RMS textbook

Pilot, Lucas, and Murray

2025-10-28

Table of contents

Preface

This is a booked intended to be use for Dr. Pilot's PSY 303 course at the University of Southern Indiana, home of the screaming eagles.

This book was a collaboration between Dr. Pilot, Liam Murray, and Jacob Lucas.

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

2 Fundamentals of R

2.1 2.1 Introduction

This chapter will go over the fundamental tools you will need in order to work in R and to create projects and to experiment with the building blocks of Rstudio. If you have not already gone to chapter one that teaches you how to set up Rstudio do that now.

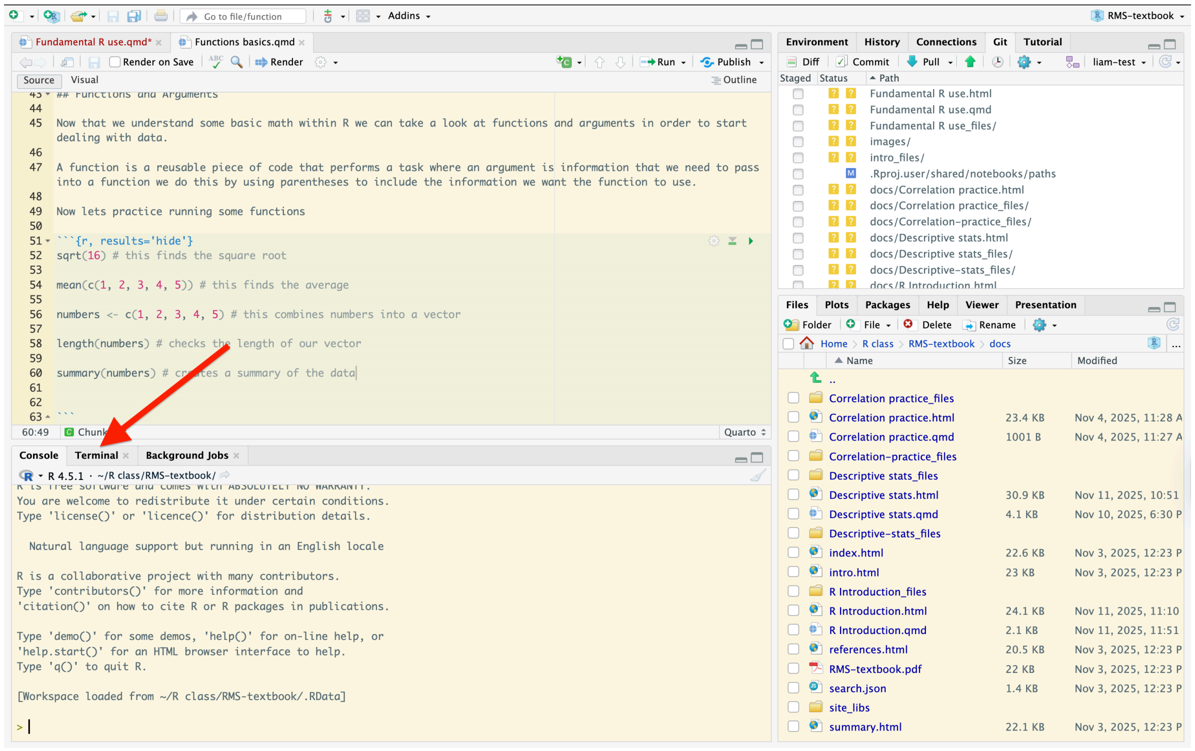
- How to use packages and what you will need to do in order to update them.
- How to create and manipulate data.
- How to use functions, objects, and pipes.
- Common errors and different ways to deal with them.
- And the main types of data that you will be running into in Rstudio.

2.2 2.2 Rstudio Overview

Once you open Rstudio you might notice that there are four different panes on your screen that each look different from the other. Although it might look overwhelming these panes are all important when operating in Rstudio and will all be used

2.2.1 2.2.1 Console Frame

The console frame is the **bottom left frame on your screen**. This is where you will be viewing some of your code. You can also see outputs from your code directly in this frame and is one location of where you will be trouble shooting as well. Think of this frame as a chat box in R in order to see behind the scenes of what you will be running. You should make note that the console is like a scrap piece of paper or and etch an sketch its a great place to do some troubleshooting or viewing data through the `glimpse()` function but **any work you do here will not be saved.** The console is also next to the terminal which is the tab right next to it. **They both can run commands but are different in nature**.The console is intended to run commands that work mainly inside Rstudio itself. The terminal however runs systems commands like something you do on your computer.



~~~
## Functions and Arguments

Now that we understand some basic math within R we can take a look at functions and arguments in order to start
dealing with data.

A function is a reusable piece of code that performs a task where an argument is information that we need to pass
into a function we do this by using parentheses to include the information we want the function to use.

Now lets practice running some functions

```{r, results='hide'}
sqrt(16) # this finds the square root

mean(c(1, 2, 3, 4, 5)) # this finds the average

numbers <- c(1, 2, 3, 4, 5) # this combines numbers into a vector

length(numbers) # checks the length of our vector

summary(numbers) # creates a summary of the data


```
~~~

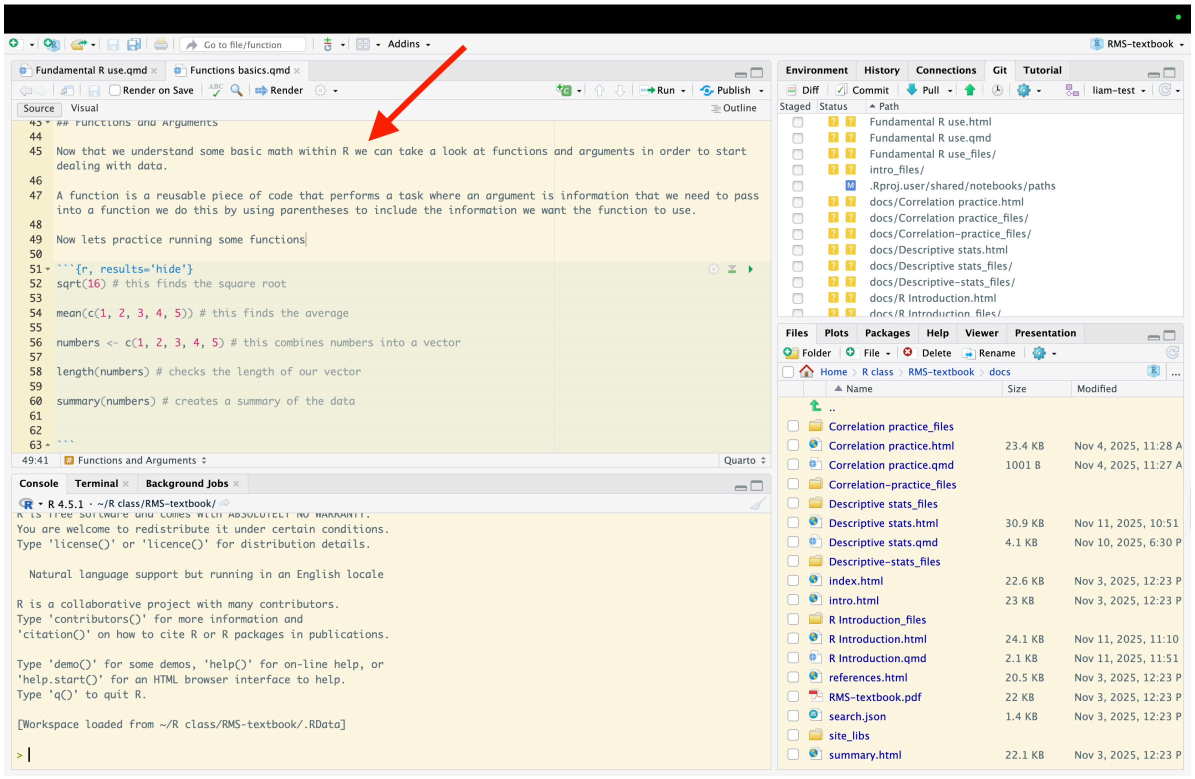
```
R 4.5.1 · ~/R class/RMS-textbook/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  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.

[Workspace loaded from ~/R class/RMS-textbook/.RData]

> 
```

Staged	Status	Path
☐	? ?	Fundamental R use.html
☐	? ?	Fundamental R use.qmd
☐	? ?	Fundamental R use_files/
☐	? ?	images/
☐	? ?	intro_files/
☐	M	.Rproj.user/shared/notebooks/paths
☐	? ?	docs/Correlation practice.html
☐	? ?	docs/Correlation practice_files/
☐	? ?	docs/Correlation-practice_files/
☐	? ?	docs/Descriptive stats.html
☐	? ?	docs/Descriptive stats_files/
☐	? ?	docs/Descriptive-stats_files/
☐	? ?	docs/R Introduction.html

Name	Size	Modified
..		
Correlation practice_files		
Correlation practice.html	23.4 KB	Nov 4, 2025, 11:28 A
Correlation practice.qmd	1001 B	Nov 4, 2025, 11:27 A
Correlation-practice_files		
Descriptive stats_files		
Descriptive stats.html	30.9 KB	Nov 11, 2025, 10:51
Descriptive stats.qmd	4.1 KB	Nov 10, 2025, 6:30 P
Descriptive-stats_files		
index.html	22.6 KB	Nov 3, 2025, 12:23 P
intro.html	23 KB	Nov 3, 2025, 12:23 P
R Introduction_files		
R Introduction.html	24.1 KB	Nov 11, 2025, 11:10
R Introduction.qmd	2.1 KB	Nov 11, 2025, 11:51
references.html	20.5 KB	Nov 3, 2025, 12:23 P
RMS-textbook.pdf	22 KB	Nov 3, 2025, 12:23 P
search.json	1.4 KB	Nov 3, 2025, 12:23 P
site_libs		
summary.html	22.1 KB	Nov 3, 2025, 12:23 P

2.2.2 2.2.2 Source Frame

The source frame or editor frame is **the top left frame** on your screen. This frame is the primary location of where you will be doing your work and typing all of your code. When you create a new code chunk you will do that in the source frame. You should be able to see all the lines and numbers of each code line. You can also save and open new documents and files at the farthest top left corner of this frame. Opening your Rmarkdown files like you created in **chapter 1** is a great way to practice using the source frame and where you will do this.



~~~
## Functions and Arguments

Now that we understand some basic math within R we can take a look at functions and arguments in order to start
dealing with data.

A function is a reusable piece of code that performs a task where an argument is information that we need to pass
into a function we do this by using parentheses to include the information we want the function to use.

Now lets practice running some functions

```{r, results='hide'}
sqrt(16) # this finds the square root

mean(c(1, 2, 3, 4, 5)) # this finds the average

numbers <- c(1, 2, 3, 4, 5) # this combines numbers into a vector

length(numbers) # checks the length of our vector

summary(numbers) # creates a summary of the data



```
~~~

```
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.

[Workspace loaded from ~/R class/RMS-textbook/.RData]

> 
```

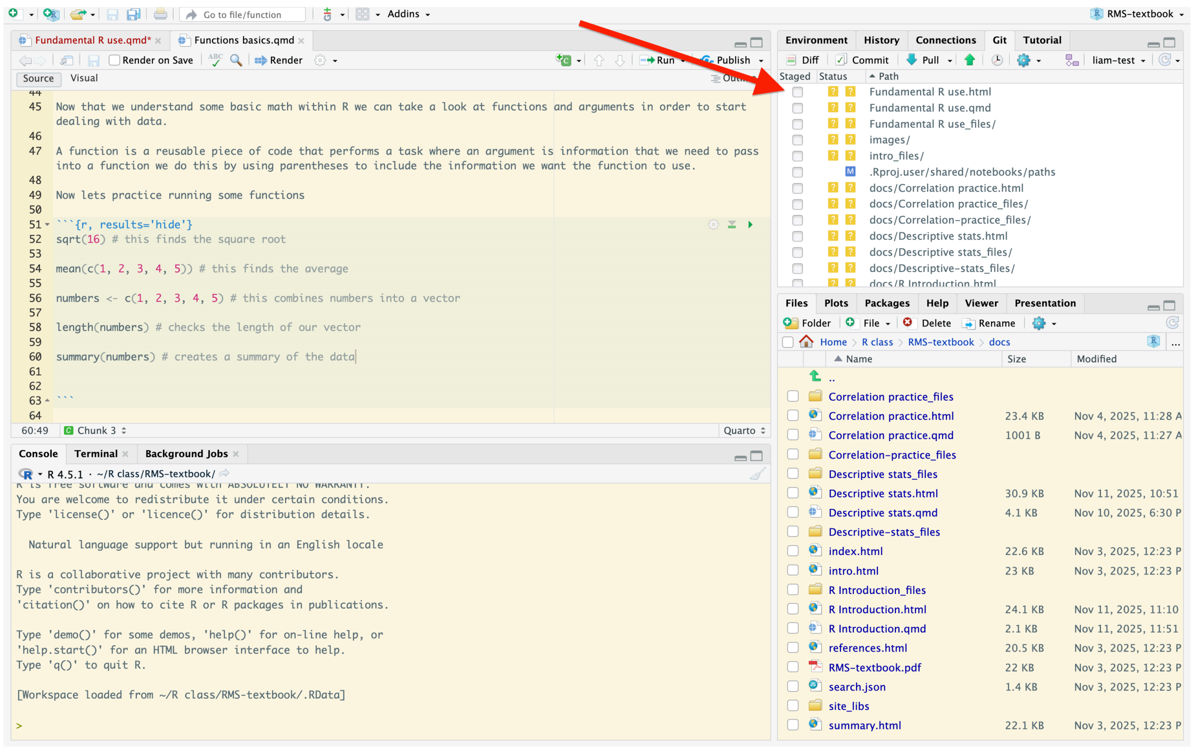
Staged	Status	Path
	? ?	Fundamental R use.html
	? ?	Fundamental R use.qmd
	? ?	Fundamental R use_files/
	? ?	intro_files/
	M	.Rproj.user/shared/notebooks/paths
	? ?	docs/Correlation practice.html
	? ?	docs/Correlation practice_files/
	? ?	docs/Correlation-practice_files/
	? ?	docs/Descriptive stats.html
	? ?	docs/Descriptive stats_files/
	? ?	docs/Descriptive-stats_files/
	? ?	docs/R Introduction.html
	? ?	docs/R Introduction_files/

Home › R class › RMS-textbook › docs

Name	Size	Modified
..		
Correlation practice_files		
Correlation practice.html	23.4 KB	Nov 4, 2025, 11:28 A
Correlation practice.qmd	1001 B	Nov 4, 2025, 11:27 A
Correlation-practice_files		
Descriptive stats_files		
Descriptive stats.html	30.9 KB	Nov 11, 2025, 10:51
Descriptive stats.qmd	4.1 KB	Nov 10, 2025, 6:30 P
Descriptive-stats_files		
index.html	22.6 KB	Nov 3, 2025, 12:23 P
intro.html	23 KB	Nov 3, 2025, 12:23 P
R Introduction_files		
R Introduction.html	24.1 KB	Nov 11, 2025, 11:10
R Introduction.qmd	2.1 KB	Nov 11, 2025, 11:51
references.html	20.5 KB	Nov 3, 2025, 12:23 P
RMS-textbook.pdf	22 KB	Nov 3, 2025, 12:23 P
search.json	1.4 KB	Nov 3, 2025, 12:23 P
site_libs		
summary.html	22.1 KB	Nov 3, 2025, 12:23 P

2.2.3 2.2.3 Environment/History

The environment or history frame can be seen in **the top right frame on your screen.** This is where you can view current objects that you have created which we will discuss further in the chapter as well as data sets and where you can track your steps by viewing the command history that you have run.



~~~
Now that we understand some basic math within R we can take a look at functions and arguments in order to start
dealing with data.

A function is a reusable piece of code that performs a task where an argument is information that we need to pass
into a function we do this by using parentheses to include the information we want the function to use.

Now lets practice running some functions

```{r, results='hide'}
sqrt(16) # this finds the square root

mean(c(1, 2, 3, 4, 5)) # this finds the average

numbers <- c(1, 2, 3, 4, 5) # this combines numbers into a vector

length(numbers) # checks the length of our vector

summary(numbers) # creates a summary of the data


```
~~~

```
R 4.5.1 · ~/R class/RMS-textbook/
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.

[Workspace loaded from ~/R class/RMS-textbook/.RData]

>
```

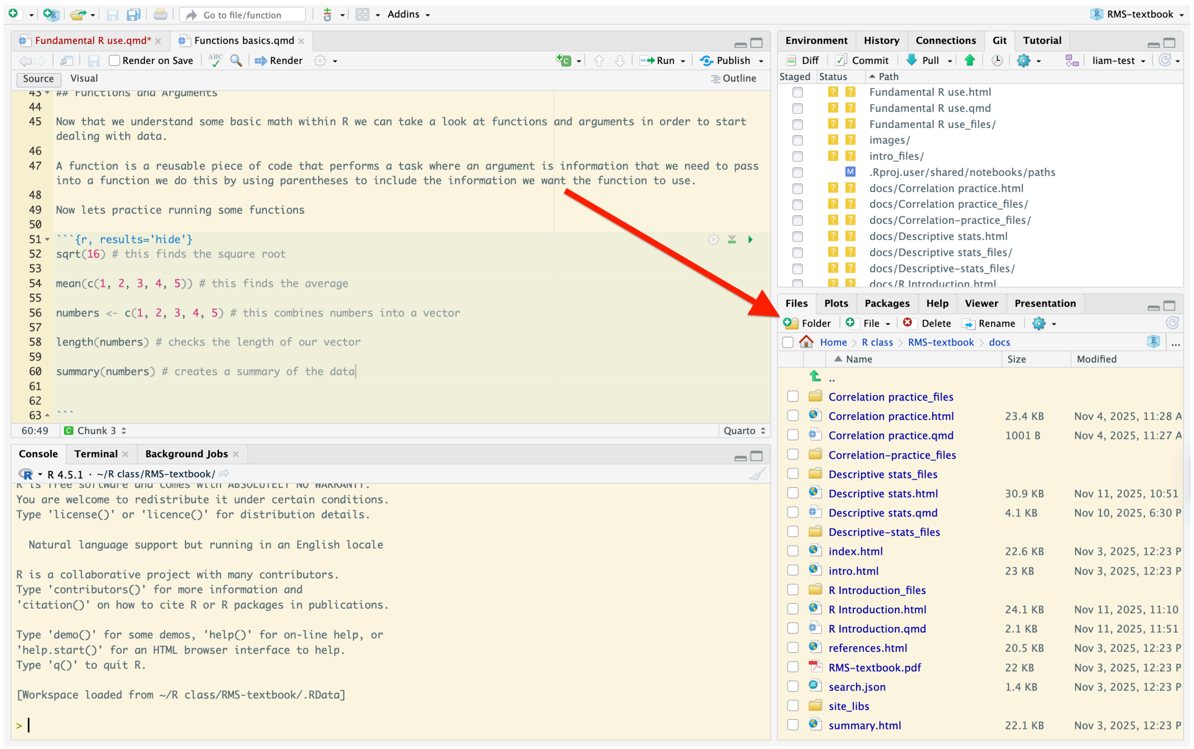
Staged	Status	Path
☐	? ?	Fundamental R use.html
☐	? ?	Fundamental R use.qmd
☐	? ?	Fundamental R use_files/
☐	? ?	images/
☐	? ?	intro_files/
☐	M	.Rproj.user/shared/notebooks/paths
☐	? ?	docs/Correlation practice.html
☐	? ?	docs/Correlation practice_files/
☐	? ?	docs/Correlation-practice_files/
☐	? ?	docs/Descriptive stats.html
☐	? ?	docs/Descriptive stats_files/
☐	? ?	docs/Descriptive-stats_files/
☐	? ?	docs/R Introduction.html

Home > R class > RMS-textbook > docs

Name	Size	Modified
..		
Correlation practice_files		
Correlation practice.html	23.4 KB	Nov 4, 2025, 11:28 A
Correlation practice.qmd	1001 B	Nov 4, 2025, 11:27 A
Correlation-practice_files		
Descriptive stats_files		
Descriptive stats.html	30.9 KB	Nov 11, 2025, 10:51
Descriptive stats.qmd	4.1 KB	Nov 10, 2025, 6:30 P
Descriptive-stats_files		
index.html	22.6 KB	Nov 3, 2025, 12:23 P
intro.html	23 KB	Nov 3, 2025, 12:23 P
R Introduction_files		
R Introduction.html	24.1 KB	Nov 11, 2025, 11:10
R Introduction.qmd	2.1 KB	Nov 11, 2025, 11:51
references.html	20.5 KB	Nov 3, 2025, 12:23 P
RMS-textbook.pdf	22 KB	Nov 3, 2025, 12:23 P
search.json	1.4 KB	Nov 3, 2025, 12:23 P
site_libs		
summary.html	22.1 KB	Nov 3, 2025, 12:23 P

2.2.4 2.2.4 Files/Packages

The last frame on **the bottom right is your files and packages** frame. This frame is more of a window into your own computer itself in the sense that you are able to view the packages and files that you have on your current computer. You can also use it to check any plots you might have and to read any documents on your computer as well. You only will need to download a package one time but whenever you open Rstudio you will need to load it up every time.



~~~
## Functions and Arguments

Now that we understand some basic math within R we can take a look at functions and arguments in order to start
dealing with data.

A function is a reusable piece of code that performs a task where an argument is information that we need to pass
into a function we do this by using parentheses to include the information we want the function to use.

Now lets practice running some functions

```{r, results='hide'}
sqrt(16) # this finds the square root

mean(c(1, 2, 3, 4, 5)) # this finds the average

numbers <- c(1, 2, 3, 4, 5) # this combines numbers into a vector

length(numbers) # checks the length of our vector

summary(numbers) # creates a summary of the data


```
~~~

```
R 4.5.1 · ~/R class/RMS-textbook/
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.

[Workspace loaded from ~/R class/RMS-textbook/.RData]

>
```

```
Staged  Status  Path
        ? ?     Fundamental R use.html
        ? ?     Fundamental R use.qmd
        ? ?     Fundamental R use_files/
        ? ?     images/
        ? ?     intro_files/
          M     .Rproj.user/shared/notebooks/paths
        ? ?     docs/Correlation practice.html
        ? ?     docs/Correlation practice_files/
        ? ?     docs/Correlation-practice_files/
        ? ?     docs/Descriptive stats.html
        ? ?     docs/Descriptive stats_files/
        ? ?     docs/Descriptive-stats_files/
        ? ?     docs/R Introduction.html
```

```
Home > R class > RMS-textbook > docs
Name                          Size      Modified
..
Correlation practice_files
Correlation practice.html     23.4 KB   Nov 4, 2025, 11:28 A
Correlation practice.qmd      1001 B    Nov 4, 2025, 11:27 A
Correlation-practice_files
Descriptive stats_files
Descriptive stats.html        30.9 KB   Nov 11, 2025, 10:51
Descriptive stats.qmd         4.1 KB    Nov 10, 2025, 6:30 P
Descriptive-stats_files
index.html                    22.6 KB   Nov 3, 2025, 12:23 P
intro.html                    23 KB     Nov 3, 2025, 12:23 P
R Introduction_files
R Introduction.html           24.1 KB   Nov 11, 2025, 11:10
R Introduction.qmd            2.1 KB    Nov 11, 2025, 11:51
references.html               20.5 KB   Nov 3, 2025, 12:23 P
RMS-textbook.pdf              22 KB     Nov 3, 2025, 12:23 P
search.json                   1.4 KB    Nov 3, 2025, 12:23 P
site_libs
summary.html                  22.1 KB   Nov 3, 2025, 12:23 P
```

Suggestion: If you wish to change the layout of your frames to customize it simply click tools at the top of your computer then global options then to pane layout to customize the layout to whatever is the best for you.

2.3 2.3 Packages and the tidyverse

Now that you are familiar with where you can view and access your packages we will now take a closer look as to what they are and different tips fro them as well as introducing you to a very useful package that will make writing code much easier and it is called the tidyverse.

2.3.1 2.3.1 What is a package?

You might be wondering how we are going to be using something that you typically get in the mail on your computer. In R a package is where R gets its main power from. We have packages because just like in real life they contain something. In R packages can contain a multitude of functions, data, and documents.

2.3.2 2.3.2 Installing and finding packages

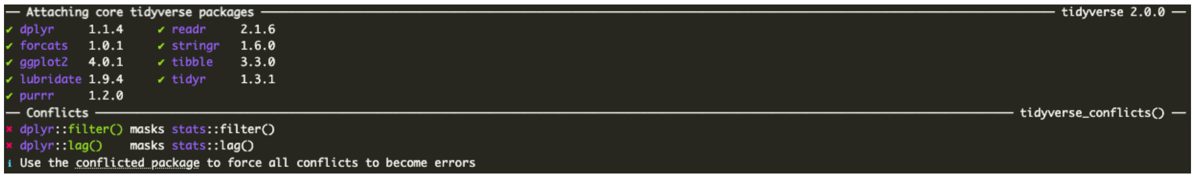
Whenever you want to install a package you will have to run a command. When you are working in a document you will need to load your packages each time or else **your code will not run because the package has not been reloaded.** You can install packages by going to your console frame in the bottom left. Once you are there you will type the following code. Do not be afraid if you see a warning in yellow that is perfectly normal.

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

Then to load it into your active session the first thing you always want to do is to load it into your session. You will want to do this by running the following in your source frame or the top left. We do this by using the the **library** function which calls up any package that you have installed it will not work if you do not have the package installed.

```
library(tidyverse)
library(dplyr)
```

Once you have successfully loaded the tidyverse into your current session you should get the following result in your console:



```
── Attaching core tidyverse packages ──────────────────────────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.6
✔ forcats   1.0.1     ✔ stringr   1.6.0
✔ ggplot2   4.0.1     ✔ tibble    3.3.0
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.2.0
── Conflicts ──────────────────────────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package to force all conflicts to become errors
```

THIS IS OK IT MEANS YOU HAVE SUCCESFULLY LOADED YOUR PACKAGE

Now sometimes you will need to update your packages because if you don't it can cause them to not load and your code will not run. If this happens all you will have to do is run this code.

```
update.packages()
```

To see what packages you have installed you can either go to the bottom right pane and find the packages tab to find a list of all packages and which are installed. Or you could also run this code.

```r
installed.packages()
```

2.3.3 2.3.3 The Tidyverse

Now that you have successfully installed the **tidyverse** we can examine it and what it does. When thinking about the tidyverse the best way to explain it is think of it like a universe. A universe is a space that contains different things like planets which contain different things like people. Well just like a universe the tidyverse contains a large amount of packages inside of it that are meant to make using R easier and work more efficiently. All the packages inside the tidyverse all share a common philosophy and syntax. Think of it like the world having america and how we all share a common philosophy and syntax. The packages in the tidyverse work the same way. Because all of these functions use the same grammar it makes it easier for us to read R code. Take a look at this example

```r
library(tidyverse)

# Example of data transformation
starwars %>%
  select(name, height, mass, species) %>%
  filter(species == "human") %>%
  arrange(desc(mass))
```

2.4 2.4 Functions

Now that we are able to successfully load up packages we can start looking at some functions. A **function** is a command that begins an action this is a basic example of a function

```r
function_name(argument1 = value1, argument2 = value2)
```

2.4.1 2.4.1 Applied Functions

The function is made up of two arguments that we give values to.

Next we can look at a basic function called the **combine** function or it can be viewed as this `c()`. It takes every value we have in the function and applies it to all of them. This can be used for mathematical purposes when using Rstudio like a calculator. Lets take a look at an example

```
mean(c(1, 2, 3, 4, 5))
```

```
[1] 3
```

As you can see we get the output after and we can see the answer is 3. We ran this function and all the values were applied to each other while we also utilized the **mean** function combining two functions.

2.4.2 2.4.2 Creating Functions

Creating your own function is a very useful skill because it allows you to easily apply values to data without you having to type it out over and over again. In order to create a function you must name it then assign it a value by using the `<-` symbol. This assigns whatever you write to that name which then should appear in the top right frame or your environment frame.

```
plus_two <- function(x) {
  x + 2
}

plus_two(5)
```

```
[1] 7
```

As you see we created the function `plus_two` Which will take whatever is in our function and add 2 to it so when we use the function and put 5 in there it takes 5 and already adds the two to it.

2.4.3 2.4.3 Assignment operator

Whenever we want to assign a value to something we need to use the `<-` symbol. This will then take whatever we assign to the value we create. As you can see in the example above the function we created was assigned to the value `plus_two` because we used the assignment operator to give it that value.

2.4.4 2.4.4 Arguments

When we are using functions we must also know that we are dealing with arguments as well. An argument is the information that we put into a function so the function knows what to do with it. So think of the function like a machine and the arguments are the ingredients that we give it.

```r
mean(x = c(1, 2, 3, 4, 5))
```

As you can see here the mean is the function we are using and the argument is everything inside of the parenthesis.

2.5 2.5 Creating Data

There are several different ways that we can create data which is important when we want different functions

2.5.1 2.5.1 Vectors

A **vector** is a simple data structure that holds elements of the same kind. So if we want to combine a bunch of names we can use a vector to do so but they need to all be the same type of data. Which we will go over the different types of data later in the chapter. Here are some simple basic examples of a vector

```r
numbers <- c(1, 2, 3, 4, 5, 6, 7)
names <- c("Adam", "Steven", "James")
```

Now once we have these saved we can use them again if we want since we have combined them all into one vector.

```r
numbers
```

```
[1] 1 2 3 4 5 6 7
```

2.5.2 2.5.2 Data Frames

A **data frame** is a different way of creating data. When you use a data frame think of it like a window frame. In a data frame the data is arranged into a rectangular shape and uses rows and columns. The columns in a data frame are called **variables** and the rows in our data frame are called **observations**. Now we can use data frames to combine two different types of data into one output.

```r
data.frame(
  name = c("James", "Henry"),
  age = c(21,67),
  sport = c("Baseball", "Soccer")
)
```

```
   name age    sport
1 James  21 Baseball
2 Henry  67   Soccer
```

You can see our output of how the data is arranged into the order that we make it to where the same location is applied so that they go in order. You can also see at the bottom of the output where it displays the number of rows that we have.

2.5.3 2.5.3 Tribbles

A tribble is short for transposed tibble and it is a different way to create small data frames.

```r
library(tibble)

people <- tribble(
  ~name,    ~age, ~city,
  "James",  21,    "Evansville",
  "Hunter", 38,    "Toledo"
)
people
```

```
# A tibble: 2 x 3
  name     age city
  <chr>  <dbl> <chr>
1 James     21 Evansville
2 Hunter    38 Toledo
```

Now you can see our tribble fully completed with both rows and columns where you can see in the bottom of the output that the tribble shows the rows. It also tells us the type of data that we are seeing in the output which you will learn more about later in the chapter.

2.5.4 2.5.4 Glimpse

When you want to examine the characteristics of your data you can use glimpse to get more information about it. It provides the full data for your set and gives all the details you could need.

```
glimpse(people)
```

```
Rows: 2
Columns: 3
$ name <chr> "James", "Hunter"
$ age  <dbl> 21, 38
$ city <chr> "Evansville", "Toledo"
```

Look at the output above you can see next to name the `<chr>` that means the column contains data that is characters. The one below that is the `<dbl>` so you can see that is double data. This shows us that vectors simple a column of names and values that are the same type.

2.6 2.6 Objects

In R the only thing we use are objects and a object can be data, a function or even models that we need. In order to assign a object we use the `<-`.

```
x <- 67
y <- "Good Morning"
```

2.6.1 2.6.1 Listing objects

If you ever want to see what your current objects are you can always look in your environment frame to see it. You can also use the list function `ls()` to see this as well.

```
ls()
```

2.7 2.7 The pipe

Now we will be working with the pipe or `|>`. The pipe is a very important tool because it works kind of like a river with bridges. Usually in R when we run multiple functions we would have to think of the output like a boat going through a river. Normally we would have to manually open each bridge in order for it to pass through to the next part of the river. The pipe makes it easier for us by doing that automatically. So whenever we are running multiple functions in a code chunk we use the pipe to channel the output right into the next function without having to do the work ourselves. In the tidyverse the pipe is represented as `%>%` which can be easily created with the shortcut ***command + shift + m.*** Now lets look at an example of how the pipe works.

```
mtcars %>%
  group_by(cyl) %>%
  summarize(mean_mpg = mean(mpg))
```

You can see in the output now that in our pipe we took the car data set then funneled in the function of grouping cars by their cylinders then taking that output and funneling it into finding the average miles per gallon which creates our final output.

Now try running the same code without the use of the pipe and see if it still works.

2.8 2.8 Types of data

There are different types of data in R and it is important to be able to distinguish the differences because we work with different types in different ways so being able to tell them apart will help us better interpret data.

2.8.1 2.8.1 Numeric data

Numeric data is the type of data that we see that represents whole numbers. These are pretty common and are just regular numbers that can be represented differently. When looking at an output in R we can see if the data we are looking at is numeric by looking for how it is abbreviated which is by `num`. Whenever we see a column whit this title we know the data in that column is numeric. You can use the `typeof()` function in R to see exactly what type of data you are dealing with.

```r
typeof(42)
```

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

```r
typeof(FALSE)
```

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

```r
typeof("jack")
```

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

2.8.2 2.8.2 Character data

Character data is words that we use in R that are not functions but represent something in the data we are dealing with. These could be examples like names in a data set that represent something. It is important to note though we can tell if the data is a character because it must always be in `""`. This shows that it is just text we are dealing with and not some function. The abbreviation for character data in R is `chr`. So whenever you see a column with `chr` you know the data in it is character data.

```r
    my_vector <- c("apple", "banana", "cherry")
    str(my_vector)
```

2.8.3 2.8.3 Logical data

Logical data is a type of Boolean data in the sense that it can only represent one of two values. This means that when we look at logical data we are looking to see if something is either true or false. Kind of like in real life when we use logic to see if we believe something or not. So in R logical data will be represented as either `TRUE` or `FALSE`. It is important that they are all capitalized so they cannot be confused for something else. The abbreviations in R are simple because it will either be `T` or `F`.

```r
# Assigning logical values using full names
bool1 <- TRUE
bool2 <- FALSE

# Assigning logical values using abbreviations
bool3 <- T
bool4 <- F
```

2.8.4 2.8.4 Factor data

In R we refer to factor data as the type of categories that variables are stored as a factor. It works with variables that have a fixed and already known set of possible values. We use factor data differently depending on the data we are trying to use. We use `factor()` to create a new factor from a vector. We use `as.factor()` to move an object like a character list into a vector. `is.factor()` is when we want to check if an object is already a factor.

```r
class(factor(c("Low"), "High"))
```

2.8.5 2.8.5 Double data

Double data is how we refer to data that are numbers but not whole numbers. Not to be confused with numeric data double data deals with decimals that are numbers. This can be represented as `dbl`. So this tells us that whenever we see a column that has that list it means that we are dealing with numbers but they are decimals and not whole numbers

2.9 2.9 Tips and trouble shooting

When working with R there is always going to be something that will end up needing fixing or something will go wrong and you have no idea what to do. That is OK because there are a few different ways to figure things out when you need help.

2.9.1 2.9.1 ? Tool

The question mark tool is a great too that can help explain anything you need. Lets say for example you don't know what a mean is. You can type in `?mean` and the help tab in your bottom right frame will open with whatever you need. It provides arguments and explanations as it is a great tool to help you figure things out.

```
?mean
```

2.9.2 2.9.2 Help

The help function is another way to get info on objects you might be struggling with. It works the same as the question and can give you more information on it.

```
help("mean")
```

2.9.3 2.9.3 Traceback

Whenever we are working in R sometimes we might get an error. This can be confusing because R doesn't always tell you where you made this error it usually tells you what is wrong and when you are writing lots of code it can be difficult to find where you went wrong. Well you can use the `traceback()` function to find exactly where your code stopped working.

2.10 2.10 Practice

Now that you learned the basic fundamentals of R and operating inside of it here are some practice problems to make yourself more comfortable with working in R

1. Create a `tribble()` with four of your friends names, ages, and cities they are from
2. Use `glimpse()` to inspect the data
3. Write a function that doubles any number
4. Use the pipe to select certain columns
5. Find and list each columns type of data

3 What are Behavioral Sciences

3.1 What is Psychology?

Psychology is the study of behavior and mental processes.

Psychology is rooted in philosophical thought and exploration.

Wilhelm Wundt created the first psychology lab.

Your Lineage:

Wundt -> Titchener -> Boring -> Tulving -> Habib -> Me->You

Behavioral research is involved in a multitude of different disciplines; like Social work, Criminology, and Communication.

3.2 Goals of Behavioral Research

Describe

Patterns of behavior, thought, and emotion.

Predict behavior

Focus on developing equations that predict behavior.

Explain behavior

Develop theoretical explanations for patterns of behavior.

3.3 Two Schools of Research

Basic Research

Research conducted without regard for whether the knowledge is immediately applicable

Ex. Does drinking coffee influence long-term memory?

Applied Research

Research conducted to find solutions for problems rather than to enhance general knowledge

Ex. Does giving paid maternal/paternal leave increase employee happiness at USI?

3.4 Scientific Approach

Systematic Empiricism

Observing behavior with clear guidelines for the purpose of drawing conclusions.

Public Verification

Allows others to replicate and discuss your findings.

Solvable Problems

Research questions must be solvable with the current technology.

Examples of currently unsolvable problems: Whether Freud's "unconscious" exists, angels, souls, quantum theory?, vampires, fairies

3.5 Purpose of Behavioral Research

Detect

Discover and document new phenomena.

Explain

Develop and evaluate theories that explain phenomena.

3.6 How to Explain

Theory

Describes relationships between ideas.

Ex. Theory of Multiple Intelligences- Gardner suggests that there are 8-10 distinct modalities of intelligence instead of one general factor.

Example: Theory of Evolution

Model

A representation of a process.

Example: Assortative Mating Model-People tend to marry a partner who is has similar interests, lives close, makes a similar amount of money.

3.7 How to create a hypothesis

Hypothesis

An idea suggested as a way to explain a phenomena.

Post-hoc

Explanations made after the fact.

A priori

Predictions made before experimentation.

All hypotheses must be falsifiable, able to be unsupported, or shown to be false.

3.8 What is a variable?

A variable is something that you measure.

Two ways to define variables:

Conceptual definition

A dictionary definition.

Ex: Drunk = affected by alcohol to the extent of losing control of one's faculties or behavior.

Operational definition

Definition that specifies precisely how a concept is measured, think about behaviors you can see.

Ex. Drunk = Blood Alcohol Content over .08.

3.9 Proof, Disproof, and Progress

Scientists do not prove anything, they find information that supports hypotheses.

Scientists may disprove.

Example: How would one disprove the statement "Unicorns do not exist."? They would find a unicorn.

Scientific progress depends on replication and accumulated evidence.

3.10 Research Strategies

Descriptive

Describes behavior, thoughts, or feelings.

Correlational

Investigates relationship between two or more variables.

Quasi-experimental

Examines naturally occurring variables.

Experimental

Determines whether certain variables cause changes.

Non-human animals can be studied in controlled conditions, for extended periods of time, and can be utilized in many types of research inappropriate for human beings.

4 T-tests

4.1 14.1 Introduction

This chapter will be teaching you about a T-test and the use we have for them in psychology and also the real life applications there are when we use them. T-tests are a very useful tool when it comes to inferential statistics and they allow us to compare means and to also determine if group differences that we observe are statistically significant. In this chapter you will learn how to:

- Run one sample and independent sample t-tests and paired t-test.
- perform t-tests using R and interpret them
- find different assumptions like normality and homogeneity of variance
- create apa style results using the `report` and `apaTables` packages

4.2 14.2 Pre-requisites

This chapter will be using different packages in R to make calculating t-tests easier. One function we will be using is the `t_test` function from the `rstatix` package which utilizes the following formula (x = DV ~ 1, mu = number). The 1 after the ~ is used for one sample t tests because we aren't comparing two samples. You will also need to load up the following packages.

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.6
v forcats   1.0.1     v stringr   1.6.0
v ggplot2   4.0.1     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.2.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
```

```
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(rstatix)
```

```
Attaching package: 'rstatix'

The following object is masked from 'package:stats':

    filter
```

```r
library(report)
library(apaTables)
```

TIP: Make note of the warning that comes after loading `rstatix` because it says that the use of the `filter()` function now falls under the use of rstatix so the regular stats filter must be called directly if you want to use it but we will be using rstatix in this chapter.

4.3 14.3 What is a t-test?

Now before we get into working inside of R using t-tests first we have to understand what they are and some different ways we use them. A t-test is a inferential statistic that compares **means** while also making note of variability and sample sizes. There are different ways in psychology that we can use these tests.

4.3.1 14.3.1 One-sample t-test

When we want to compare one group against a known value like for example the population average this is when we will use a one-sample t-test because we are only taking one group and applying it to a value that is already known.

4.3.2 14.3.2 Independent-sample t-test

Now when we want compare two completely different groups like say for example gender using males and females this is where we will use the independent sample t-test because the two are independent from each other and results will not rely on the other.

4.3.3 14.3.3 Paired t-test

We use a paired t-test when we want to examine one group that usually experiences a change so lets say for example we want to measure a groups stress levels before and after listening to music we would use a paired t-test

4.3.4 14.3.4 Data-set Example

Now lets practice creating a small data-set that we could see in a real study that you might do for your project.

Lets say a researcher wants to see if listening to calming music will help reduce stress. In the experiment stress is measured on a scale of 0-100.

The hypothesis for this example is that students who listen to music will report lower stress levels.

```r
stress_data <- data.frame(
  group = c(rep("Control", 8), rep("Music", 8)),
  stress = c(75, 80, 78, 82, 79, 76, 81, 74,
             60, 65, 62, 58, 63, 61, 59, 64)
)
```

4.4 14.4 Descriptive Statistics

Now we are going to go through some practice examples to help you get a better use of applying different t-tests to scenarios

4.4.1 14.4.1 Example using data

Now lets run through an example fo using a one sample t-test in a psychology example

Does the average stress level in the **music** group differ from the control group?

First we must separate the music only group into their own section from the control group then use the `get_summary_stats` function to get a summary of the stats from our previous data.

```r
stress_data %>%
  group_by(group) %>%
  get_summary_stats(stress, type = "mean_sd")
```

```
# A tibble: 2 x 5
  group   variable     n  mean    sd
  <chr>   <fct>    <dbl> <dbl> <dbl>
1 Control stress       8  78.1  2.9 
2 Music   stress       8  61.5  2.45
```

As you can see in the output above the results suggest that the people in the music group are less stressed than the people in the control group.

4.4.2 14.4.2 Assumption Testing

Now that we have our data it is important that we check to see if it is normal enough for a t-test by using the Shapiro-wilk test by using our stress variable

4.4.3 14.4.3 Shapiro-Wilk test

Now we must use the Shapiro-wilk test to see if p > .05.

```
stress_data %>%
  group_by(group) %>%
  shapiro_test(stress)
```

```
# A tibble: 2 x 4
  group   variable statistic     p
  <chr>   <chr>        <dbl> <dbl>
1 Control stress       0.954 0.753
2 Music   stress       0.975 0.933
```

As you can see that our p value is > than .05 so we can now run a t-test on it.

4.5 14.5 Independent Sample t-test

Now lets take the data from before and use an independent sample t-test to see if our p value is < .05 making it significant.

4.5.1 14.5.1 Homogeneity of Variance

We need to use the homogeneity of variance test which is the `levene_test` for independent t-tests to see if we are able to conduct a t-test on our data

```
stress_data %>%
  levene_test(stress ~ group)
```

```
# A tibble: 1 x 4
    df1   df2 statistic     p
  <int> <int>     <dbl> <dbl>
1     1    14     0.317 0.583
```

As you see in the output our F-value for this test is .317 which shows how the group variance is different compared to each other. So a low F-value means that variance is similar.

The p-value of .583 shows us that our assumption is met because variance is equal between groups.

4.5.2 14.5.2 Practice with independent sample

Lets now take our data and run it through a t-test

```
t_test(stress_data, stress ~ group)
```

```
# A tibble: 1 x 8
  .y.    group1  group2    n1    n2 statistic    df            p
* <chr>  <chr>   <chr>  <int> <int>     <dbl> <dbl>        <dbl>
1 stress Control Music      8     8      12.4  13.6 0.0000000085
```

As you see in our output we get a lot of data so lets analyze each part of the output

You can see in `group1` that this is the control group and `group2` is the music group.

The `n1` and `n2` represent the participant number in each group which for this data set is 8

The t stat you can find under `statistic` this is 12.38 and it is a large t value which suggests a big difference between the variability of scores from the groups.

Our p value for this test is 8.5e-09. This is a very small p value which indicates a statistically significant result which means that we reject the null hypothesis fro this experiment.

4.5.3 14.5.3 APA Conclusion

Now lets write a APA conclusion for the output that we have just analyzed above.

Participants in the music group reported significantly lowered stress (M = 12.38, SD = 2.45) than the control group (M = 13.61, SD = 2.90), t(16) = 12.38, p < .05.

4.6 14.6 One Sample t-test

Now we can use the same data we just used to compare the mean from one group to a known population mean

4.6.1 14.6.1 Running the t-test

Now lets take the music group and use the known population mean which will be 70 in this example and compare that to the means of the people in our data

```
music_only <- stress_data %>% filter(group == "Music")
t_test(music_only, stress ~ 1, mu = 70)
```

```
# A tibble: 1 x 7
  .y.    group1 group2         n statistic    df         p
* <chr>  <chr>  <chr>      <int>     <dbl> <dbl>     <dbl>
1 stress 1      null model     8     -9.81     7 0.0000242
```

You can see in this output that our sample mean is below the population mean we compared it to by seeing our statistic is `-9.81` suggesting a large difference relative to variability. We also have a smaller p value than .05 so it means we can reject the null and that our sample mean differs from the population mean.

4.7 14.7 Paired t-test

We use a paired t-test when we want to examine two scores before and after a group receives a stimulus.

So lets now do some practice imaging that we weant to see students stress levels before and after listening to some calming music

4.7.1 14.7.1 Practice Problem

First we need to create our data frame so we can use our data

```
paired_data <- data.frame(
  student = 1:8,
  stress_before = c(75, 80, 78, 82, 79, 76, 81, 74),
  stress_after  = c(60, 65, 62, 58, 63, 61, 59, 64)
)
```

4.7.2 14.7.2 Checking for Normality

Now we need to check for normality to see the difference of scores before and after treatment by using the **Shapiro wick test.**

```
paired_data %>%
  mutate(diff = stress_before - stress_after) %>%
  shapiro_test(diff)
```

```
# A tibble: 1 x 3
  variable statistic     p
  <chr>        <dbl> <dbl>
1 diff         0.877 0.177
```

Since we have a $p > .05$ we can now run our paired t-test.

4.7.3 14.7.3 Running the t-test

Now we are going to be running our paired t-test to see if stress significantly decreased from the same participants after they have received the treatment.

```
t.test(paired_data$stress_before,
       paired_data$stress_after,
       paired = TRUE)
```

```

    Paired t-test

data:  paired_data$stress_before and paired_data$stress_after
```

```
t = 10.673, df = 7, p-value = 1.39e-05
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 12.94169 20.30831
sample estimates:
mean difference
         16.625
```

4.7.4 14.7.4 APA Conclusion

Now that we have our output from our t-test we can use the `report` package to create an APA style conclusion for us automatically

```r
paired_test <- t.test(paired_data$stress_before,
                      paired_data$stress_after,
                      paired = TRUE)

report(paired_test)
```

```
For paired samples, 'repeated_measures_d()' provides more options.

Effect sizes were labelled following Cohen's (1988) recommendations.

The Paired t-test testing the difference between paired_data$stress_before and
paired_data$stress_after (mean difference = 16.62) suggests that the effect is
positive, statistically significant, and large (difference = 16.62, 95% CI
[12.94, 20.31], t(7) = 10.67, p < .001; Cohen's d = 3.77, 95% CI [1.71, 5.81])
```

Our output shows us that the effect is positive and significant and the treatment was effective.

4.8 14.8 APA Tables

APA tables is a very useful package that allows us to present our data in APA formatted tables

4.8.1 14.8.1 Creating an APA Table

Now we are going to create an APA table for us to use that combines our control and music groups.

```r
apa.1way.table(
  data = stress_data,
  dv = stress,
  iv = group,
  filename = "stress_ttest_table.doc"
)
```

```


Descriptive statistics for stress as a function of group.  

   group     M   SD
 Control 78.12 2.90
   Music 61.50 2.45

Note. M and SD represent mean and standard deviation, respectively.
 
```

As you can see `APA tables` creates a simple and detailed table that you can put into a word document when you are writing your final paper.

4.9 14.9 Report Package

The `report` package automatically formats any data or results into a APA style sentence that works very will for research papers

It generates an APA style paragraph that you can put right into your paper

```r
report(paired_test)
```

```
For paired samples, 'repeated_measures_d()' provides more options.

Effect sizes were labelled following Cohen's (1988) recommendations.

The Paired t-test testing the difference between paired_data$stress_before and
```

```
paired_data$stress_after (mean difference = 16.62) suggests that the effect is
positive, statistically significant, and large (difference = 16.62, 95% CI
[12.94, 20.31], t(7) = 10.67, p < .001; Cohen's d = 3.77, 95% CI [1.71, 5.81])
```

5 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

References

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.