# 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.

```r
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.

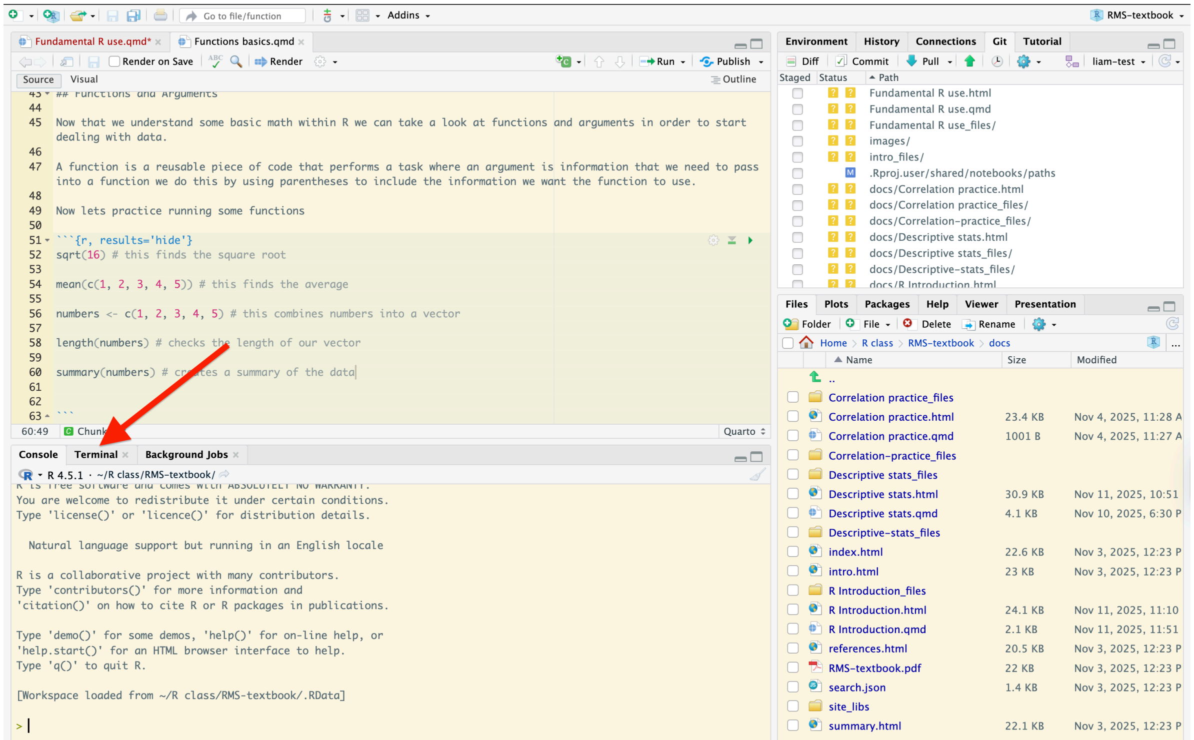
- How to use packages and what you will need to do in order to update them.
- How to create and manipulate data.
- How to use functions, objects, and pipes.
- Common errors and different ways to deal with them.
- And the main types of data that you will be running into in Rstudio.

## 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

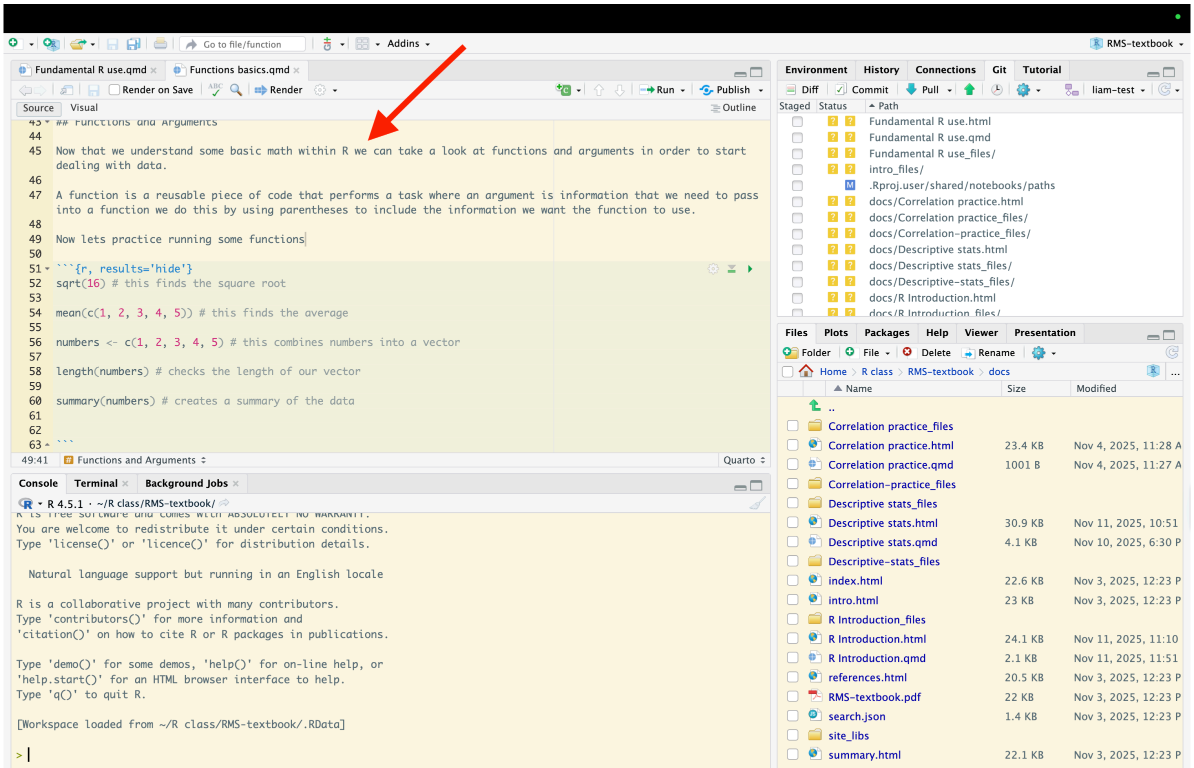
### 2.2.1 2.2.1 Console Frame

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.



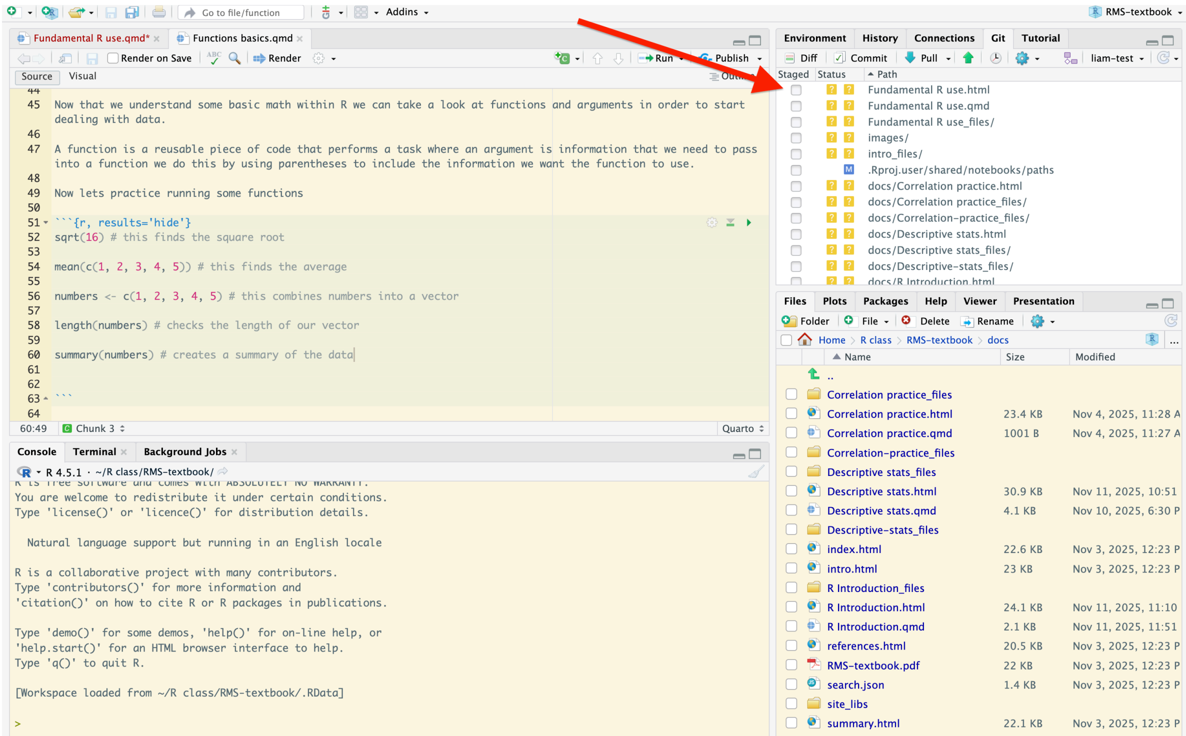
### 2.2.2 2.2.2 Source Frame

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.



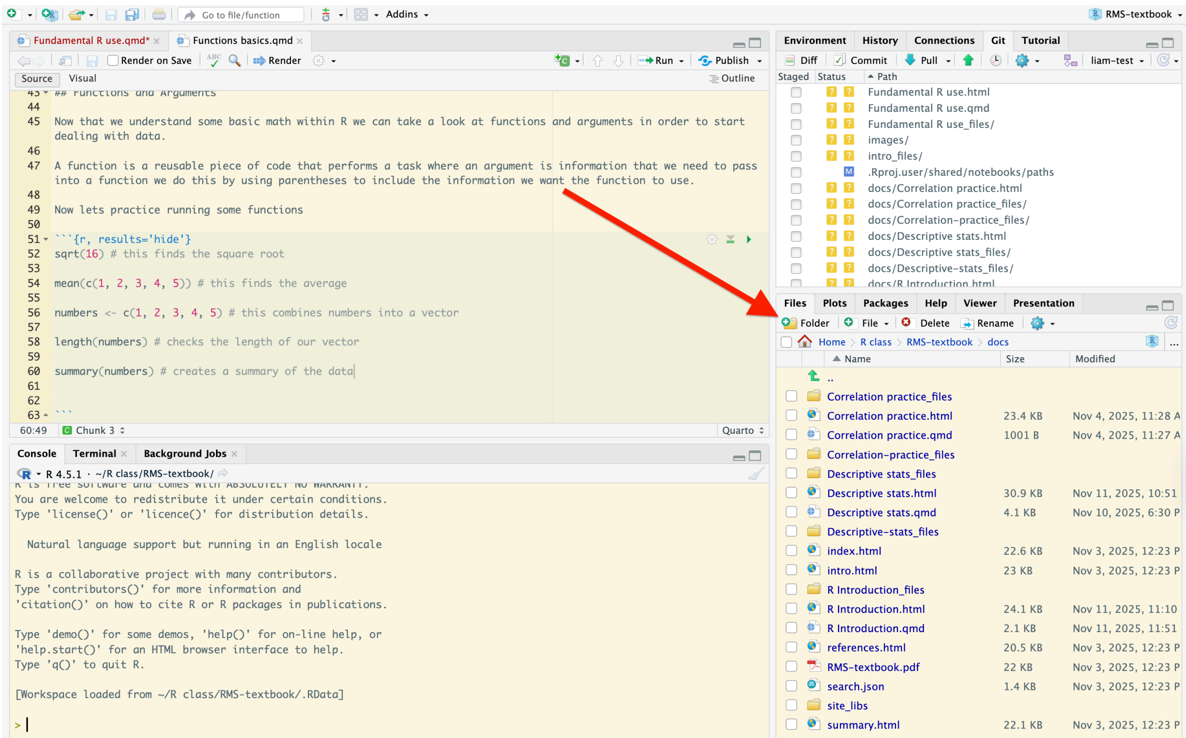
## 2.2.3 2.2.3 Environment/History

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.




## 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.



**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:


**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

```r
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.

```r
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.

```
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.

```
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`.

```
# 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.

```
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'.

```
# 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.

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

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

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

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

Now we can use these functions to calculate the mean.

```
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

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

```
[1] 1.414214
```

```
# 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).

## 7.5 Transforming Data Sets

### 7.5.1 Adding a constant

When you add a value to every score in the data set it does not change the standard deviation. Try it out calculate the sample standard deviation of the example object.

```
example_1 <- c(1, 2, 3, 4, 5)

s <- sd(example_1)
s
```

```
[1] 1.581139
```

Now, let's add a constant value of 2 to every score in the example object.

```
example_2 <- example_1 + 2
example_2
```

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

Let's see if the sample standard deviation has changed.

```
s <- sd(example_2)
```

There is no difference because the distance between scores is unchanged. The shape of the distribution is the same, just shifted to the right.

```
<theme> List of 144
 $ line                          : <ggplot2::element_line>
  ..@ colour       : chr "black"
  ..@ linewidth    : num 0.5
  ..@ linetype     : num 1
  ..@ lineend      : chr "butt"
  ..@ linejoin     : chr "round"
  ..@ arrow        : logi FALSE
  ..@ arrow.fill   : chr "black"
  ..@ inherit.blank: logi TRUE
 $ rect                          : <ggplot2::element_rect>
  ..@ fill         : chr "white"
```

```
  ..@ colour       : chr "black"
  ..@ linewidth    : num 0.5
  ..@ linetype     : num 1
  ..@ linejoin     : chr "round"
  ..@ inherit.blank: logi TRUE
 $ text                             : <ggplot2::element_text>
  ..@ family       : chr ""
  ..@ face         : chr "plain"
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : chr "black"
  ..@ size         : num 11
  ..@ hjust        : num 0.5
  ..@ vjust        : num 0.5
  ..@ angle        : num 0
  ..@ lineheight   : num 0.9
  ..@ margin       : <ggplot2::margin> num [1:4] 0 0 0 0
  ..@ debug        : logi FALSE
  ..@ inherit.blank: logi TRUE
 $ title                            : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : NULL
  ..@ vjust        : NULL
  ..@ angle        : NULL
  ..@ lineheight   : NULL
  ..@ margin       : NULL
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ point                            : <ggplot2::element_point>
  ..@ colour       : chr "black"
  ..@ shape        : num 19
  ..@ size         : num 1.5
  ..@ fill         : chr "white"
  ..@ stroke       : num 0.5
  ..@ inherit.blank: logi TRUE
 $ polygon                          : <ggplot2::element_polygon>
```

```
  ..@ fill         : chr "white"
  ..@ colour       : chr "black"
  ..@ linewidth    : num 0.5
  ..@ linetype     : num 1
  ..@ linejoin     : chr "round"
  ..@ inherit.blank: logi TRUE
 $ geom                            : <ggplot2::element_geom>
  ..@ ink        : chr "black"
  ..@ paper      : chr "white"
  ..@ accent     : chr "#3366FF"
  ..@ linewidth  : num 0.5
  ..@ borderwidth: num 0.5
  ..@ linetype   : int 1
  ..@ bordertype : int 1
  ..@ family     : chr ""
  ..@ fontsize   : num 3.87
  ..@ pointsize  : num 1.5
  ..@ pointshape : num 19
  ..@ colour     : NULL
  ..@ fill       : NULL
 $ spacing                         : 'simpleUnit' num 5.5points
  ..- attr(*, "unit")= int 8
 $ margins                         : <ggplot2::margin> num [1:4] 5.5 5.5 5.5 5.5
 $ aspect.ratio                    : NULL
 $ axis.title                      : NULL
 $ axis.title.x                    : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : NULL
  ..@ vjust        : num 1
  ..@ angle        : NULL
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 2.75 0 0 0
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.title.x.top                : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
```

```
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : NULL
  ..@ vjust        : num 0
  ..@ angle        : NULL
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 0 0 2.75 0
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.title.x.bottom             : NULL
 $ axis.title.y                    : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : NULL
  ..@ vjust        : num 1
  ..@ angle        : num 90
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 0 2.75 0 0
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.title.y.left               : NULL
 $ axis.title.y.right              : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : NULL
  ..@ vjust        : num 1
  ..@ angle        : num -90
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 0 0 0 2.75
  ..@ debug        : NULL
```

```
 ..@ inherit.blank: logi TRUE
$ axis.text                       : <ggplot2::element_text>
 ..@ family       : NULL
 ..@ face         : NULL
 ..@ italic       : chr NA
 ..@ fontweight   : num NA
 ..@ fontwidth    : num NA
 ..@ colour       : NULL
 ..@ size         : 'rel' num 0.8
 ..@ hjust        : NULL
 ..@ vjust        : NULL
 ..@ angle        : NULL
 ..@ lineheight   : NULL
 ..@ margin       : NULL
 ..@ debug        : NULL
 ..@ inherit.blank: logi TRUE
$ axis.text.x                     : <ggplot2::element_text>
 ..@ family       : NULL
 ..@ face         : NULL
 ..@ italic       : chr NA
 ..@ fontweight   : num NA
 ..@ fontwidth    : num NA
 ..@ colour       : NULL
 ..@ size         : NULL
 ..@ hjust        : NULL
 ..@ vjust        : num 1
 ..@ angle        : NULL
 ..@ lineheight   : NULL
 ..@ margin       : <ggplot2::margin> num [1:4] 2.2 0 0 0
 ..@ debug        : NULL
 ..@ inherit.blank: logi TRUE
$ axis.text.x.top                 : <ggplot2::element_text>
 ..@ family       : NULL
 ..@ face         : NULL
 ..@ italic       : chr NA
 ..@ fontweight   : num NA
 ..@ fontwidth    : num NA
 ..@ colour       : NULL
 ..@ size         : NULL
 ..@ hjust        : NULL
 ..@ vjust        : num 0
 ..@ angle        : NULL
 ..@ lineheight   : NULL
```

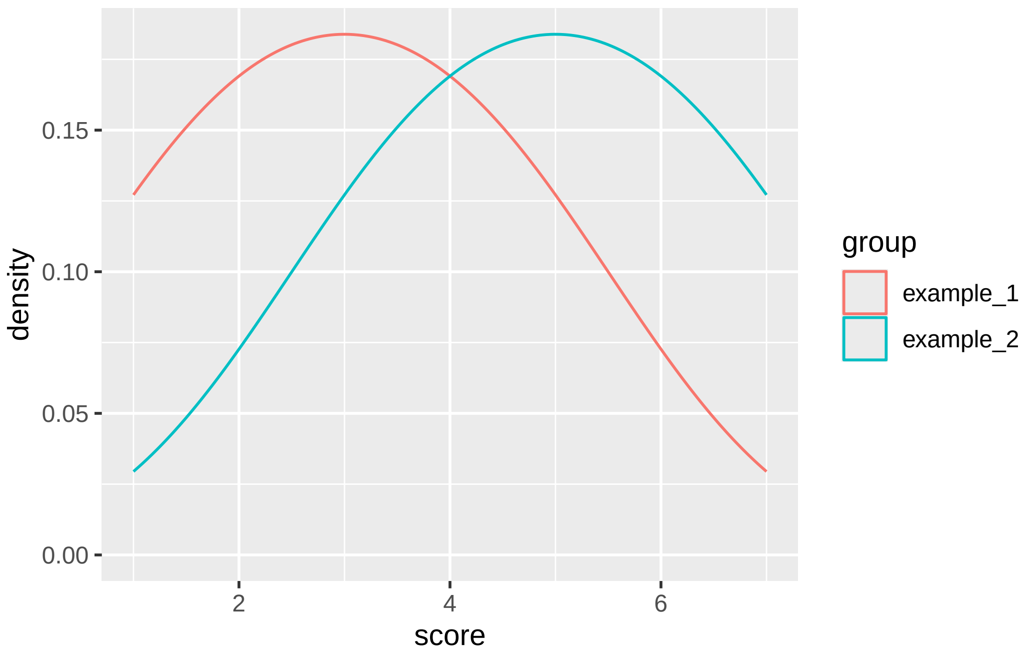
```
  ..@ margin       : <ggplot2::margin> num [1:4] 0 0 2.2 0
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.text.x.bottom              : NULL
 $ axis.text.y                     : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : num 1
  ..@ vjust        : NULL
  ..@ angle        : NULL
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 0 2.2 0 0
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.text.y.left                : NULL
 $ axis.text.y.right               : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
  ..@ size         : NULL
  ..@ hjust        : num 0
  ..@ vjust        : NULL
  ..@ angle        : NULL
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 0 0 0 2.2
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.text.theta                 : NULL
 $ axis.text.r                     : <ggplot2::element_text>
  ..@ family       : NULL
  ..@ face         : NULL
  ..@ italic       : chr NA
  ..@ fontweight   : num NA
  ..@ fontwidth    : num NA
  ..@ colour       : NULL
```

```
  ..@ size         : NULL
  ..@ hjust        : num 0.5
  ..@ vjust        : NULL
  ..@ angle        : NULL
  ..@ lineheight   : NULL
  ..@ margin       : <ggplot2::margin> num [1:4] 0 2.2 0 2.2
  ..@ debug        : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.ticks                      : <ggplot2::element_line>
  ..@ colour       : NULL
  ..@ linewidth    : NULL
  ..@ linetype     : NULL
  ..@ lineend      : NULL
  ..@ linejoin     : NULL
  ..@ arrow        : logi FALSE
  ..@ arrow.fill   : NULL
  ..@ inherit.blank: logi TRUE
 $ axis.ticks.x                    : NULL
 $ axis.ticks.x.top                : NULL
 $ axis.ticks.x.bottom             : NULL
 $ axis.ticks.y                    : NULL
 $ axis.ticks.y.left               : NULL
 $ axis.ticks.y.right              : NULL
 $ axis.ticks.theta                : NULL
 $ axis.ticks.r                    : NULL
 $ axis.minor.ticks.x.top          : NULL
 $ axis.minor.ticks.x.bottom       : NULL
 $ axis.minor.ticks.y.left         : NULL
 $ axis.minor.ticks.y.right        : NULL
 $ axis.minor.ticks.theta          : NULL
 $ axis.minor.ticks.r              : NULL
 $ axis.ticks.length               : 'rel' num 0.5
 $ axis.ticks.length.x             : NULL
 $ axis.ticks.length.x.top         : NULL
 $ axis.ticks.length.x.bottom      : NULL
 $ axis.ticks.length.y             : NULL
 $ axis.ticks.length.y.left        : NULL
 $ axis.ticks.length.y.right       : NULL
 $ axis.ticks.length.theta         : NULL
 $ axis.ticks.length.r             : NULL
 $ axis.minor.ticks.length         : 'rel' num 0.75
 $ axis.minor.ticks.length.x       : NULL
 $ axis.minor.ticks.length.x.top   : NULL
```

```
$ axis.minor.ticks.length.x.bottom: NULL
$ axis.minor.ticks.length.y       : NULL
$ axis.minor.ticks.length.y.left  : NULL
$ axis.minor.ticks.length.y.right : NULL
$ axis.minor.ticks.length.theta   : NULL
$ axis.minor.ticks.length.r       : NULL
$ axis.line                       : <ggplot2::element_line>
 ..@ colour       : NULL
 ..@ linewidth    : NULL
 ..@ linetype     : NULL
 ..@ lineend      : chr "square"
 ..@ linejoin     : NULL
 ..@ arrow        : logi FALSE
 ..@ arrow.fill   : NULL
 ..@ inherit.blank: logi TRUE
$ axis.line.x                     : NULL
$ axis.line.x.top                 : NULL
$ axis.line.x.bottom              : NULL
$ axis.line.y                     : NULL
$ axis.line.y.left                : NULL
$ axis.line.y.right               : NULL
$ axis.line.theta                 : NULL
$ axis.line.r                     : NULL
$ legend.background               : <ggplot2::element_rect>
 ..@ fill         : NULL
 ..@ colour       : logi NA
 ..@ linewidth    : NULL
 ..@ linetype     : NULL
 ..@ linejoin     : NULL
 ..@ inherit.blank: logi TRUE
$ legend.margin                   : NULL
$ legend.spacing                  : 'rel' num 2
$ legend.spacing.x                : NULL
$ legend.spacing.y                : NULL
$ legend.key                      : NULL
$ legend.key.size                 : 'simpleUnit' num 1.2lines
  ..- attr(*, "unit")= int 3
$ legend.key.height               : NULL
$ legend.key.width                : NULL
$ legend.key.spacing              : NULL
$ legend.key.spacing.x            : NULL
$ legend.key.spacing.y            : NULL
$ legend.key.justification        : NULL
```

```
$ legend.frame                   : NULL
$ legend.ticks                   : NULL
$ legend.ticks.length            : 'rel' num 0.2
$ legend.axis.line               : NULL
$ legend.text                    : <ggplot2::element_text>
 ..@ family       : NULL
 ..@ face         : NULL
 ..@ italic       : chr NA
 ..@ fontweight   : num NA
 ..@ fontwidth    : num NA
 ..@ colour       : NULL
 ..@ size         : 'rel' num 0.8
 ..@ hjust        : NULL
 ..@ vjust        : NULL
 ..@ angle        : NULL
 ..@ lineheight   : NULL
 ..@ margin       : NULL
 ..@ debug        : NULL
 ..@ inherit.blank: logi TRUE
$ legend.text.position           : NULL
$ legend.title                   : <ggplot2::element_text>
 ..@ family       : NULL
 ..@ face         : NULL
 ..@ italic       : chr NA
 ..@ fontweight   : num NA
 ..@ fontwidth    : num NA
 ..@ colour       : NULL
 ..@ size         : NULL
 ..@ hjust        : num 0
 ..@ vjust        : NULL
 ..@ angle        : NULL
 ..@ lineheight   : NULL
 ..@ margin       : NULL
 ..@ debug        : NULL
 ..@ inherit.blank: logi TRUE
$ legend.title.position          : NULL
$ legend.position                : chr "right"
$ legend.position.inside         : NULL
$ legend.direction               : NULL
$ legend.byrow                   : NULL
$ legend.justification           : chr "center"
$ legend.justification.top       : NULL
$ legend.justification.bottom    : NULL
```

```
$ legend.justification.left       : NULL
$ legend.justification.right      : NULL
$ legend.justification.inside     : NULL
 [list output truncated]
@ complete: logi TRUE
@ validate: logi TRUE
```




### 7.5.2 Multiplying by a constant

When you multiply every score in the data set by a constant changes standard deviation. Try it out. Calculate the sample standard deviation of the example object.

```r
example_1 <- c(1, 2, 3, 4, 5)

s <- sd(example_1)
s
```

```
[1] 1.581139
```

Now, let's multiply every score in the dataset by a contant of 3.

```r
example_2 <- example_1 * 3
example_2
```

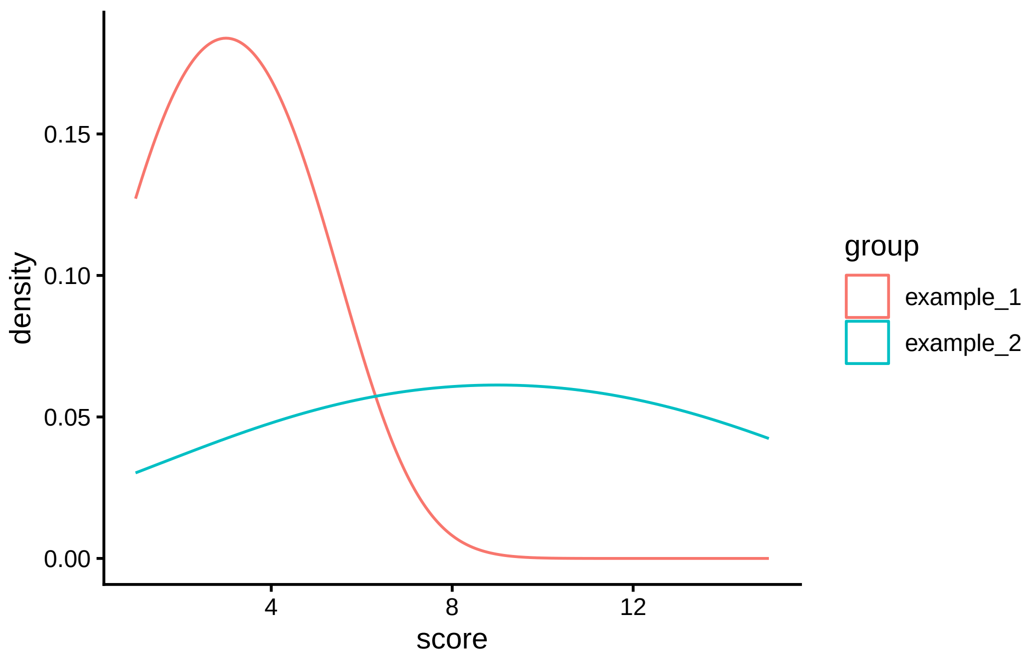
```
[1]  3  6  9 12 15
```

Let's see if the sample standard deviation has changed.

```r
s <- sd(example_2)
s
```

```
[1] 4.743416
```

There distributions now look very different. The distance between scores in the `example_2` object has changed by a multiple of 3. The distance between scores is now greater, resulting in a wider distribution.



# 8 Selecting participants

## 8.1 Sampling

Difficult to study everyone (population)

*Ex. Every single person with PTSD*

**Probability Samples**

Can quantify the likelihood of being selected because we know how many people are in the population of interest

Know selection probability

Accurately describes a population

Must be representative of the population

Rarely used in behavioral sciences

*Ex. A 1 in a million chance of winning the lottery*

**Non-Probability**

Can't quantify the likelihood of being selected, probably because we don't know how many people are in the population being measured.

## 8.2 Error of Estimation

Samples rarely mirror the population perfectly

The difference between them is called sampling error

Sampling error can be estimated for probability samples because we know the population size

## 8.3 Probability sampling

## 8.4 Types of Probability Sampling

**Simple Random Sampling**

Known Population

Random Selection

Equal selection probability for every sample

*Ex. Pulling a name out of a hat*

Require a **sampling frame**

- An outline of the people you will be sampling from

*ex. USI students enrolled in Introduction to Psychology this year*

**Systematic Sampling**

Not concerned with population size

Every $n$th person gets chosen

It is not random, a system is used to choose participants

*Ex. The every third person who enters the classroom gets to participate in an experiment*

**Stratified Random Sampling**

Divide population into groups based on a shared characteristic (called **strata**)

Randomly sample people from each strata

This sampling method ensures an adequate number of participants from each group

*Ex. Separate participants into groups based on major, then take 5 random participants from those strata*

**Clustered Sampling**

Clusters occur naturally

Sample the clusters, then sample participants

More efficient than stratified random sampling because only the clusters are sampled

Clusters are essentially the same as one another, whereas strata are fundamentally different

*Ex. Randomly sample 5 of the counties in Illinois, then randomly sample the participants in those counties.*

## 8.5 Issues with Probability Sampling

**Non-response**

Some participants do not respond, these participants may be different in some important way from those that do respond

**Misgeneralization**

Generalizing to the wrong population

## 8.6 Non-probability Samples

Can't calculate selection probability

Not random

Used to study relationships among variables

Behavioral sciences use non-probability samples most often

### 8.6.1 Types of Non-probability Samples

**Convenience Sampling**

Uses participants that are easy to obtain

Most typical type of non-probability sampling

Replication of experiments shows generalizability

**Quota Sampling**

Specific proportions of people with selected characteristics are selected

Often used for market research

*Ex: 20 USI students who enjoy hiking*

**Snowball Sampling**

Used for hard to reach groups

Can use incentives (cautiously) to increase response

*Ex: For every person you bring in to the experiment I will give you 5 dollars*

## 8.7 How Many Participants is Enough?

Economic samples are typical, only collect as much data as needed.

Determine how many are needed via a power analysis.

Aim for reasonable accuracy and cost for the scope of the project.

# 9 Ethical Issues in Behavioral Research

## 9.1 Researcher Obligations

Enhance Understanding

Protect participants

Ethical questions arise when enhancing understanding and protecting participants conflict

## 9.2 Some approaches to ethical decisions

Deontology

Ethical decisions are made based on a moral code

Utilitarianism

Ethical decisions are made based on weighing the benefits and consequences

Ethical Skepticism

Belief that a concrete moral code cannot exist

### 9.2.1 Benefits of Utilitarianism

Basic knowledge

Improved techniques

Practical outcomes

Benefits for researchers

Benefits for participants

### 9.2.2 Costs of Utilitarianism

Time and effort

Participant's welfare

Money

Deception; creating distrust

## 9.3 Institutional Review Board (IRB)

Scientific and nonscientific board members

Researchers describes purpose, procedures, and risks

IRB must approve study before it can be conducted

## 9.4 Lack of adequate informed consent

Informed consent is a disclosing of the nature of participation

A researcher must obtain explicit agreement from participants

Possible problems:

Compromise study validity

Some participants are unable to consent

## 9.5 Ethical Considerations

### 9.5.1 Invasion of privacy

Participants may decide when, where, to whom, and what responses to reveal

Public observation is not an invasion of privacy

### 9.5.2 Coercion

Pressure to participate from an authority figure

To prevent coercion, alternate activity must be available to opt of out research

### 9.5.3 Potential Harm

Pain, stress, failure, anxiety, or other negative emotions

Minimal risk – no greater than that ordinarily encountered

More than minimal risk requires strong justification

### 9.5.4 Deception

False purpose of study

Experimental Confederate

False feedback

Presenting related studies as unrelated

Giving incorrect information regarding stimulus materials

### 9.5.5 Violation of Confidentiality

Data may only be used for research

Data may not be disclosed to others

Anonymity is the easiest way to ensure confidentiality

### 9.5.6 Debriefing

Clarify nature of study

Remove any stress or negative study-indices consequences

Obtain participant reactions

Ensure participants leave feeling good about participation

### 9.5.7 Vulnerable Populations

Children

Prisoners

People with impaired decision making

People at risk for suicide

Pregnant women, fetuses, newborns

### 9.5.8 Scientific Misconduct

Fabrication, falsification, plagiarism

Questionable research practices

Unethical behavior

# 10 z scores

## 10.1 z-Scores and Location

Raw scores provide very little information by themselves

Ex. Terrence got 34 points on the test. Did he do well?

The mean and standard deviation provide a context with which to interpret the raw score.

Ex. Archibald's class average was 75 points with a 2 point standard deviation. His score of 34 is not very good.

z-scores tell us where an x value is in relation to the mean and standard deviation with one number.

## 10.2 Z-scores

Signed (+ or -) number

The sign tells you whether the x value is above the mean (positive) or below the mean (negative)

The number tells you the amount of standard deviations between the raw score and the mean of the distribution

Ex: What does a z-score of ____________ mean: -1? +2? -3?

## 10.3 Z-score Formula

## 10.4 Z-scores as a Standard Distribution

z-scores do not change the shape of the original distribution or the location of a score relative to others

When x values are transformed into z-scores, the resulting distribution:

The mean is always zero

The standard deviation is always one

Most z-scores are between z = -2.00 and z = +2.00

## 10.5 Z-scores and Locations

As descriptive statistics, describe exactly where an individual is located

As inferential statistics, determine whether a sample is representative of its population

## 10.6 Z-scores and Samples

Z-scores are standardized, so we can compare different distributions.

# 11 Probability

## 11.1 Probability

The likelihood of all possible outcomes

"Probability" = p

Can use fractions, decimals, or percentages:

p = ½ = .50 = 50%

Goes from 0% - 100% or 0 – 1

Equals the desired outcome divided by all possible outcomes

## 11.2 Probability & Sampling

Samples are used in inferential statistics, to make inferences about larger populations

Probability can be used to quantify the relationship between samples and population

When something occurs in a sample, how likely is it that it represents the population?

Probability calculation requires independent random sampling, has an equal probability of being selected and replacement

## 11.3 Probability and Inferential Statistics

Low probability values = Special/Rare, not common or likely to happen

High probability values = Common/likely to happen

In terms of finding an effect:

Low probability = effect

Means your finding is unlikely to happen by chance, there is a different cause

High probability = no effect

Means your finding is common or likely to happen

## 11.4 Probability and Frequency Distributions

If a distribution displays a population of scores a portion of the graph represents a portion of the population

Probability can be defined by a proportion of the graph

Can determine this by using the z-score and the unit normal table

# 12 Distribution of Sample Means

## 12.1 Samples vs. Populations

Though most behavioral science uses sample to test hypotheses, often times those hypotheses are about populations.

A sample is not a perfect representation of a population, so any statistics you calculate for that sample are also not representative of the population.

Sampling Error

The difference between sample statistics and population parameters.

## 12.2 Sampling Distributions

Sampling Distribution

A distribution of statistics of all possible samples of a given size from a population

One example of a sampling distribution is…..

Distribution of Sample Means

A collection of sample means for all possible random samples of a given size that could be obtained from a population.

A distribution of sample means should form a normal distribution, with most of the sample means grouping around the population mean if the number of scores in each sample is more than 30.

Larger sample are more representative of populations than smaller samples.

The mean of the distribution of sample means is the same as the population mean.

## 12.3 Standard Error of the Mean

The standard deviation of the distribution of sample means is the

Standard Error of the Mean

= √ M =

n

## 12.4 Z-score for a Sample

By using a sampling distribution you can calculate the location of an entire sample within a population

z = M - µ

M

# 13 Hypothesis Testing

## 13.1 Purpose of Hypothesis Testing

Behavioral scientists often can't measure all individuals in a population.

*Ex. Measuring the inhibition of all lawyers in the United States*

Use samples to test a hypothesis that is made about the population.

Expose a sample to your IV, evaluate the results, and make an inference that the same effect would be seen in the population if you could actually measure everyone.

Hypothesis testing takes several steps

## 13.2 Steps of Hypothesis Testing

1. State the Hypothesis
   - **Null Hypothesis**
     - H0
     - The treatment had no effect on the DV
   - **Alternative Hypothesis**
     - H1
     - The treatment does have an effect on the DV
2. Set the decision criteria
   - **Alpha ( )**
   - The alpha value tells us how willing we are to make a mistake (as there is never a flawless study) and what probability of error we are willing to accept.
   - In social sciences less than 5% probability (p) is acceptable
   - = .05 = 5% = 5/100
   - Probability of error (p) should be $< .05$
3. Collect data and compute statistic
   - T test, Correlation, ANOVA

4. Make a decision

    - Can reject the null hypothesis or fail to reject the null hypothesis
    - **Reject the Null hypothesis**
        - Claims there no significant effect
    - **Fail to Reject the Null Hypothesis**
        - Claims there is a significant effect
    - There is a 5% chance of making a mistake (false positive or false negative)

## 13.3 Decision Making Errors

**Type 1 Error (False Positive)**

Rejecting a true Null hypothesis, claiming there is a significant effect when there really is not.

*Ex. Claiming that a medical treatment will cure cancer, but it does not.*

**Type II Error (False Negative)**

Failing to reject a false Null Hypothesis, claiming there is not a significant effect when there really is.

*Ex. Claiming there is no difference between smokers and non-smokers, but there really is.*

## 13.4 Hypothesis Testing Table

Use this table to help you determine whether the correct decision has been made about the Null and Alternative Hypotheses.

## 13.5 Assumptions of Hypothesis Testing

The variability of scores and the number of scores in a sample influence the results of a hypothesis test, so several assumptions must be met before conducting one.

1. Must have random sampling
2. Must have independent observations
3. The value of sigma must remain unchanged by the treatment
4. The data must form a normal sampling distribution

# 14 T-tests

## 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

## 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.

## 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.

### 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.

### 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.

### 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

### 14.3.4 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.

```
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)
)
```

## 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

### 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.

```
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.

### 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

### 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.

## 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.

### 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.

### 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.

### 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.***

## 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

### 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.

## 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

### 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)
)
```

### 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.

### 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
```

### 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.

## 14.8 APA Tables

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

### 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.

```
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.

## 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

```
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])

# References

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