RMS textbook

Pilot, Lucas, and Murray

2025-10-28

Table of contents

Preface

This is a book intended to be used 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.

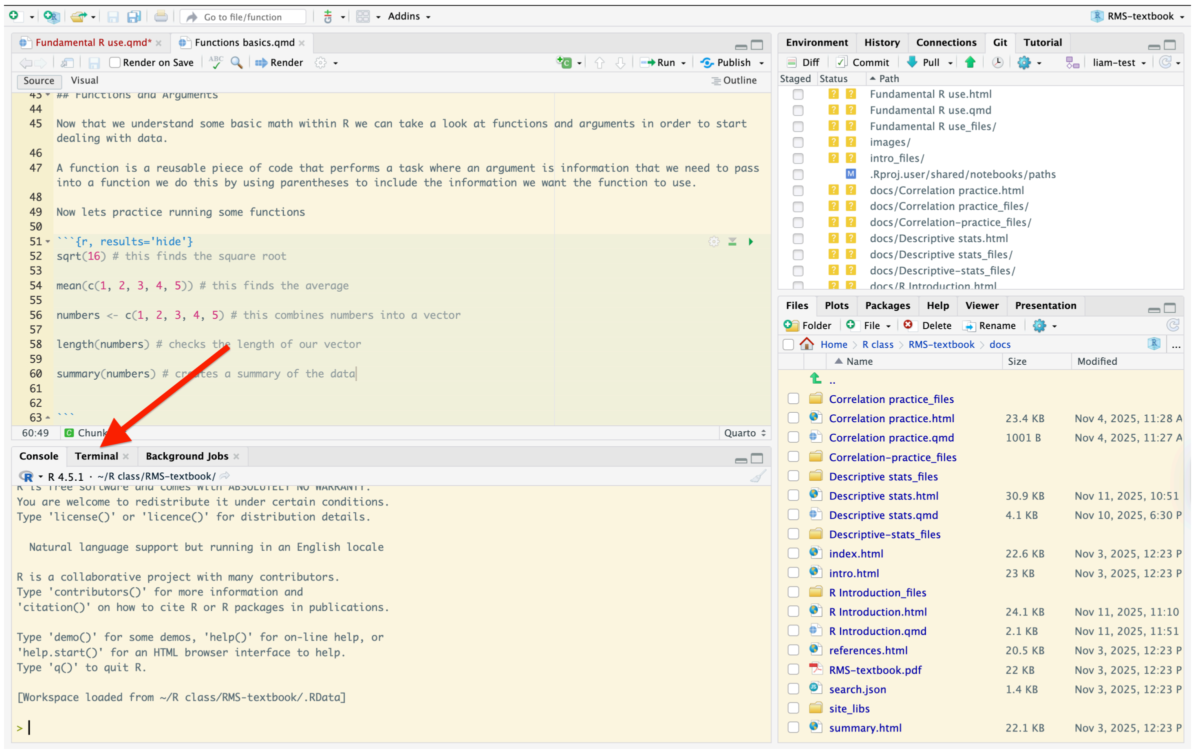
- How to use packages and what you will need to do in order to update them.
- How to create and manipulate data.
- How to use functions, objects, and pipes.
- Common errors and different ways to deal with them.
- And the main types of data that you will be running into in Rstudio.

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]

>
```

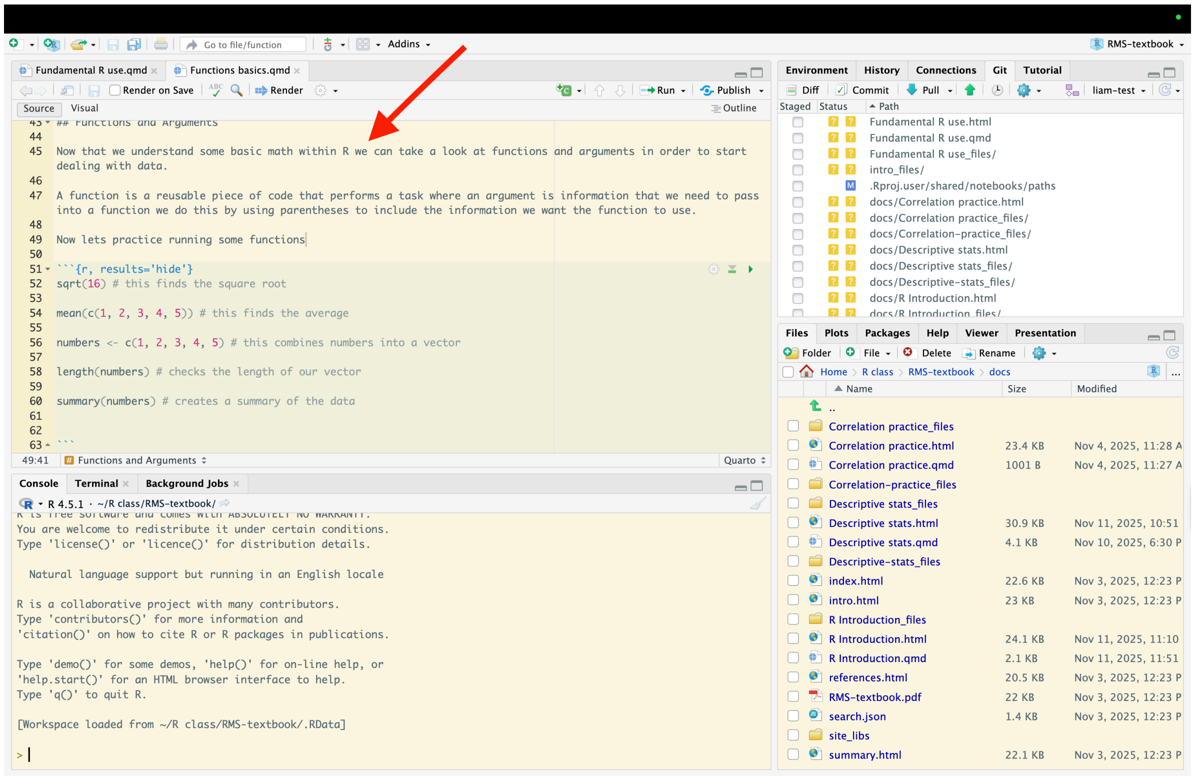
Staged	Status	Path
	? ?	Fundamental R use.html
	? ?	Fundamental R use.qmd
	? ?	Fundamental R use_files/
	? ?	images/
	? ?	intro_files/
	M	.Rproj.user/shared/notebooks/paths
	? ?	docs/Correlation practice.html
	? ?	docs/Correlation practice_files/
	? ?	docs/Correlation-practice_files/
	? ?	docs/Descriptive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

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

2.2.2 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


```
~~~

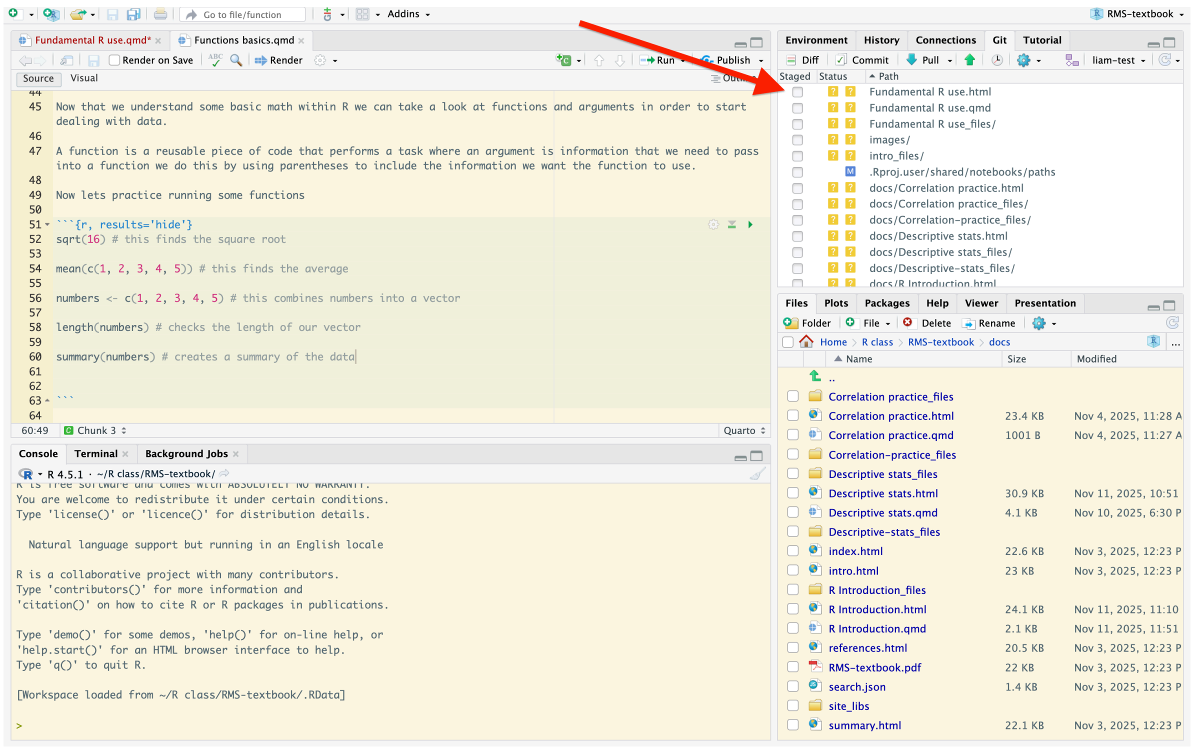
```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  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/

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


```
~~~

```
60:49   Chunk 3   Quarto
```

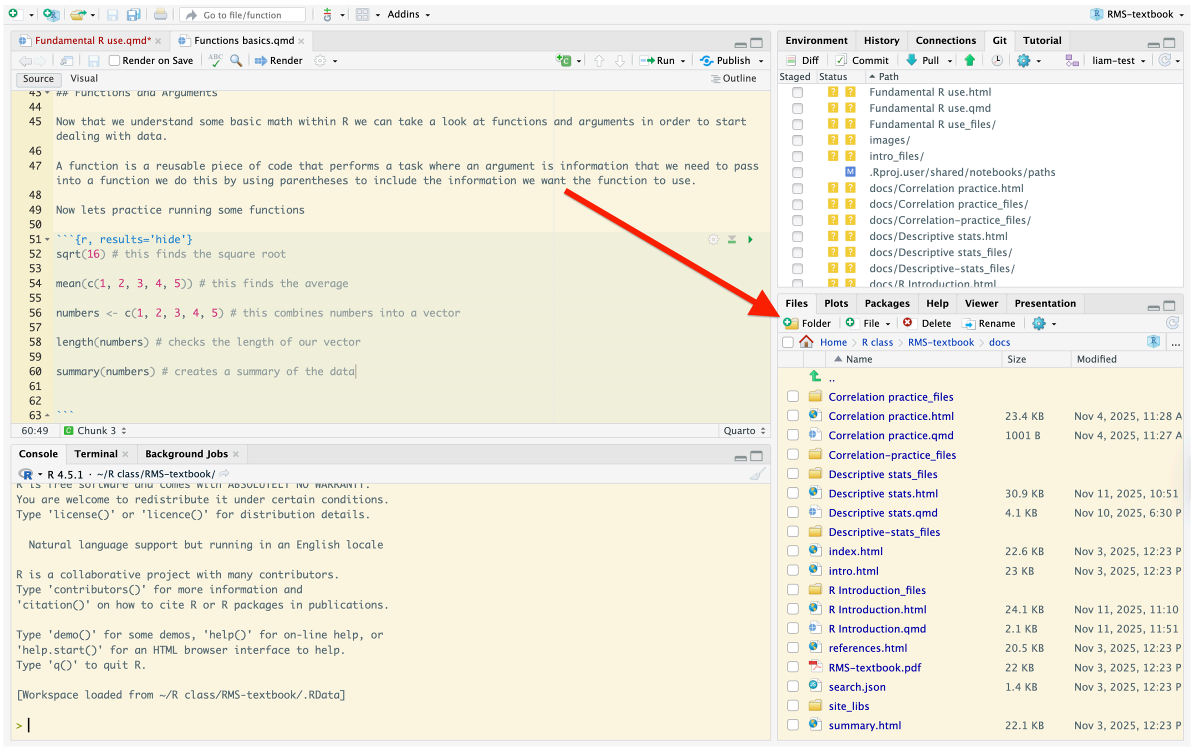
```
R 4.5.1 · ~/R class/RMS-textbook/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  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

2.2.4 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 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

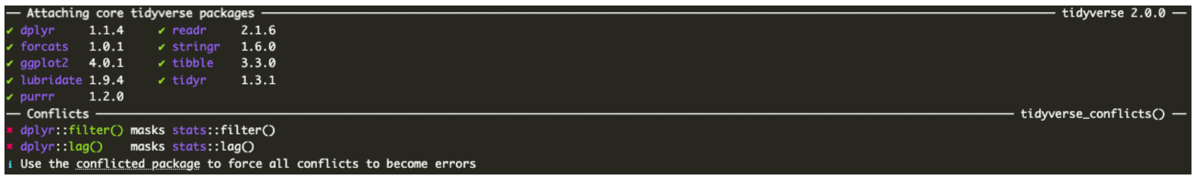
Whenever you want to install a package you will have to run a command. When you are working in a document you will need to load your packages each time or else **your code will not run because the package has not been reloaded.** You can install packages by going to your console frame in the bottom left. 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.

```
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

```
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

```
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.

```
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.

```
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.

```r
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 `<-`.

```r
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.

```r
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.

```r
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.

```
typeof(42)
```

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

```
typeof(FALSE)
```

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

```
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.

```
    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 How to plan an experiment

4.1 Experimental Research

Allows us to study causes of behavior.

4.2 Three components of Experimental Research

1. Manipulate a variable
    - Exercise experimental control
2. Systematically put participants in groups
    - Ensure equivalent groups
3. Control extraneous variables
    - Make sure factors that are unimportant don't influence results

4.3 Independent variable (IV)

The variable that the researcher manipulates. All experimental research must have AT LEAST one. Researchers can manipulate the environment, the instructions, or they may use an invasive variable (like giving someone a caffeine pill).

Must have at least 2 levels

Ex. Temperature may have two levels; hot and cold

4.4 Subject (Participant) Variables

Based off of a personal characteristic. Something you cannot manipulate in the lab.

*Ex. Ethnicity/Race, Hob*bies

4.5 Evaluating Your Independent Variable

A bad independent variable can result in a failed experiment.

Pilot Study

Test your experiment on a small group of people to ensure that it works.

Manipulation Check

Done to ensure that the level of your IV manipulation is strong enough.

Ex. Is 5mg of caffeine enough or should I use 10mg?

4.6 Dependent Variable

The variable a researcher measures.

Experiments must have AT LEAST one.

Ex. Heart rate, response to questionnaire, performance on test

4.7 Groups in an Experiment

Experimental

The group that receives the independent variable manipulation.

Ex. In a study about sleep, this group is required to stay awake for 2 days.

Control

The group that is not exposed to any independent variable manipulation.

Ex. In the same sleep study, this group sleeps however much they usually sleep.

4.8 Assigning Participants

Simple Random Assignment

Everyone has an equal chance of being assigned to any group/condition.

Ex. Roll dice

In this situation people with any attribute are equally likely to be in either group.

Matched Random Assignment

Participants are matched into homogeneous blocks. Participants in each block are then randomly assigned to conditions.

In this situation conditions will be similar along specific dimensions.

4.9 Within Subject Designs (repeated measures)

Participants are exposed to ALL conditions in an experiment.

No need for random assignment.

Ex. *In an experiment testing a new drug all participants receive all doses of the drug (5mg, 10mg, 15mg)*

Pros

More powerful

Cons

Order Effects

4.10 Power

The ability to detect IV effects.

Requires fewer participants.

4.11 Order Effects

Carryover effects

One condition influences the following condition.

Practice effects

Participants have learned how to perform from previous experimental trials.

Fatigue effects

Participants are tired, bored, have no energy over time.

Sensitization

Participants realize the hypothesis being tested and do not perform naturally.

Counterbalancing

can be used to correct for order effects.

A researcher presents the levels of the IV in different orders for different participants.

4.11.1 Between-Subjects Design (Independent Measures)

Participants experience only one condition of the IV.

Typically requires random assignment.

Ex. In an experiment testing a new drug each participant will receive only one of the three levels of drug dosage (5mg or 10 mg or 15 mg).

4.12 Experimental Control

Treatment effect

Systematic differences due to the IV

Confounds

Variable other than the IV that differs systematically between conditions.

Confounds invalidate your experiment because, if confounds are present, it is unclear whether the observed differences are due to the IV or the confound.

Must be eliminated to draw accurate conclusions.

Error

Unsystematic effects due to extraneous (uncontrolled) variables.

4.13 Sources of Error

Individual Differences

Transient states

Environmental factors

Differential treatment

Measurement error

4.14 Types of Validity

Internal Validity

Degree to which we can draw accurate conclusions about the effects of the IV

Gain internal validity when all confounds are eliminated and you can conclude that the observed differences were due to the IV.

Has experimental control.

External Validity

Inverse relationship with internal validity

The greater experimental control in your experiment, the less likely it will be externally valid or generalizable to the "real world".

Internal validity is more critical and desirable than external validity.

4.15 Threats to internal validity

Biased assignment of participants

Occurs when random assignment isn't possible or doesn't produce equivalent groups.

Differential attrition

Participants drop out of the experiment differently across levels of the IV.

Demand Characteristics

Participants perform in the way they believe the experimenter wants them to.

Placebo Effects

Change as the result of the mere suggestion of change.

Pretest sensitization

Exposure to pretest affects one IV level differently than another

History

External events that participants experience affect one level of the IV differently than another

Experimenter expectancy effects

Experimenter with certain expectations interacts with participants differently

4.15.1 Reducing threats to validity

Double Blind Procedure

The researcher administering the IV and the participant both do not know what level of the IV is being administered.

Can eliminate expectancy effects and demand characteristics

5 Measuring behavior

5.1 Types

- Observational
- Physiological
- Self-report

5.2 Scales of Measurement

5.2.1 Nominal Scales

Numbers are assigned as labels for characteristics or behaviors.

Provides the least amount of information.

Ex. Jersey Number

5.2.2 Ordinal Scale

Rank ordering of people's behaviors or characteristics.

Doesn't specify the distance between participants on the variable being measured.

Ex. Sizes at a fast food restaurant: baby, small, medium, large, ex large, super size, super duper size

5.2.3 Interval Scale

Equal distance between the numbers reflect equal differences between participants

Does not have a "true" zero point

Ex. Temperature, IQ score

5.2.4 Ratio Scale

Has a “true” zero point.

Provides greatest amount of information.

Should be used when possible.

ex. duration, weight, accuracy

5.3 Central Tendency

A descriptive measure which represents the entire distribution of scores (mean, median, mode).

Goal: Find a single value that is representative of all the data.

Can condense large data set into a single value.

Allows comparison of 2 or more data sets using the central tendency.

5.3.1 The Mean

Most commonly used measure of central tendency.

Used in interval or ratio scales.

5.3.1.1 How to compute:

Compute the sum of all scores ( )

Divide the sum by the number of scores.

In manuscripts, the sample mean is identified as “*M*”

The sum ( ) of all scores (X) = (X)

4 + 5 + 3 = 12

Divide the sum by the number of scores (N) = (X/N)

12/3 = 4

Weekly high temperatures in Chicago last winter:

29, 31, 28, 32, 29, 27, 55

What was the average high temperature in Chicago last winter?

All distances below the mean are equal to all the distances above the mean.

Changing any score (adding or subtracting) will influence the mean.

5.3.1.2 When you shouldn't use the mean

The mean is not appropriate for nominal or ordinal scales.

- It is impossible to calculate.

The mean is not appropriate when you have extreme scores (outliers).

- The mean will be pulled towards the extreme score, rendering it no longer representative of the rest of the data.

The mean is not appropriate when there is missing data

5.3.2 The Median

Scores are listed in order from smallest to largest

The median is the midpoint, it equally divides the scores

When you have an even number of scores you take the average of the two middle scores.

The median can be used on ordinal, interval, or ratio scales.

The median is unaffected by extreme scores (outliers).

Weekly high temperatures in Chicago last winter:

29, 31, 28, 32, 29, 27, 55

What was the median temperature in Chicago last winter?

5.3.3 The Mode

The most frequently occurring score.

Can be used on ALL scales of measurement.

Weekly high temperatures in Chicago last winter:

29, 31, 28, 32, 29, 27, 55

What was the mode temperature in Chicago last winter?

6 Measurement

6.1 Measurement

Measurement Error

Variability in scores due to factors that distort the true score.

True Score

The score a participant would obtain if a measure were perfect and we could measure without error.

Measurement error + True Score = Observed score

6.2 Sources of Measurement Error

Transient States

Temporary state

Ex. mood

Stable Attribute

A lasting state

Ex. Ambitious personality

Situational factor

Research setting (Ex. Noise/temperature in the room)

Characteristics of the measure

The measure itself is ambiguous or too long

Mistakes in recording

Incorrect data

6.3 Reliability

Consistency/dependability of the measuring technique

Inverse relationship with measurement error

If observed score is close to the true score, your measure has high reliability

Can be assessed using several measurements of the same behavior and comparing to see if they resulted in similar scores (typically through a correlation)

Correlation Coefficient

Value that describes relationship between two measures

Ranges from -1.00 to +1.00, sign indicates direction

Correlation of .00 indicates no relationship

6.4 Forms of Reliability

Inter-rater Reliability

Consistency among two or more researchers who observe and record participants' behavior

Test-Retest Reliability

Consistency of responses on a measure over time, use the same measure twice and evaluate the correlation.

The results of a reliable measure should not change over time.

Inter-Item Reliability

Consistency between items on a scale.

Tells the researcher whether the items on the scale are measuring the same thing

If the items do not measure the same thing, measurement error increases and reliability decreases.

6.5 Indices of Inter-Item Reliability

Item-total correlation

The correlation between one item and the sum of all other items on a scale.

Split-half reliability

Divide items on a scale into two sections and examine the correlation between the sections.

Cronbach's Alpha ( )

The average of all possible split-half reliabilities

Most frequently used

$> .70$ considered acceptable

6.6 Increasing Reliability

Standardize how measure is administered

Clarify instructions and questions

Train researchers/coders

Minimize errors in coding data

6.7 Validity

How accurate is a measure at estimating what it is attempting to assess?

Do differences in scores truly reflect differences in what you are trying to measure?

6.8 Forms of Validity

Face Validity

The extent to which an assessment appears to describe what it is supposed to measure.

Does not actually impact the "true" validity

Construct Validity

How well does a measurement of a hypothetical construct relate to other measures.

- **Hypothetical Construct**
  - Something that cannot be directly observed, but is inferred based on observation or experience.
  - *Ex. Personality, Confidence*

Convergent Validity

A measure correlates with other measures that it should correlate with

Discriminant Validity

A measure does not correlate with other measures that it should not correlate with

6.9 Bias

Test Bias

When the validity of a measure is lower for some groups than others.

7 Variability

NEED RAFALIB PACKAGE

7.1 Variability

A quantitative measure of difference between a set of scores that describes how scores are scattered around a central point.

Descriptive Variability

Assesses spread or clustering of scores.

Inferential Variability

Assesses how accurately one individual score/sample represents the population.

Used to detect patterns, variability influences how easily those patterns are detected.

Variability can be small or large.

Small indicates that scores are very clustered together.

Large indicates that scores are widely dispersed.

7.2 Measures of Variability

7.2.1 Range

Total distance covered by the distribution, from highest to lowest value, also gives information about how many categories there are.

Relies on two values (extremes), ignores all others

Range = Maximum Score − Minimum Score

7.2.1.1 Calculate in R

Let's use R to work out an example. First, use the `sample` function to create 10 scores from 1:10 and assign that list of numbers (called a vector) to the object 'x'.

```r
# random sample of 10 scores from 1-10
x <- sample(1:10, 10)
```

The `range` function will produce the two values we use to calculate the range, the highest and lowest.

```r
# gives us the extreme values
range(x)
```

The `max` and `min` function will give us the largest and smallest numbers respectively.

```r
# gives us the maximum value
max(x)

# gives us the minimum value
min(x)
```

Now we can use these functions to calculate the mean.

```r
max(x) - min(x)
```

7.2.2 Variance & Standard Deviation

Calculated using all scores in a distribution

Most commonly used measure of variability

Describes average distance between a score and the mean

Used with interval and ratio scales

variance definition of variance

standard deviation definition of standard deviation

7.2.2.1 Calculate by hand

1. Calculate the mean.
2. Subtract the mean from each individual score to get a difference score for each participant. Make sure that if you add all of the difference scores they equal zero.
3. Square the difference scores to get squared scores.
4. Add all of the squared scores to get the Sum of Squared Deviations (SS).
5. Divide the SS by the size of your population (N) or sample (n-1) to get the variance (^2 or s^2).
6. Find the square root of ^2 to find the standard deviation(or s).

7.2.2.2 Calculate in R (long)

Start with a simple vector we will store in the object 'y'.

```
# data
y <- c( 1, 2, 3, 4, 5)
```

1. Calculate the mean.

```
mean(y)
```

```
[1] 3
```

2. Subtract the mean from each individual score to get a difference score for each participant. Make sure that if you add all of the difference scores they equal zero.

```
diff_score <- y - mean(y)
```

3. Square the difference scores to get squared scores.

```
sq_score <- diff_score^2
```

4. Add all of the squared scores to get the Sum of Squared Deviations (SS).

```
SS <- sum(sq_score)
```

5. Divide the SS by the size of your population (N) or sample (n-1) to get the variance (^2 or s^2).

```r
# variance for a population
pop_var <- SS/length(y)

# variance for a sample
sample_var <- SS/(length(y) - 1)
```

6. Find the square root of ^2 to find the standard deviation(or s).

```r
# population standard deviation
sqrt(pop_var)
```

```
[1] 1.414214
```

```r
# sample standard deviation
sqrt(sample_var)
```

```
[1] 1.581139
```

7.2.2.3 Calculate in R (short)

Start with the same data

```r
# data
y <- c( 1, 2, 3, 4, 5)
```

1. Calculate variance. For the calculation of population variance and standard deviation we can use the `rafalib` package. For populations we will use the `popvar` function and for sample vaiance we will use the `var` function from base R.

```r
# population variance
library(rafalib)
popvar(y)
```

```
[1] 2
```

```r
# sample variance
var(y)
```

```
[1] 2.5
```

2. Calculate stabdard deviation

```r
# population stabdard deviation
popsd(y)
```

```
[1] 1.414214
```

```r
# sample sd
sd(y)
```

```
[1] 1.581139
```

7.3 Population v Sample

Population is EVERYONE.

Variance (^2) = SS/N

Standard Deviation ( ) = √(SS/N)

Sample is a subset of everyone.

Variance (s2) = SS/n-1.

Standard Deviation (s) = √(SS/(n-1)).

We use a different formula for samples because we are using limited information from a small group (the sample) to draw inferences about a larger group (the population).

Samples have less variability than populations.

7.4 Biased and Unbiased Statistics

Biased Statistics

The average value overestimates or underestimates the population parameter

i.e. the sample before adjustment (n-1)

Unbiased Statistics

The average value is equal to the population parameter.

i.e. the sample after adjustment (n-1).