code and execute them all at once.

For those of you following along in R Studio, you can create a script by pressing "Ctrl + Shift + N" on Windows or by selecting "R Script" from the "New File" dropdown in the top left corner.



From here you can type the same command from before into the source pane. Next, you'll want to save your file by pressing "Ctrl + S" on Windows or by selecting "Save" from the "File" dropdown in the top left corner. Now just give your file a name and your file will automatically be saved as a ".R" file.

Finally, run your newly created R script by pressing the "source" button.



5.2 Comments

Comments are present in most (if not all) programming languages. They allow the user to write text in their code that isn't executed or read by computers. Comments can serve many purposes such as notes, instructions, or formatting.

Comments are created in R by using the "#" symbol. Here's an example:

```
# This is a comment
print(3+2)
```

```
[1] 5
```

Some programming languages allow you a "bulk-comment" feature which allows you to quickly comment out multiple consecutive lines of text. However, in R, there is no such option. Each line must begin with a "#" symbol, as such:

```
# This is the first line of a comment
# This is the second line of a comment
print(3+2)
```

```
[1] 5
```

Comments don't have to start at the beginning of a line. You are able to start comments anywhere on a line like in this example:

```
print(3+2) # This comment starts mid-line
```

```
[1] 5
```

5.3 Variables

Variables are used in programming to give values to a symbol. In the following example we have a variable named "rate" which is equal to 15, a variable named "hours" which is equal to 4, and a variable named "total_cost" which is equal to rate * hours.

```
rate <- 15
hours <- 4
total_cost <- rate * hours
print(total_cost)
```

```
[1] 60
```

5.4 Operators

The following image demonstrates the operators that are available to you in R.

EXPLAIN ALL OF THE OPERATORS HERE (NOT JUST THE IMAGE AS AN EXPLANATION)

Operators

Arithmetic	
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus
%/%	Integer Division

Comparison	
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Misc	
:	Creates a series of numbers in a sequence
%in%	Checks if element exists in vector
%*%	Matrix multiplication

Logical	
&	Vectorized AND operator
&&	AND
|	Vectorized OR operator
||	OR
!	NOT

Assignment	
<-, ->	local
<<-, ->>	global

5.5 Functions

Functions allow you to execute a predefined set of commands with just one command. The syntax of functions in R is as follows.

```
# Create a function called function_name
function_name <- function() {
  print("Hello World!")
}

# Call your newly created function
function_name()
```

```
[1] "Hello World!"
```

To go one step further, you can also add "arguments" to a function. Arguments allow you to pass information into the function when it is called. Here's an example:

```
# Create a function called add_numbers which will add
# two specified numbers together and print the result
add_numbers <- function(x, y) {
    print(x + y)
}

# Call your newly created function twice with different inputs
add_numbers(2, 3)
```

```
[1] 5
```

```
add_numbers(50, 50)
```

```
[1] 100
```

Finally, you can return a value from a function as such:

```
# Create a function called calculate_raise which multiplies
# base_salary and annual_adjustment and returns the result
calculate_raise <- function(base_salary, annual_adjustment) {
    raise <- base_salary * annual_adjustment
```

```
    return(raise)
}

# Calculate John's raise
johns_raise <- calculate_raise(90000, .05)

#Calculate Jane's raise
janes_raise <- calculate_raise(100000, .045)

print("John's Raise:")
```

```
[1] "John's Raise:"
```

```
print(johns_raise)
```

```
[1] 4500
```

```
print("Jane's Raise:")
```

```
[1] "Jane's Raise:"
```

```
print(janes_raise)
```

```
[1] 4500
```

5.6 Loops

There are two types of loops in R: while loops and for loops.

5.6.1 While Loops

While loops are executed as follows:

```
# Set i equal to 1
i <- 1
```

```
# While i is less than or equal to three, print i
# The loop will increment the value of i after each print
while (i <= 3) {
  print(i)
  i <- i + 1
}
```

```
[1] 1
[1] 2
[1] 3
```

Additionally, you can add 'break' statements to while loops to stop the loop early.

```
i <- 1

while (i <= 10) {
    print(i)
    if (i == 5) {
        print("Stopping halfway")
        break
    }
    i <- i + 1
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] "Stopping halfway"
```

5.6.2 For Loops

For loops are executed as follows:

```
employees <- list("jane", "john")

for (employee in employees) {
  print(employee)
```

```
}
```

```
[1] "jane"
[1] "john"
```

5.7 Conditionals

You are also able to execute a command if a condition is met by using "if" statements.

```
if (2 > 0) {
    print("true")
}
```

```
[1] "true"
```

You can add more conditions by adding "else if" statements.

```
if (2 > 3) {
    print("two is greater than three")
} else if (2 < 3) {
    print("two is not greater than three")
}
```

```
[1] "two is not greater than three"
```

Finally, you can catch anything that doesn't meet any of your conditions by adding an "else" statement at the end.

```
x <- 20
if (x < 20) {
    print("x is less than 20")
} else if (x > 20) {
    print("x is greater than 20")
} else {
    print("x is equal to 20")
}
```

```
[1] "x is equal to 20"
```

5.8 Libraries

Libraries allow you to access functions other people have created to perform common tasks.

In this example, we will be installing and loading a common package named "dplyr".

You first need to install it using the following command.

```
install.packages("dplyr")
```

Next, you will require the package by using this command.

```
library(dplyr)
```

You are now able to access all of the functions available in the dplyr library!

6 Data Types

There are five basic data types in R:

- **Numeric** - This is the default treatment for numbers. This data type includes integers and doubles.
  - *Double* - A double allows you to store numbers as decimals. This is the default treatment for numbers.
  - *Integer* - An integer is a subset of the numeric data type. This type will only allow whole numbers and is denoted by the letter "L".
- **Complex** - This type is created by using the imaginary variable "i".
- **Character** - This type is used for storing non-numeric text data.
- **Logical** - Sometimes referred to as "boolean", this data type will store either "TRUE" or "FALSE".
- **Raw** - Used less often, this data type will store data as raw bytes.

6.1 Numeric

6.1.1 Double

```
x <- 6.2
class(x)
```

```
[1] "numeric"
```

```
typeof(x)
```

```
[1] "double"
```

```
is.numeric(x)
```

```
[1] TRUE
```

```
is.integer(x)
```

```
[1] FALSE
```

```
is.double(x)
```

```
[1] TRUE
```

6.1.2 Integer

```
x <- 6L
class(x)
```

```
[1] "integer"
```

```
typeof(x)
```

```
[1] "integer"
```

```
is.numeric(x)
```

```
[1] TRUE
```

```
is.integer(x)
```

```
[1] TRUE
```

```
is.double(x)
```

```
[1] FALSE
```

6.2 Complex

```r
x <- 6i
class(x)
```

```
[1] "complex"
```

```r
typeof(x)
```

```
[1] "complex"
```

```r
is.numeric(x)
```

```
[1] FALSE
```

```r
is.integer(x)
```

```
[1] FALSE
```

```r
is.double(x)
```

```
[1] FALSE
```

6.3 Character

```r
x <- "Hello!"
class(x)
```

```
[1] "character"
```

```r
typeof(x)
```

```
[1] "character"
```

6.4 Logical

```r
x <- TRUE
class(x)
```

```
[1] "logical"
```

```r
typeof(x)
```

```
[1] "logical"
```

6.5 Raw

```r
x <- charToRaw("Hello!")
print(x)
```

```
[1] 48 65 6c 6c 6f 21
```

```r
class(x)
```

```
[1] "raw"
```

```r
typeof(x)
```

```
[1] "raw"
```

```r
x <- intToBits(6L)
typeof(x)
```

```
[1] "raw"
```

```r
print(x)
```

```
 [1] 00 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[26] 00 00 00 00 00 00 00
```

```r
class(x)
```

```
[1] "raw"
```

```r
typeof(x)
```

```
[1] "raw"
```

7 Data Structures

There are six basic data structures in R

- **Vectors** - Vectors contain of data which is ordered and of the same type.
- **Lists** - Lists are a collection of objects.
- **Matrices** - A matrix is a two-dimensional array where the data is all of the same type.
- **Factors** - Factors are used to designate levels within categorical data.
- **Data Frames** - A data frame contains two-dimensional data where the data can have different types.
- **Arrays** - Arrays are objects which have more than two dimensions (n-dimensional).

7.1 Vectors

```
x <- c(1, 3, 3, 7)

print(x)
```

```
[1] 1 3 3 7
```

7.2 Lists

```
first_name <- "John"
last_name <- "Smith"
favorite_numbers <- c(1, 3, 3, 7)

person <- list(first_name, last_name, favorite_numbers)

print(person)
```

```
[[1]]
[1] "John"

[[2]]
[1] "Smith"

[[3]]
[1] 1 3 3 7
```

7.3 Matrices

```r
x <- matrix(
        c(1,3,3,7,1,3,3,7,1,3,3,7)
        , nrow = 3
        , ncol = 4
        , byrow = TRUE)

print(x)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    3    7
[2,]    1    3    3    7
[3,]    1    3    3    7
```

7.4 Factors

```r
x <- c("Red", "Blue", "Red", "Yellow", "Yellow")

colors <- factor(x)

print(colors)
```

```
[1] Red    Blue   Red    Yellow Yellow
Levels: Blue Red Yellow
```

7.5 Data Frames

```r
people <- c("John", "Jane")
id <- c(1, 2)
df <- data.frame(id = id, person = people)

print(df)
```

```
  id person
1  1   John
2  2   Jane
```

7.6 Arrays

```r
x <- array(
        c(1,3,3,7,1,3,3,7,1,3,3,7)
        , dim = c(1,4,3))

print(x)
```

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    3    3    7

, , 2

     [,1] [,2] [,3] [,4]
[1,]    1    3    3    7

, , 3

     [,1] [,2] [,3] [,4]
[1,]    1    3    3    7
```

Part III

Part II: Data Acquisition

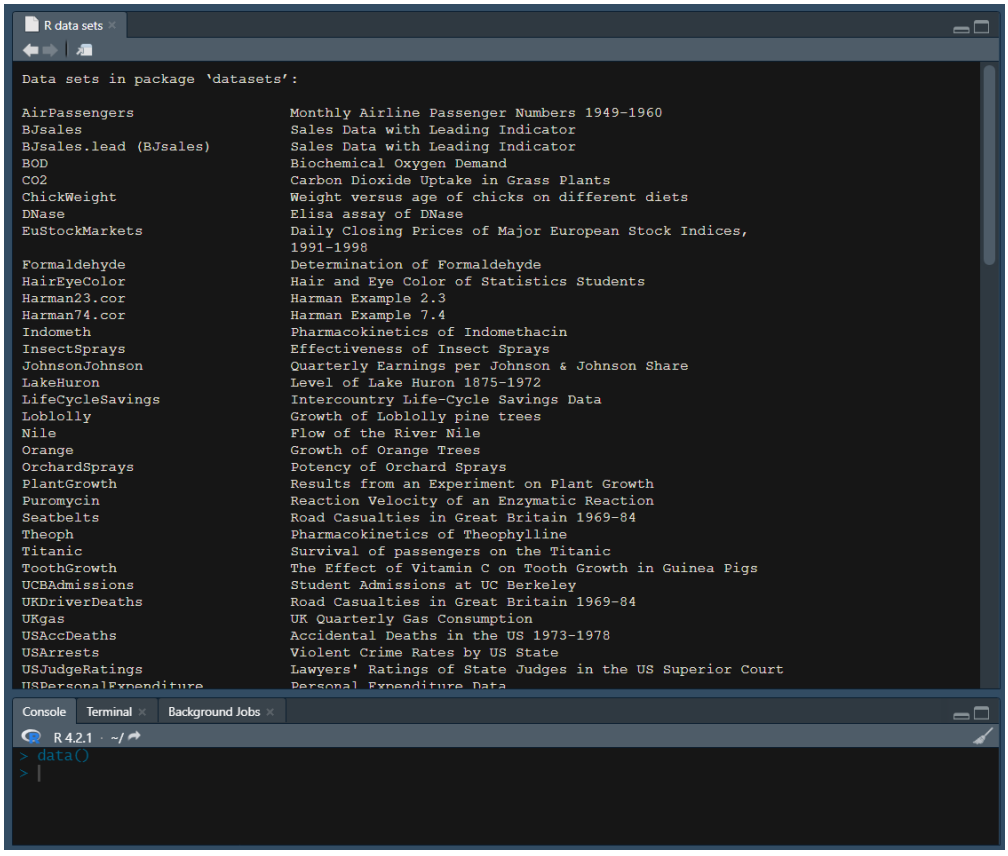
As you might imagine, you must acquire your data before conducting an analysis. This may be done through the methods such as manual creation of datasets, importing pre-constructed data, or leveraging APIs.

1.2 Structure of the Book Part I (Fundamentals) will introduce you to the basics of programming in the context of R. Part II (Data Acquisition) will teach you how to create, import, and access data. Part III (Data Preparation) will show you how to begin preparing your data for analysis. Part IV (Developing Insights) goes through the process of searching for and extracting insights from your data. Part V (Reporting) demonstrates how to wrap your analysis up by developing and automating reports. Each part will be concluded with practical exercises for you to test your skills.

8 Included Datasets

R comes with an assortment of datasets included by default. You can view a complete list of datsets available along with a brief description for each one by typing "data()" into your console.

```
data()
```




```
data("iris")
head(iris)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

9 Import from Spreadsheets

9.1 Import from .csv Files

```
input <- "C:/File Location/example.csv"
df <- read.csv(input)
```

9.2 Import from .xlsx Files

```
library(readxl)
input <- "C:/File Location/example.xlsx"
df <- read_excel(input)
```

10 Working with APIs

10.1 Install Packages

```
install.packages(c('httr', 'jsonlite'))
```

10.2 Require Packages

```
library('httr')
library('jsonlite')
```

10.3 Make Request

Pass a URL into the 'GET' function and store the response in a variable called 'res'.

```
res = GET("https://api.helium.io/v1/stats")
print(res)
```

```
Response [https://api.helium.io/v1/stats]
  Date: 2022-08-04 01:25
  Status: 200
  Content-Type: application/json; charset=utf-8
  Size: 922 B
```

10.4 Parse & Explore Data

Use the 'fromJSON' function from the 'jsonlite' package to parse the response data and then print out the names in the resulting data set.

```
data = fromJSON(rawToChar(res$content))

names(data)
```

```
[1] "data"
```

Go one level deeper into the data set and print out the names again.

```
data = data$data

names(data)
```

```
[1] "token_supply"     "election_times"   "counts"           "challenge_counts" "block_tim
```

Alternatively, you can loop through the names as follows.

```
for (name in names(data)){print(name)}
```

```
[1] "token_supply"
[1] "election_times"
[1] "counts"
[1] "challenge_counts"
[1] "block_times"
```

Get the 'token_supply' field from the data.

```
token_supply = data$token_supply

print(token_supply)
```

```
[1] 124675821
```

10.5 Adding Parameters to Requests

Add 'min_time' and 'max_time' as parameters on a different endpoint and print the resulting 'fee' data.

```
res = GET("https://api.helium.io/v1/dc_burns/sum",
    query = list(min_time = "2020-07-27T00:00:00Z"
                 , max_time = "2021-07-27T00:00:00Z"))

data = fromJSON(rawToChar(res$content))
fee = data$data$fee
print(fee)
```

```
[1] 10112755000
```

10.6 Adding Headers to Requests

Execute the same query as above except this time specify headers. This will likely be necessary when working with an API which requires an API Key.

```
res = GET("https://api.helium.io/v1/dc_burns/sum",
    query = list(min_time = "2020-07-27T00:00:00Z"
                 , max_time = "2021-07-27T00:00:00Z"),
    add_headers(`Accept`='application/json', `Connection`='keep-live'))

data = fromJSON(rawToChar(res$content))
fee = data$data$fee
print(fee)
```

```
[1] 10112755000
```

Part IV

Part III: Data Preparation

Most data will not be received in the precise format you need to begin your analysis. The process of data preparation is where you will structure and add features to your data.

1.2 Structure of the Book Part I (Fundamentals) will introduce you to the basics of programming in the context of R. Part II (Data Acquisition) will teach you how to create, import, and access data. Part III (Data Preparation) will show you how to begin preparing your data for analysis. Part IV (Developing Insights) goes through the process of searching for and extracting insights from your data. Part V (Reporting) demonstrates how to wrap your analysis up by developing and automating reports. Each part will be concluded with practical exercises for you to test your skills.

11 Data Cleaning

11.1 Renaming Variables

11.2 Text to Columns

11.3 Replace Values

11.4 Drop Columns

11.5 Drop Rows

12 Handling Missing Data

12.1 Replacing Nulls/NAs

12.2 Mean Imputation

12.3 Multiple Imputation?

13 Outliers

13.1 Finding Outliers

13.2 Removing Outliers

14 Organizing Data

14.1 Sorting

14.2 Filtering

14.3 Grouping

Part V

Part IV: Developing Insights

Once your data is prepared, you can now begin to make sense of your data and develop insights about it's meaning.

1.2 Structure of the Book Part I (Fundamentals) will introduce you to the basics of programming in the context of R. Part II (Data Acquisition) will teach you how to create, import, and access data. Part III (Data Preparation) will show you how to begin preparing your data for analysis. Part IV (Developing Insights) goes through the process of searching for and extracting insights from your data. Part V (Reporting) demonstrates how to wrap your analysis up by developing and automating reports. Each part will be concluded with practical exercises for you to test your skills.

15 Summary Statistics

16 Regression

17 Plotting

17.1 Base R

17.1.1 Different types of plots?

17.2 ggplot2

17.2.1 Different types of plots?

- Resource: https://ggplot2.tidyverse.org/

Part VI

Part V: Reporting

Finally, it's important to report on your data in such a way that the information is able to be digested by the people who need to see it when they need to see it.

1.2 Structure of the Book Part I (Fundamentals) will introduce you to the basics of programming in the context of R. Part II (Data Acquisition) will teach you how to create, import, and access data. Part III (Data Preparation) will show you how to begin preparing your data for analysis. Part IV (Developing Insights) goes through the process of searching for and extracting insights from your data. Part V (Reporting) demonstrates how to wrap your analysis up by developing and automating reports. Each part will be concluded with practical exercises for you to test your skills.

18 Spreadsheets

18.1 Export

-csvs and xlsx ## Formatting - colors - tabs - font - etc

19 R Markdown

R Markdown allows you to create documents in a programmatic fashion that lends itself towards reproducibility.

19.1 Including Plots

- Formats: https://rmarkdown.rstudio.com/formats.html
- Resource: https://rmarkdown.rstudio.com/

20 R Notebook

Some more technical audiences may require reporting which includes your methodology. This is where R Notebooks come in.

- Subset of R Markdown
- Resource: https://rmarkdown.rstudio.com/lesson-10.html

21 R Shiny

R Shiny is a tool used to develop web applications and is commonly deployed in the use of creating dashboards, hosting static reports, and custom tooling.

- Shiny apps account
- Quick Start
- Resource: https://shiny.rstudio.com/

References

Eremenko, Kirill. 2020. "Hadley Wickham Talks Integration and Future of r and Python [Audio Podcast]." SuperDataScience. https://www.superdatascience.com/podcast/hadley-wickham-talks-integration-and-future-of-python-and-r.

Hermans, Felienne. 2021. "Hadley Wickham on r and Tidyverse [Audio Podcast]." Software Engineering Radio. https://www.se-radio.net/2021/03/episode-450-hadley-wickham-on-r-and-tidyverse/.

Hofmann, J. R. 1996. *Enlightenment and Electrodynamics*. Cambridge University Press.

Ihaka, Ross. 1998. "R : Past and Future History." https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf.

Paulson, Josh. 2022. *Navigating Code in the RStudio IDE*. https://support.rstudio.com/hc/en-us/articles/200710523-Navigating-Code-in-the-RStudio-IDE.