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 Setup

R is used in this course for several reasons.

- It is free, so people create things for it that we can use for free.
- Students can have it on any and all personal devices.
- It is widely used in research, industry, and elsewhere.
- It has great visualization tools.
- The understanding of data and how it is manipulated is fundamental to understanding statistics.

Now that we are convinced, lets get R on all of our personal devices.

When we say R, we are often referring to both R and R Studio. R studio is a program that makes R much easier to use, and that's usually what we're referring to...but we need R for R Studio to work.

1.1 Download R

R typically has a major update once a year and several smaller updates throughout the year. When these come out you have to download all of your packages again, but it's good to keep it updated. Link to download R.

1.2 Download R Studio

We will do all of our work in R Studio, download it here, after base R has been downloaded. It also has several updates a year, but it doesn't change anything about the packages so keep it updated.

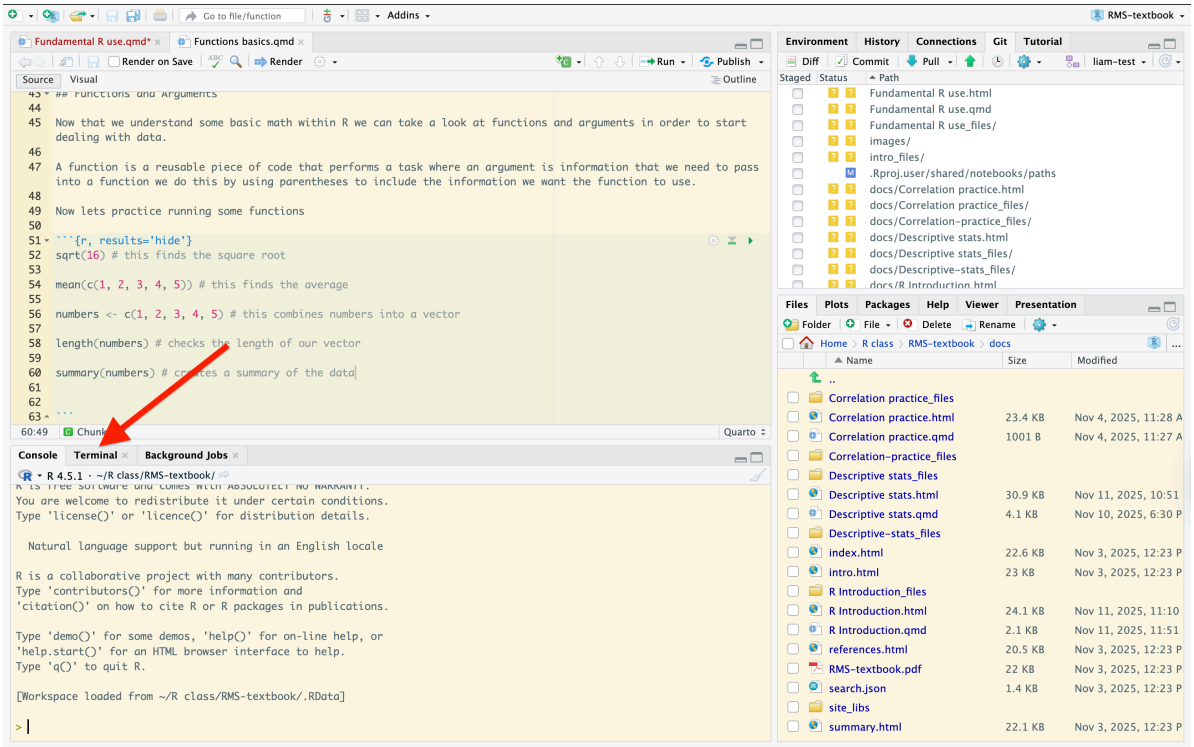
1.3 R Studio Overview

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.

1.3.1 Console Frame

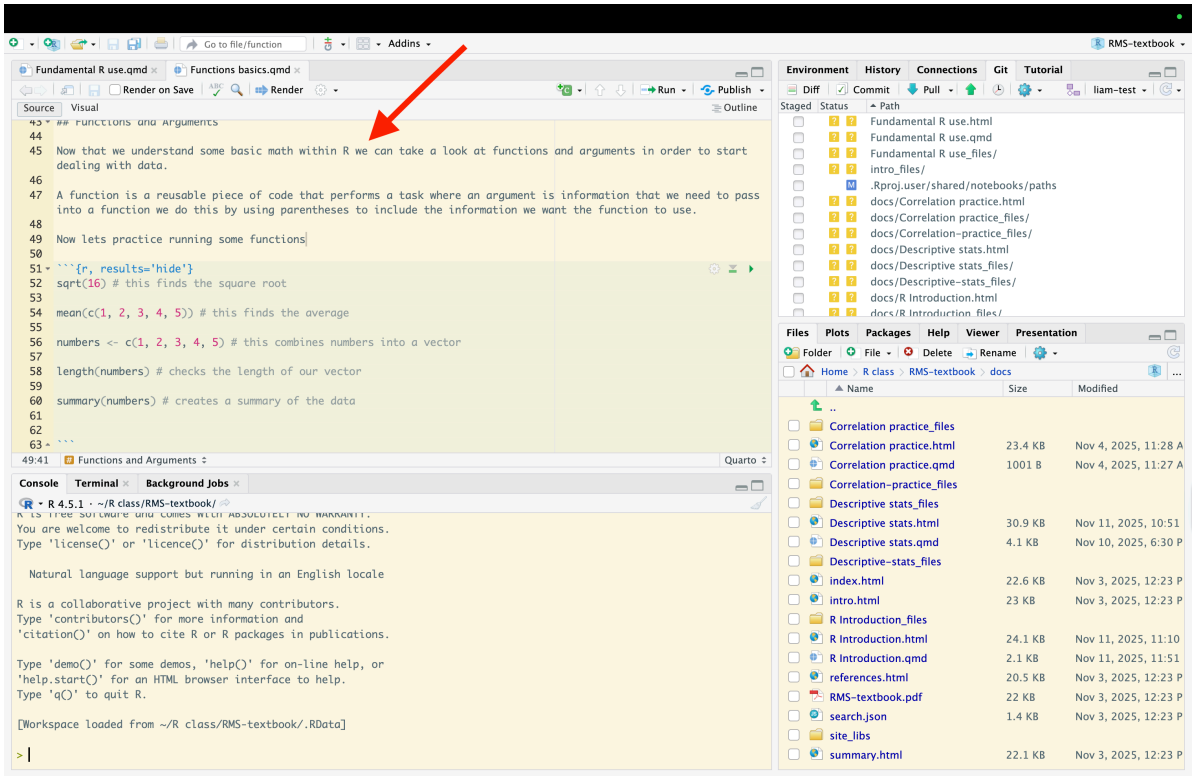
The console frame is the **bottom left frame on your screen**. The console is like a scrap piece of paper or and etch an sketch, its a great place to do some troubleshooting or viewing data through the `glimpse()` function but **any work you do here will not be saved.** To run your code in the console frame you type the code and press the `enter` or `return` key. You can also see output from your code directly in this frame. T

he 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.



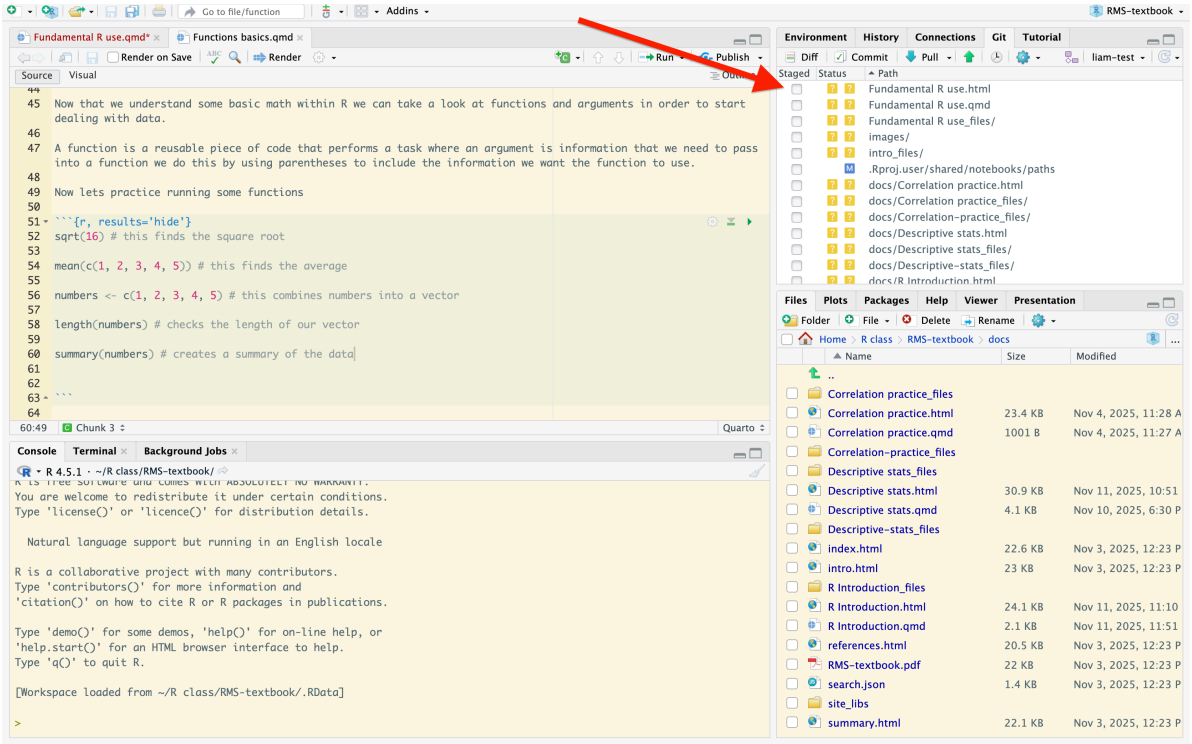
1.3.2 Source Frame

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. The source frame is used to open RMarkdown documents, scripts, and other ways to type code. You can also save and open new documents and files at the farthest top left corner of this frame.



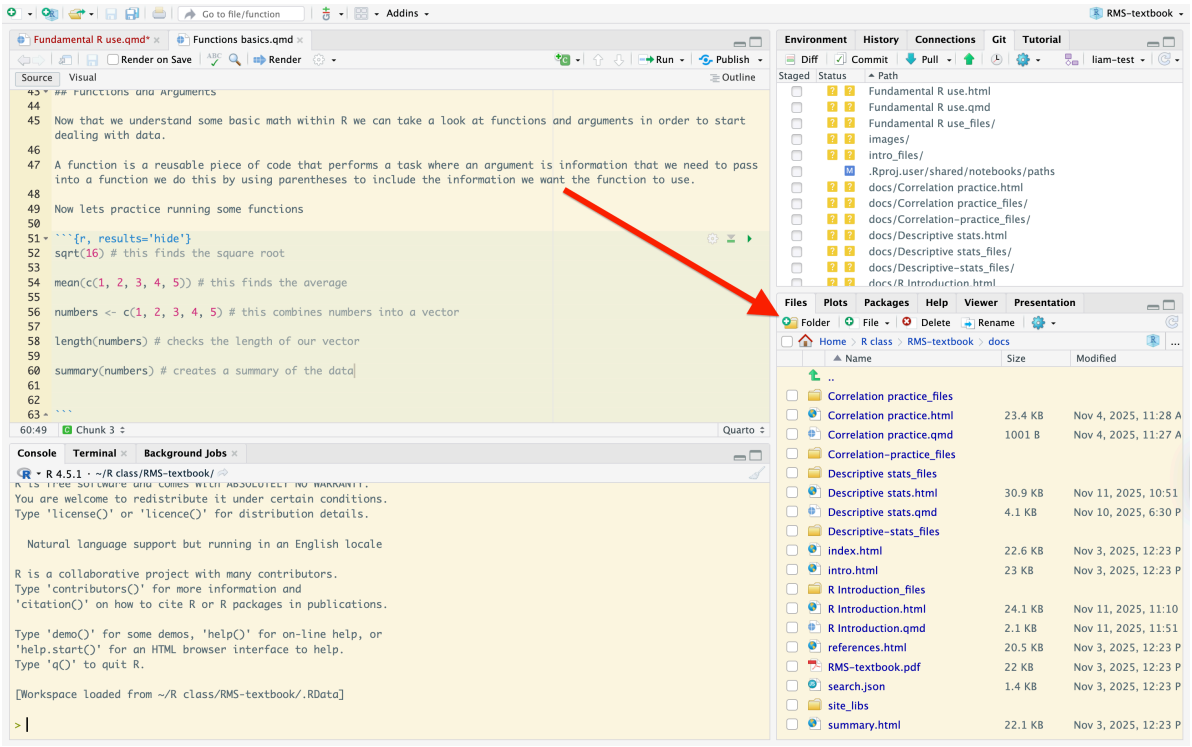
1.3.3 Environment/History

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.



1.3.4 Files/Packages

The last frame on **the bottom right is your files and packages** frame. This frame is more of a window into your computer in the sense that you are able to view the packages and files that you have on stored on it. You can also use it to check any plots you might have and to read any documents on your computer as well. You only need to download a package one time, then it is saved. However, every time you open Rstudio packages must be loaded into the R environment.



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.

1.4 Saving

Science should be reproducible, so should your work. When you close R and start a new session you will be starting from a blank slate unless you save your work.

By default, R will save your work in a **working directory**, The **working directory** is a location on your computer or the cloud. See where your things are being saved by typing `getwd()` in **the source pane** now. This is probably not a good place to save the *many* files we will be creating in this course. Instead, we will create an **R Studio Project**.

1.4.1 R Studio Project

Follow the steps below to create an R Studio project where all of your things will be saved.

- Above the console pane you will see several clickable buttons with green pluses on them. Click the second button from the left to create a new project.

- Click “New Directory”
- Click “New Project”
- Name your project something like “rms” or “psy303”
- Choose a sensible subdirectory by clicking the “Browse” button so you can find all of your files. I suggest your desktop on your device, not the cloud.
- Click “Create Project”
    - Now, you should notice in the top right of your screen the name of your project, indicating that you are working within that project.

Check your new working directory by typing `getwd()` in the source pane again, it should reflect your subdirectory and project name.

As long as this project is open everything you save, which we will get to in the sections below, will be placed in that working directory and you will have easy access to anything in that directory from R. We will be creating code documents in this class from a package called RMarkdown. But to get there we need to understand what packages are.

1.5 Packages

Now that you are familiar with the R interface we will get into the things that make R run, packages.

1.5.1 What is a package?

A package is where R gets its main power. Packages are compilations of data, functions, etc. that we must download from CRAN - the Comprehensive R Archive Network. Packages can contain lots of different things. They are most useful because they allow us to perform actions we can do with base R alone- which is just R without any packages downloaded.

1.5.2 Installing and finding packages

There are two ways to install packages. Install both of these packages because we will be using them a lot.

1. Via your files and packages pane.
    - Click the packages tab in the Files/Packages pane
    - Click the install button on the top left of that pane

- type “rmarkdown”
- click install

2. Via code.
    - click to the right of the `>` symbol in your bottom left console pane
    - use the `install.packages()` function to install the desired package
    - 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")
```

1.5.3 R Markdown

Now that we have R Markdown downloaded we are going to open a markdown document.

1. Look to the upper left of the console pane, you should see a white page with a green plus arrow over top of it. Click it.
2. In the dropdown menu click “R Markdown”.
3. You will be prompted to name your document - usually you want this to be obvious and helpful. For now, call it “fundamentals”

Your document should open and it will be automatically filled with some stuff. Do not be alarmed, you can do many things in R Markdown, but we will start out slow.

1.5.3.1 Source/Visual Editor

Above the document itself, just under the save icon, you should see the words “Source” and “Visual”. These are buttons that change your view from the source code view and a rendered view that works more like Microsoft Word.

Click back and forth.You should notice some differences. What is most important for us is the use of `##`. These tell Markdown to make a header. It’s super useful when organizing your code.

We will be staying in the Source editor.

1.5.3.2 YAML Header

At the very top of the document you see a YAML header, this formats your document. For now, we will leave it be.

—

title: “Untitled”

author: “zp”

date: “2026-01-08”

output: html_document

—

1.5.3.3 Code Chunks

Immediately below the header is the first code chunk. Code chunks are placed within a Markdown document where code can be executed. Text outside of a code chunk is interpreted just like a word processor - so code doesn’t work.

Code chunks always begin with “‘{r}

and end with “‘

You can name code chunks for organization, the first chunk in this markdown document is named “setup”. The next thing, “include = false” tells markdown to do something specific with that code chunk. In this case, it’s saying run this code in the background, but don’t show the viewer.

Code chunks can be customized in various ways. For example, in this textbook I make lots of code visible, but I choose to hide the output.

In this class you should never have one incredibly long code chunk. They should be used to complete one specific task that is explained via the word processing and then moving on to the next task and repeating the process.

To create a code chunk you can click the green C with button to the top right of the Source frame or you can use the keyboard shortcut `option` + `command` + `i`.

When code is typed into a code chunk it doesn’t run when you press `enter` like in the console. In RMarkdown you are writing a kind of script - code that is meant to be executed all at once at the end. However, that’s not how people work. Often, you will need to run the current chunk you are working on. When working in Markdown the code chunks you create have acccess to any object you have saved in the environment and any package you have loaded that session. There are two ways run code in R markdown.

1. Press the little green play button in the top right corner of the code chunk.
2. Click anywhere in the code you want to run and use the keyboard shortcut `shift` + `command` + `return`

1.5.3.4 Text

Just below the code chunk we see ## R Markdown. This tells markdown to make a second level header named R markdown. We can see this if we switch to the Visual editor.

Just below that is normal text.

That's all we need to understand for now. When you open these documents you can delete everything but the YAML header. So let's do that now and use the Markdown file you created for the remainder of this chapter and the fundamentals chapter.

1.5.4 Loading packages

When you download a package it is on your computer until you delete it or until you update R. However, when you start a new session you will not have access to that function, the package will need to be loaded. In order to use the functions within a package you must load it into the R environment every single time. For us, that means the first code chunk of our R Markdown files is just loading in the packages we will need for that document.

Packages are loaded into the environment by using the the `library()` function. It will not work if you do not have the package installed.

```
library(tidyverse)
```

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

Sometimes you will need to update your packages. You can do this by executing the code below in your console or within a markdown document or you can click the green update button in the packages pane.

```
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()
```

Part I

Introduction to R

This section will get you acquainted with the basics of how to use R.

2 Fundamentals of R

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 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 Objects

In R, we will primarily be working with object. An object is a container that you label. It can bea single number, a giant dataset, a graph, a function or even models that we need. In order to create an object we use the `assignment operator`, it looks like an arrow pointing to the left `<-`. The `assignment operator` creates an object with the name of whatever is to the left of the operator and it will contain whatever is to the right of the operator.

```
x <- 67 # object named x
y <- "Good Morning" # object named y
```

If i make changes to an object that I want to keep, I must rewrite the original object or create a new one. If not, then my changes are not saved.

```
x # open the x object
```

```
[1] 67
```

```
x + 3 # add three to get 70
```

```
[1] 70
```

```
x # x is still 67
```

```
[1] 67
```

```
x <-  x + 3 # make a new object named x using the old x object plus 3

x # the x object is 70 now
```

```
[1] 70
```

2.2.1 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()
```

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

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

2.3.1 Applied Functions

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
# Create object x that is a list of numbers
x <- (c(1, 2, 3, 4, 5))
```

2.3.2 Arguments

Functions require certain information to carry out their actions. Arguments are where we provide that information. Many functions have default values automatically used for arguments. However, we often have to supply information for arguments, even if it's just the name of the object we want to use the function on. Arguments are separated by a comma. Let's look at the `sum` function. It has two arguments, the first is the data itself or the vector object containing the data and the second is `na.rm =` which has a default value of `FALSE`. `na.rm` is a logical argument asking if we want to remove data coded as NA, which could be missing data or incompatible data. Default arguments don't need to be typed out, they are implied, unless we want to give a value to the argment other than the default.

```r
x <- c( 1, 2, 3, 4, 5)
sum(x) # no need to specify na.rm here but what when we give it an NA value

y <- c(1, 2, 3, 4, 5, NA)
sum(y) # R can't calculate the sum if it doesn't know the value of NA
sum(y, na.rm = TRUE) # fixed
```

2.3.3 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
```

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 Creating Data

2.4.1 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. We will go over the different types of data later in the chapter. Here are some simple basic examples of vectors.

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

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

```
sum(numbers)
```

```
[1] 28
```

2.4.2 Data Frames

A **data frame** 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 object. The `data.frame()` function creates this framework for us. We create columns and observations in this format:

```
data.frame(
  column_1 = c(obs_1, obs_2, obs_3),
  column_2 = c(obs_1, obs_2, obs_3)
)
```

Let's put some data in there.

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

Notice we have three columns and two observations. All information in a row is part of one observation. So, James is 21 and his sport is Baseball.

2.4.3 Tibbles

A tribble is short for transposed tibble and it is a different way to create small data frames that looks more like a spreadsheet. It requires the `tibble` package to work, but that's part of the tidyverse so we should be ok. Some functions will only take data formatted as tibbles.

```
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 becor
```

```
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 tibble fully completed with both rows and columns where you can see in the bottom of the output that the tibble 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.

2.4.4 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.

2.5 Types of data

There are different types of data. In statistics we use numbers to represent all kinds of different things. We must differentiate between when a number 3 is representing a third category or the result of 1 + 2.

2.5.1 Numeric and Double data

Numeric data is the type of data that we see that represents integers and "double precision floating point" numbers….or decimals. These are pretty common and are just regular numbers as you are familiar with them. They exist on a number line, can be multiplied and divided, and so on. Data formatted as numeric or double are on interval or ratio scales of measurement.

We can see if the data we are looking at is numeric/double by looking for how it is abbreviated. You can use the `typeof()` function in R to see exactly what type of data you are dealing with when you have a question about single objects, like below.

```
typeof(42)
```

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

```
typeof(FALSE)
```

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

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

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

If we use the `glimipse` function, which tells us the format of an object and the first few observations in the object, we can see how we will more commonly interact with data formats.

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

```
 num [1:5] 1 2 3 4 5
```

2.5.2 Character data

Character data is words that are not functions. Character data may also be referred to as "string" data. As behavioral scientists we may ask participants to write a response, this would be stored as character data. 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`.

```
my_vector <- c("apple", "banana", "cherry")
typeof("apple")
glimpse(my_vector)
```

2.5.3 Logical data

Logical data can only represent one of two values, `TRUE` or `FALSE`. Logical data will be represented in R 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
typeof(bool1)

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

2.5.4 Factor data

Data are formatted as factors when we are referring to categories. 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 have. Data formatted as a factor can be on an ordinal scale or a nominal scale. 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.

```r
college <- c("freshman", "sophomore", "junior", "senior")
glimpse(college) # notice that R formats this as a character vector

college_fct <- factor(college)
glimpse(college_fct) # now it is a factor, but the order of the levels is wrong

college_fct <- factor(college, levels = c("freshman", "sophomore", "junior", "senior"))
glimpse(college_fct) # now the order is right

# factors can also be numbers

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

2.6 The pipe

Now we will be working with the pipe which is symbolized by `%>%` or `|>`. Whenever we are running multiple functions in a code chunk we use the pipe to channel the output right into the first argument of the next function without having to do the work ourselves. The pipe is can be easily created within a code chunk by using 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.

2.7 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.

2.7.1 ? 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
```

2.7.2 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")
```

2.7.3 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.

2.8 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 Data Wrangling

3.1 Introduction

The tidyverse is filled with functions that help us work with data. Most of the functions covered here come from the dplyr package.

Datasets in R have rows and columns. Rows are the participants, cases, or observations andcolumns are the variables. Dplyr has certain functions, sometimes referred to as verbs, to handle this information.

In the grammar of dplyr, the first argument is always your data frame object. The following arguments describe what to do with that data frame. The result of a dplyr function is always a new data frame.

There are two verbs used on columns: `select()` and `mutate()`

There are two verbs used for rows: `filter()` and `arrange()`

There are two verbs used for summary: `group_by()` and `summarise()`

3.1.1 Logical Operators

We can use logical operators within dplyr to help manipulate our data.

Below is a list of logical operators.

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	grater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x and y
isTRUE(x)	test if x is TRUE

3.1.2 Pipes

Pipes are super useful, they may look like this %>% or this |> . Both are fine.

I recommend using the keyboard shortcut for pipes because they're annoying to type. Shortcuts can be found in the Help menu at the top of the screen.

The pipe passes the object on its left to the first argument of the function to the right of the pipe. In dplyr, the first argument of anything is a data frame, so it's useful to know. It helps me to think of the pipe as saying "and then". So, in my head I read the code below as "load the package tidyverse, then load the diamonds data frame object and then count the rows in that object."

```
library(tidyverse)

diamonds |>
  count()
```

3.2 Functions used on columns

3.2.1 select()

The `select()` function chooses which columns we want to keep from a larger data frame.

```
# load psych package that contains sat.act data
library(psych)

# check out all the variables in the data frame
glimpse(sat.act)

# choose only gender and ACT
sat.act |>
  select(gender, ACT)

# choose everything but ACT
sat.act |>
  select(!ACT)

# choose everything from gender to age
sat.act |>
  select(gender:age)
```

```r
# choose everything that starts with SA
sat.act |>
  select(starts_with("SA"))

# choose everything that ends with T
sat.act |>
  select(ends_with("T"))

# choose everything that has a t in it
sat.act |>
  select(contains("T"))
```

3.2.2 rename()

[will do this later]

3.2.3 relocate()

[will do this later]

3.2.4 mutate()

`mutate()` is used to change existing columns or create new columns. Mutate is very useful when making subscales, calculating totals, etc.

```r
# look at the data

glimpse(USArrests)

# create a new variable for all violent crime
# by default the new column as added to the end of the data frame
USArrests |>
  mutate(violent_crime = (Murder + Assault + Rape))
```

3.3 Functions used on rows

3.3.1 case_when()

[will do this later]

3.3.2 case_ match()

[will do this later]

3.3.3 if_else()

3.3.4 filter()

`filter()` is used at the ROW level, so it will not change the columns we see. `filter()` tells us which kind of participant data to include and exclude. This is especially useful if we want to exclude certain observations from our analyses (like outliers) or only look at as a subset of our data.

Let's stick with the arrests data and only keep the data from states that are more than 50% urban population.

```
# 42 states with over 50% urban population
USArrests |>
  filter(UrbanPop >50)
```

Let's do a little more with the diamonds data set.

```
# only diamonds over 1 carat in size
# still 17k
diamonds |>
  filter(carat > 1)

# typing ?diamonds allows me to see what is the best diamond qualities
# I only want THE BEST
diamonds |>
  filter(carat >1 & cut == "Ideal" & color == "D" & clarity == "IF")

# I'll compromise a little - by using %in% I get all very good and ideal cuts
diamonds |>
  filter(carat >1 & (cut %in% c("Ideal", "Very Good")) & color == "D" & clarity == "IF")
```

3.3.5 arrange()

This one is pretty straightforward, it arranges the order of data in a column based on the values in the rows.

```r
# arrange the data in descend order by carat size
diamonds |>
  arrange(desc(carat))

# I can resolve ties by including a second statement, 3.01 has lots of ties
diamonds |>
  filter(carat  < 3.02) |>
  arrange(desc(carat), desc(cut))
```

3.4 Functions for Summarizing

3.4.1 group_by()

`group_by()` creates a "grouped table where all operations you perform are now done by the group you indicate.

I wonder how many diamonds of each cut there are.

```r
diamonds |>
  group_by(cut) |>
  count()

# when count is used outside of a grouped table it just tells me the number of cases
diamonds |>
  count()
```

3.4.2 summarise()

`summarise()` is used in conjunction with group_by() and we can use it to compare groups - super handy for t tests and ANOVAs. It collapses each group into a single row and treats each row as a case. Notice the new dataframe produced after summarize, it is much smaller.

I've heard it's tough to have big diamonds that are of high quality. So let's find out.

```r
diamonds |>
  group_by(cut) |>
  summarise(avg_carat = mean(carat)) # create new variable to compare groups

# super useful for making graphs
diamonds |>
```

```
  group_by(cut) |>
  summarise(avg_carat = mean(carat)) |>
  ggplot(aes(x = cut, y = avg_carat)) + # ggplot only uses +
  geom_col()

# multiple calculations
diamonds |>
  group_by(cut) |>
  summarise(avg_carat = mean(carat),
            n = n()) # can do multiple summarise arguments
```

4 Data Visualization

Data visualization is a critical component of research and data science. It is used for many things like data exploration, testing assumptions, and communicating findings. There are many packages that visualize data in R (and base R), but we will focus on the most common `ggplot`

4.1 ggplot

`ggplot` is part of the tidyverse, it is widely used and highly customizable. However, it has it's own rules and conventions that must be learned before making any graphs.

4.1.1 syntax

First, load in the tidyverse package and create a data object to work with.

```
library(tidyverse)

dat <- tibble(score = rnorm(100, mean = 5, sd = 1))
```

When making a graph the first function will always be `ggplot()`. The first argument of ggplot, similar to the `dplyr` functions, is the data frame object we are working with. If we utilize the pipe we can write the code this way.

```
dat %>%
  ggplot(arg_1 = whatever,
         arg2 = whatever)
```

The second argument is `aes()` or aesthetics. This is where we assign values to various characteristics of the graph, like the x and y axes, fill, color, shape, linewidth, size, and linetype. Refer to the cheat sheet in the help menu for more information.

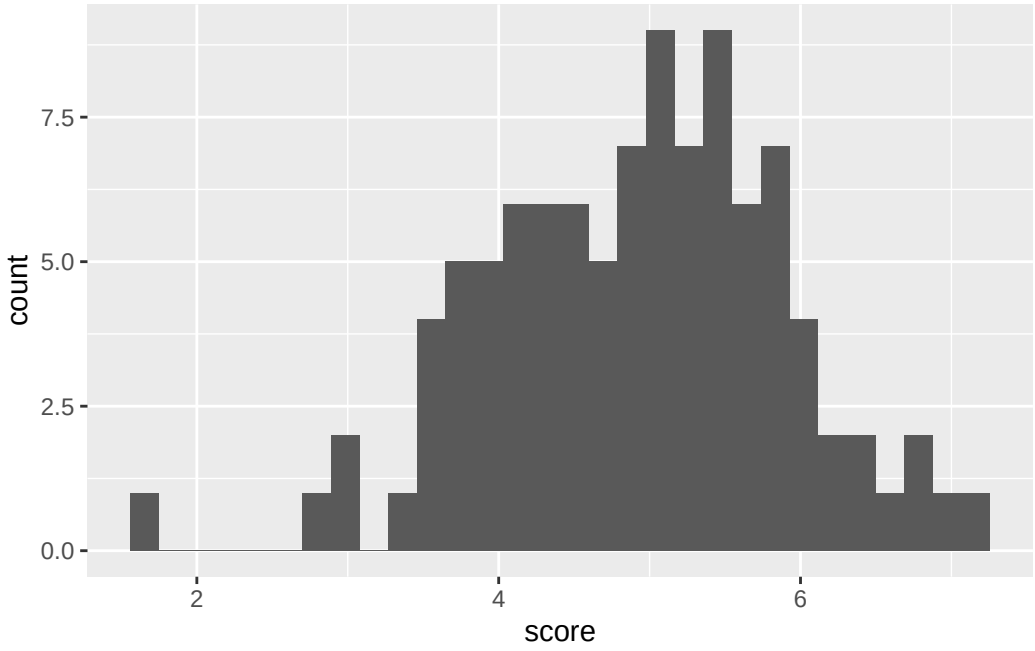
You do not have to choose values for all of these characteristics, only the ones necessary for the type of visualization you are creating. For example, when making a simple histogram we only need to assign a value to the x axis. So, the argument would look like this.

```
ggplot(dat, aes( x = score))
```

The part of ggplot most different from other code is how it handles moving from one function to the next. Normally, if we are working within the same data object we string together functions with the **pipe**. However, the pipe doesn't work within ggplot, instead we have to use the plus sign +.

After we set the aesthetic options we add our first layer to the visualization using a `geom` function. There are many geoms to work with, if you type geom R will suggest many for you. Let's make a histogram.

```
dat %>%
  ggplot(aes(x = score)) +
  geom_histogram()
```

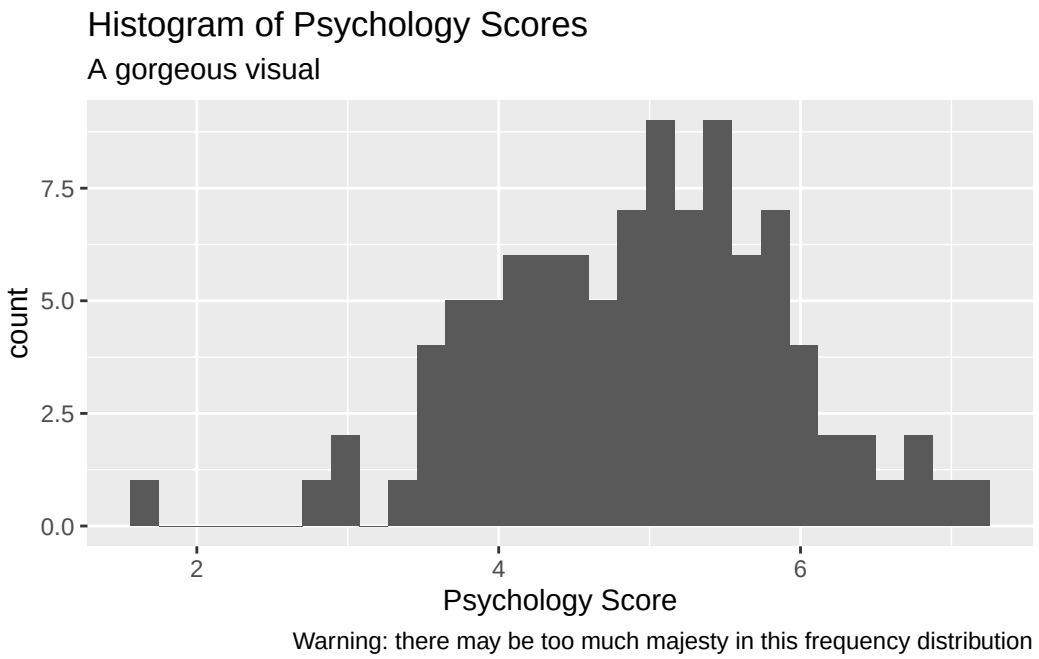


Gorgeous, let's customize it a little.

4.1.2 Title, subtitles, themes

A simple way to add titles and subtitles to your graph is with the `labs()` function. It is added to your code with another plus sign like below. All text strings must be in quotations or else R will interpret them as objects.

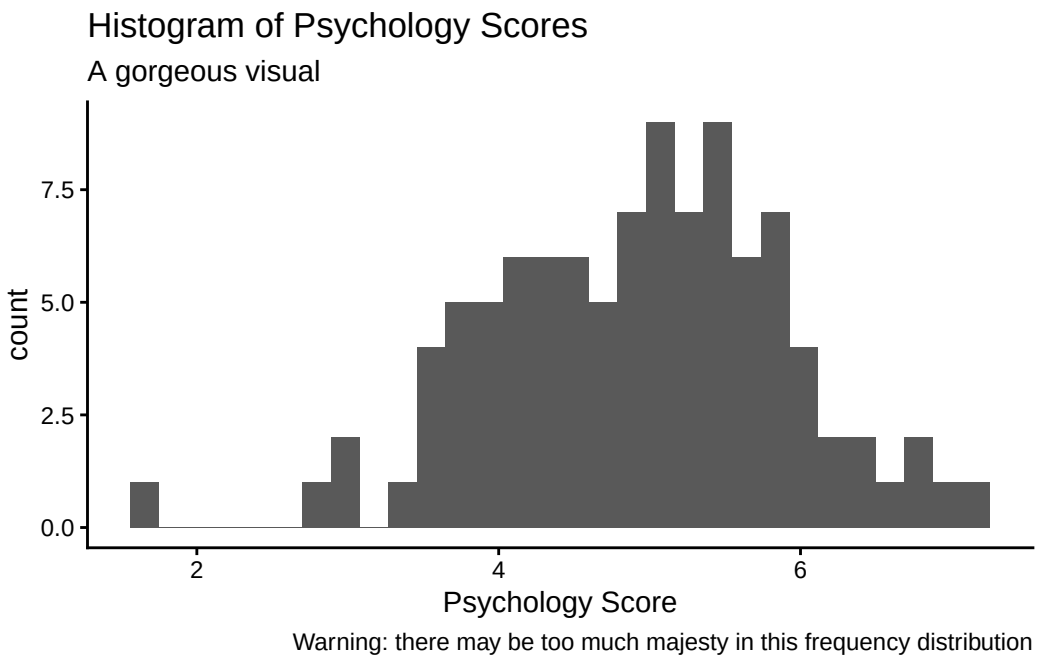
```
dat %>%
  ggplot(aes(x = score)) +
  geom_histogram() +
  labs(x = "Psychology Score",
       title = "Histogram of Psychology Scores",
       subtitle = "A gorgeous visual",
       caption = "Warning: there may be too much majesty in this frequency distribution")
```



As we can see, these graphs are highly customizeable, but the caption at the bottom is probably overkill.

Another easy way to improve these graphs is by using themes. by using the `theme()` function. Again, there are many options to choose from.

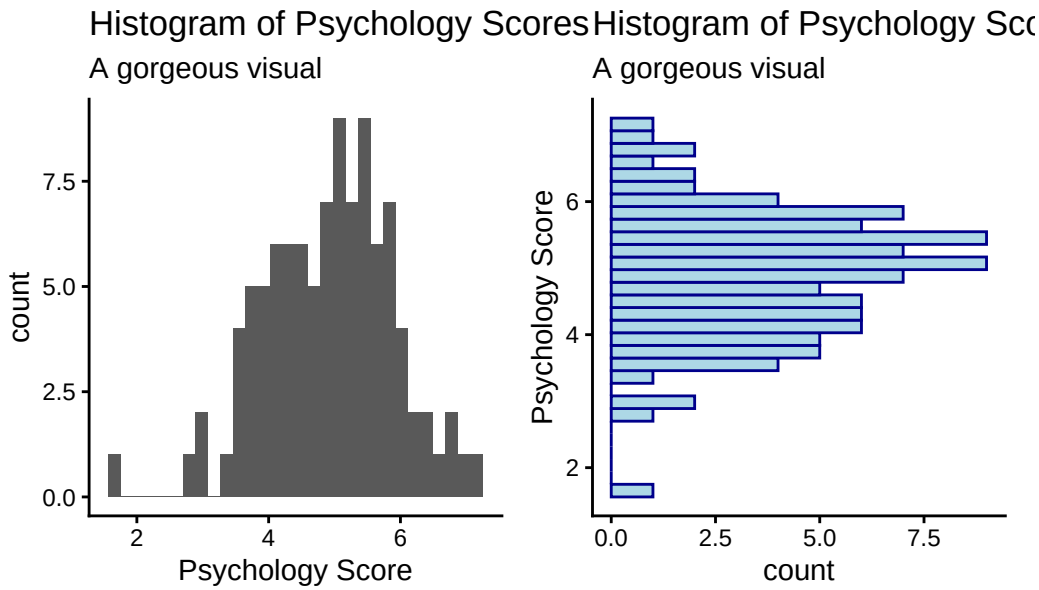
```
dat %>%
  ggplot(aes(x = score)) +
  geom_histogram() +
  labs(x = "Psychology Score",
       title = "Histogram of Psychology Scores",
       subtitle = "A gorgeous visual",
       caption = "Warning: there may be too much majesty in this frequency distribution") +
  theme_classic()
```



We can save this plot to an object using the assignment operator which allows us to manipulate it a little more. For example, with the help of the `patchwork` package we can easily manipulate the arrangement of multiple graphs.

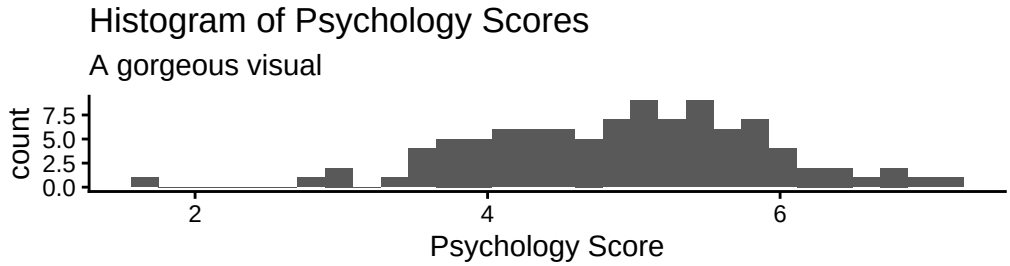
```
x <- dat %>%
 ggplot(aes(x = score)) +
 geom_histogram() +
 labs(x = "Psychology Score",
      title = "Histogram of Psychology Scores",
      subtitle = "A gorgeous visual",
      caption = "Warning: there may be too much majesty in this frequency distribution") +
 theme_classic()

y <- dat %>%
 ggplot(aes(y = score)) +
 geom_histogram(fill = "lightblue", color = "darkblue") +
 labs(y = "Psychology Score",
      title = "Histogram of Psychology Scores",
      subtitle = "A gorgeous visual",
      caption = "Warning: there may be too much majesty in this frequency distribution") +
 theme_classic()

library(patchwork)
```

```
x + y
```

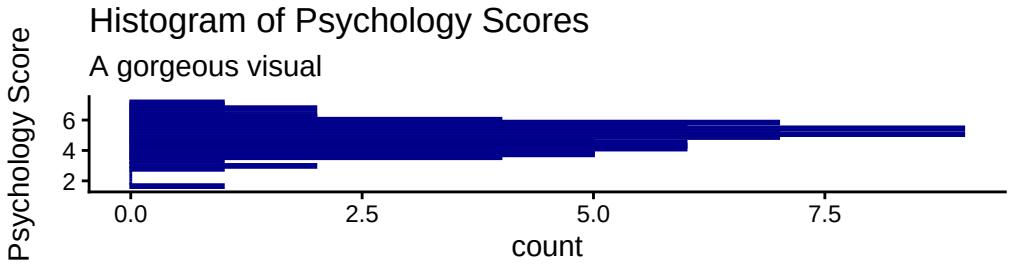


be too much majesty in this frequency distribution Warning: there may be too much majesty in this frequency distribution

```
x/y
```



Warning: there may be too much majesty in this frequency distribution



Warning: there may be too much majesty in this frequency distribution

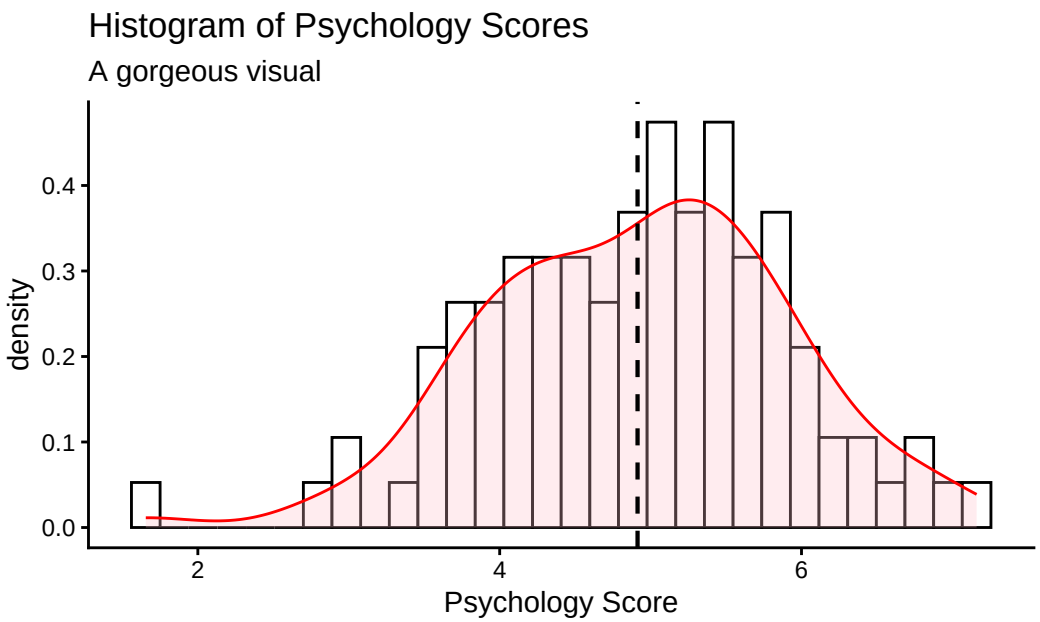
adding layers

Ggplot creates graphs by adding a series of layers on top of one another. So, we can add a density curve to this histogram to make it look cool. MAY AS WELL ADD A VERTICAL LINE FOR THE MEAN WHILE WE'RE T IT.

You'll notice I changed a lot of things, that's ok, play around of check out the ggplot cheat sheet for things to change. The `alpha` value changes the transparency of the layer. We had to lower it because the curve is placed on top of the histogram.

```
dat %>%
 ggplot(aes(x = score)) +
 geom_histogram(aes(y = after_stat(density)), fill = "white", color = "black") +
  geom_density(color = "red", fill = "pink", alpha = .3) +
  geom_vline(xintercept = mean(dat$score),
             color = "black",
             linetype = "dashed",
             linewidth = .75) +
 labs(x = "Psychology Score",
      title = "Histogram of Psychology Scores",
      subtitle = "A gorgeous visual",
      caption = "Warning: there may be too much majesty in this frequency distribution") +
 theme_classic()
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```


Histogram of Psychology Scores
A gorgeous visual
density
0.0
0.1
0.2
0.3
0.4
2
4
6
Psychology Score
Warning: there may be too much majesty in this frequency distribution

Part II

Research Methods

This section is about understanding research methodology and measurement.

5 What are Behavioral Sciences

5.1 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.

5.2 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.

5.3 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?

5.4 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

5.5 Purpose of Behavioral Research

Detect

Discover and document new phenomena.

Explain

Develop and evaluate theories that explain phenomena.

5.6 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.

5.7 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.

5.8 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.

5.9 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.

5.10 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.

6 How to plan an experiment

6.1 Experimental Research

Allows us to study causes of behavior.

6.2 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

6.3 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

6.4 Subject (Participant) Variables

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

*Ex. Ethnicity/Race, Hob*bies

6.5 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?

6.6 Dependent Variable

The variable a researcher measures.

Experiments must have AT LEAST one.

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

6.7 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.

6.8 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.

6.9 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

6.10 Power

The ability to detect IV effects.

Requires fewer participants.

6.11 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.

6.11.1 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).

6.12 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.

6.13 Sources of Error

Individual Differences

Transient states

Environmental factors

Differential treatment

Measurement error

6.14 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. If you are not confident in the outcome of your experiment what value does it have?

6.15 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

6.15.1 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

7 Measuring behavior

7.1 Types

- Observational
- Physiological
- Self-report

7.2 Scales of Measurement

7.2.1 Nominal Scales

Numbers are assigned as labels for characteristics or behaviors.

Provides the least amount of information.

Ex. Jersey Number

7.2.2 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

7.2.3 Interval Scale

Equal distance between the numbers reflect equal differences between participants

Does not have a "true" zero point

Ex. Temperature, IQ score

7.2.4 Ratio Scale

Has a "true" zero point.

Provides greatest amount of information.

Should be used when possible.

ex. duration, weight, accuracy

7.3 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.

7.3.1 The Mean

Most commonly used measure of central tendency.

Used in interval or ratio scales.

Is influenced by extreme scores (outliers)

7.3.1.1 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

7.3.1.2 Compute in R

You can calculate the mean in R with the function `mean()`

```
# create vector of scores
x <- c(4, 5, 3)

# use the mean function on the object
mean(x)
```

```
[1] 4
```

The mean is influenced by extreme scores. It gets pulled in the direction of the extreme score, making it less representative of the whole dataset and more reflective of the outlier (which isn't what we want).

```
# create vector of scores with an outlier
x <- c(4, 5, 3, 46)

# use the mean function on the object
mean(x)
```

```
[1] 14.5
```

This score is not very reflective of our dataset, so we should use an alternative measure of central tendency, like the median.

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.

7.3.1.3 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

7.3.2 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).

```
# create vector of data
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

# calculate median of the data object x
median(x)
```

```
[1] 5
```

Remember the datset with an outlier from before? How does the median handle the outlier?

```
# create vector of scores with an outlier
x <- c(4, 5, 3, 46)

# use the median function on the object
median(x)
```

```
[1] 4.5
```

That is a much more representative measure for the entire dataset.

Weekly high temperatures in Chicago last winter:

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

What was the median temperature in Chicago last winter?

7.3.3 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?

8 Measurement

8.1 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

8.2 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

8.3 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

8.4 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.

8.5 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

8.6 Increasing Reliability

Standardize how measure is administered

Clarify instructions and questions

Train researchers/coders

Minimize errors in coding data

8.7 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?

8.8 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

8.9 Bias

Test Bias

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

9 Variability

NEED RAFALIB PACKAGE

9.1 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.

9.2 Measures of Variability

9.2.1 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

9.2.1.1 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 range.

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

9.2.2 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

9.2.2.1 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).

9.2.2.2 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).

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

```
[1] 2
```

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

```
[1] 2.5
```

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

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

```
[1] 1.414214
```

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

```
[1] 1.581139
```

9.2.2.3 Calculate in R (short)

Start with the same data

```
# 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.

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

```
[1] 2
```

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

```
[1] 2.5
```

2. Calculate standard deviation

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

```
[1] 1.414214
```

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

```
[1] 1.581139
```

9.3 Population v Sample

Population is EVERYONE.

Variance (^{2}) = SS/N

Standard Deviation ( ) = √(SS/N)

Sample is a subset of everyone.

Variance (s^2) = 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, so statistics we run on them must be adjusted to account for this.

9.4 Biased and Unbiased Statistics

Biased Statistics

The average value calculated for a sample 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).

9.5 Transforming Data Sets

9.5.1 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_1`.

```
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.

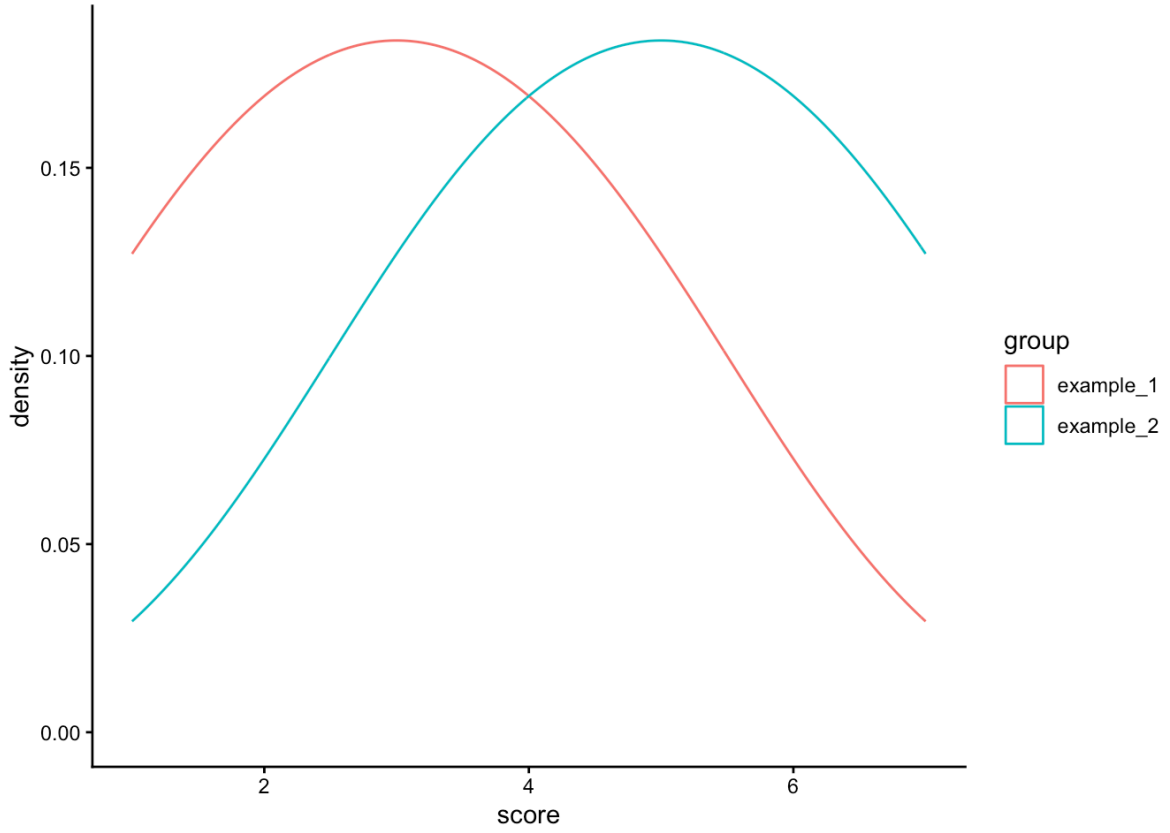
```
example_2 <- example_1 + 2
example_2
```

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

Let's see if the sample standard deviation 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.



9.5.2 Multiplying by a constant

Multiplying every score in the data set by a constant changes standard deviation. Try it out. Calculate the sample standard deviation of `example_1`.

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

s <- sd(example_1)
s
```

```
[1] 1.581139
```

Now, let's multiply every score by a constant of 3.

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

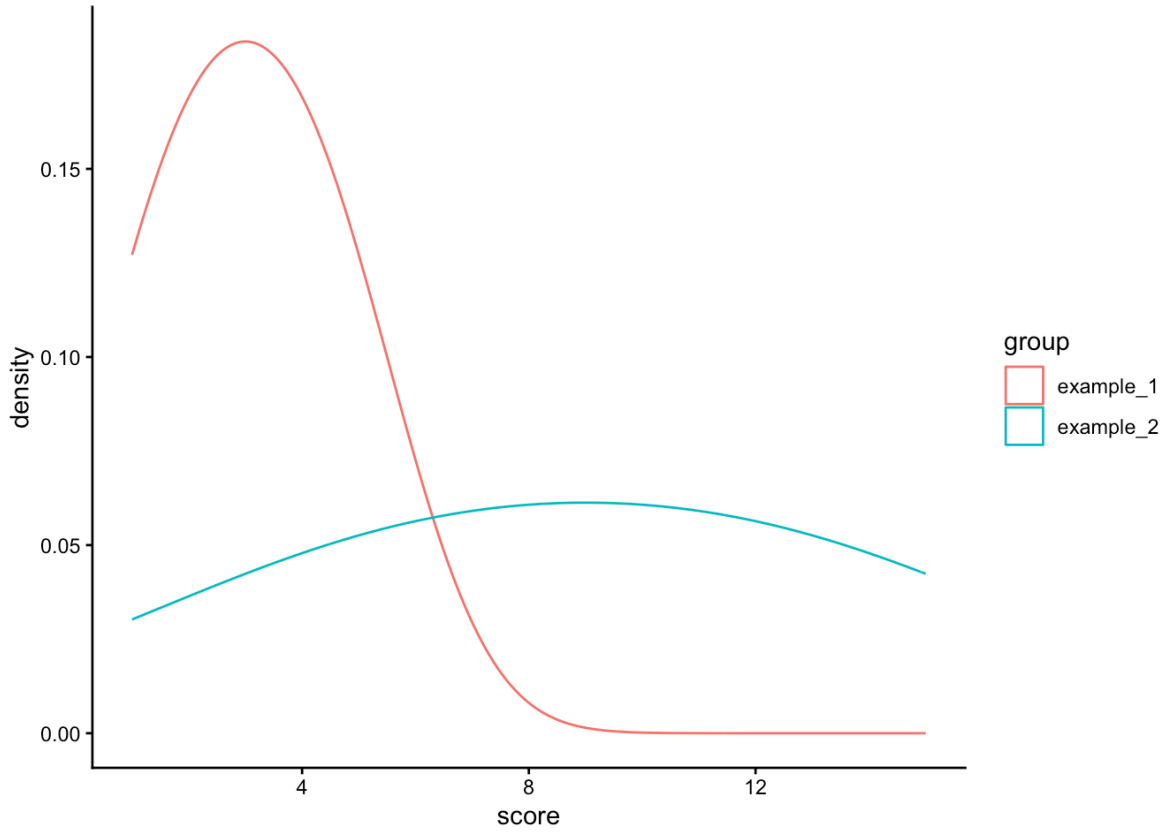
```
[1]  3  6  9 12 15
```

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

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

```
[1] 4.743416
```

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



10 Selecting participants

10.1 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.

10.2 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

10.3 Probability sampling

10.4 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.

10.5 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

10.6 Non-probability Samples

Can't calculate selection probability

Not random

Used to study relationships among variables

Behavioral sciences use non-probability samples most often

10.6.1 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

10.7 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.

11 Ethical Issues in Behavioral Research

11.1 Researcher Obligations

Enhance Understanding

Protect participants

Ethical questions arise when enhancing understanding and protecting participants conflict

11.2 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

11.2.1 Benefits of Utilitarianism

Basic knowledge

Improved techniques

Practical outcomes

Benefits for researchers

Benefits for participants

11.2.2 Costs of Utilitarianism

Time and effort

Participant's welfare

Money

Deception; creating distrust

11.3 Institutional Review Board (IRB)

Scientific and nonscientific board members

Researchers describes purpose, procedures, and risks

IRB must approve study before it can be conducted

11.4 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

11.5 Ethical Considerations

11.5.1 Invasion of privacy

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

Public observation is not an invasion of privacy

11.5.2 Coercion

Pressure to participate from an authority figure

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

11.5.3 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

11.5.4 Deception

False purpose of study

Experimental Confederate

False feedback

Presenting related studies as unrelated

Giving incorrect information regarding stimulus materials

11.5.5 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

11.5.6 Debriefing

Clarify nature of study

Remove any stress or negative study-indices consequences

Obtain participant reactions

Ensure participants leave feeling good about participation

11.5.7 Vulnerable Populations

Children

Prisoners

People with impaired decision making

People at risk for suicide

Pregnant women, fetuses, newborns

11.5.8 Scientific Misconduct

Fabrication, falsification, plagiarism

Questionable research practices

Unethical behavior

Part III

Statistics

This section includes statistics necessary for the behavioral sciences.

12 z scores

12.1 z-scores

Raw scores provide very little information by themselves

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

We have no idea - how many points were there? How did his peers perform?

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

Ex. Terrence's class average was 75 points with a 2 point standard deviation. His score of 34 is uniquely terrible.

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

A z score is a signed (+ or -) number, like -1.5 or +2.

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.

12.2 Z-scores as a Standardized Distribution

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

Because the scores are standardized we can compare scores from two completely different scales. For example, if a biology exam was scored out of 100 points and a psychology exam was scored of 75 points we could compare our performance on them via z scores. The first observation in the dataset below got 60 points on both exams. However, the 60 on the psych exam is a "worse" score because everyone in the class did better.

```
# A tibble: 7 x 4
    bio   z_bio   psy  z_psy
  <dbl>   <dbl> <dbl>  <dbl>
1    60 -1.10      60 -1.73
2    78  0.0456    75  1.25
3    88  0.685     65 -0.739
4    57 -1.30      70  0.256
5    98  1.32      73  0.853
6    70 -0.466     70  0.256
7    90  0.812     68 -0.142
```

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

- Always has a mean of 0
- Always has a standard deviation of 1
- Most z-scores are between z = -2.00 and z = +2.00

12.3 Z-score Formula

12.3.1 Samples

$(x - M) / s$

12.3.2 Populations

$(x - M) / \sigma$

12.4 Calculating z-scores in R

To calculate a z score we will need to use the `scale()` function. First, we will create a vector of scores using the `sample()` function. When using the `sample()` function the first argument is the range of scores you are asking R to generate values from, then the `size` argument is how many observations you want, the final argument is `replace`. When `replace = TRUE` then the same number can appear twice. This is called sampling WITH replacement.

```
library(pacman)
p_load(rio, tidyverse, skimr)
```

```
sample(min value:max value, size = number of observations, replace = TRUE)
```

When combining `sample()` with the `tibble()` function to create a small dataframe our code looks like this.

```
small <- tibble(raw_scores = sample(1:10, size = 25, replace = TRUE))
small
```

```
# A tibble: 25 x 1
   raw_scores
        <int>
 1          9
 2          1
 3          1
 4          8
 5          9
 6          6
 7          2
 8          2
 9         10
10          4
# i 15 more rows
```

Now that we have a set of raw scores, let's transform them into z scores.

```
small <- small %>%
  mutate(z_scores = scale(raw_scores))

small
```

```
# A tibble: 25 x 2
   raw_scores z_scores[,1]
        <int>        <dbl>
 1          9       1.03
 2          1      -1.75
 3          1      -1.75
 4          8       0.681
 5          9       1.03
 6          6      -0.0139
 7          2      -1.40
```

```
 8           2      -1.40
 9          10       1.38
10           4      -0.709
# i 15 more rows
```

Did it work? Let's check out a quick snapshot of our data with the `skim()` function. We can see that the mean for the z scores is incredibly close to zero. Good enough. We also see that the standard deviation is 1. That is what we'd expect.

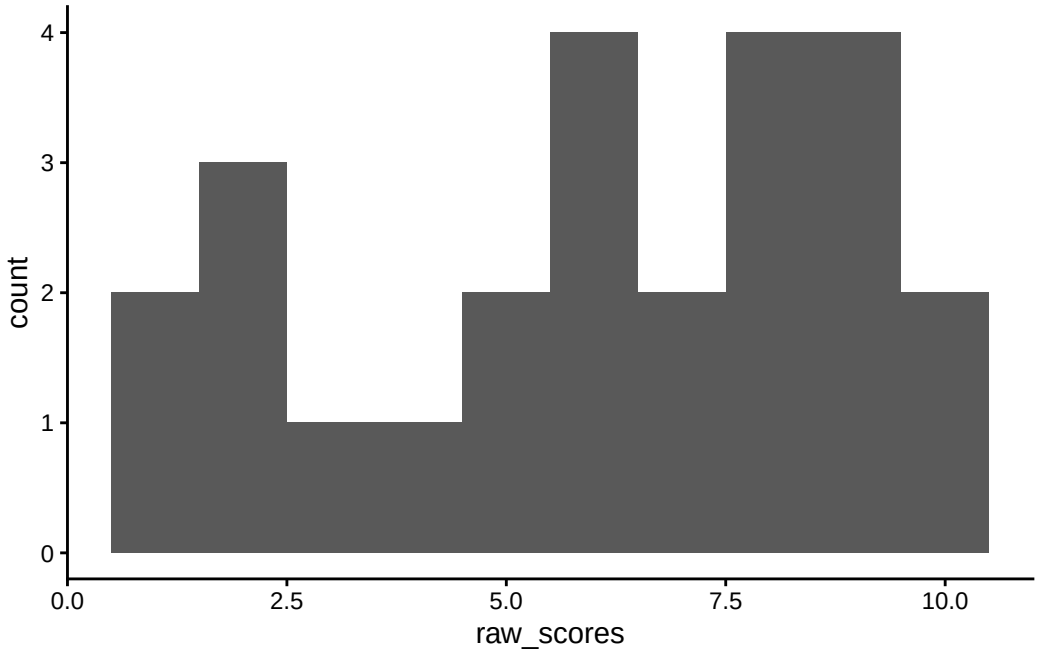
```
skim(small)
```

12.5 Visualize

- What do you notice when you compare the two distributions to one another?
- Are the x axes the same?
- Are the distributions the same shape?

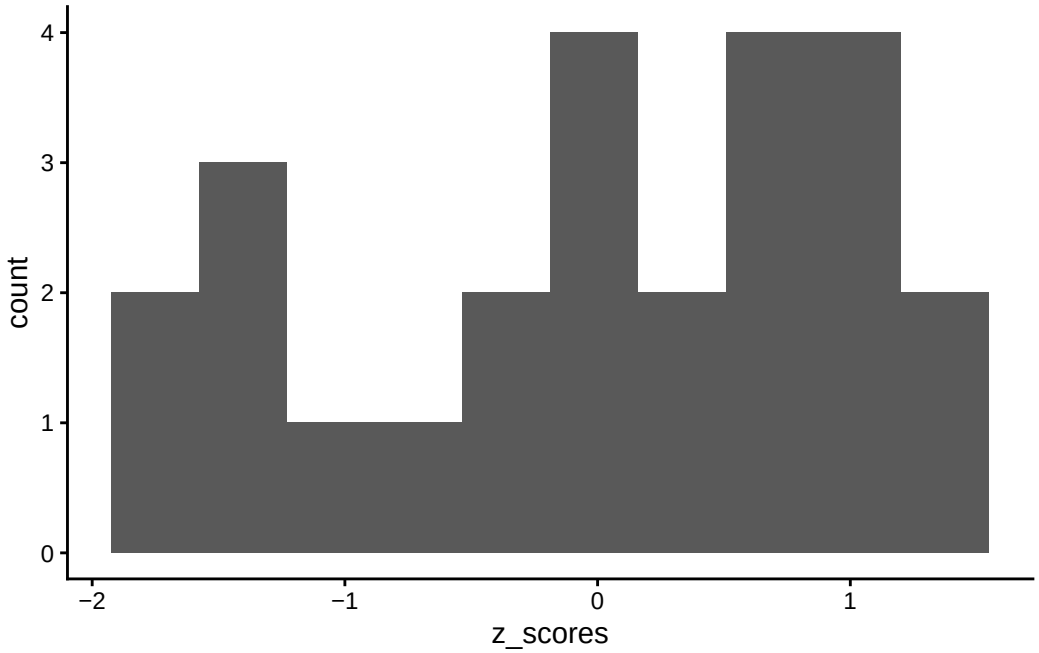
12.5.1 Raw score distribution

```
small %>%
  ggplot(aes(x = raw_scores)) +
  geom_histogram(bins = 10) +
  theme_classic()
```



12.5.2 Z score distribution

```
small %>%
  ggplot(aes(x = z_scores)) +
  geom_histogram(bins = 10) +
  theme_classic()
```



12.6 Larger Datasets

Let's try with larger datasets, one where the data is randomly distributed and one where the data is normally distributed.

```
random_dat <- tibble(raw_scores =sample(1:100, size = 1000, replace = TRUE))

norm_dat <- tibble(raw_scores = rnorm(1000, mean = 100, sd = 10))
```

12.6.1 find z for all scores in a dataset

```
# create a column in the dataframe named "z-score" using scale function

random_dat <- random_dat |>
  mutate(z_score = scale(raw_scores))

norm_dat <- norm_dat |>
  mutate(z_score = scale(raw_scores))
```

12.6.2 compare

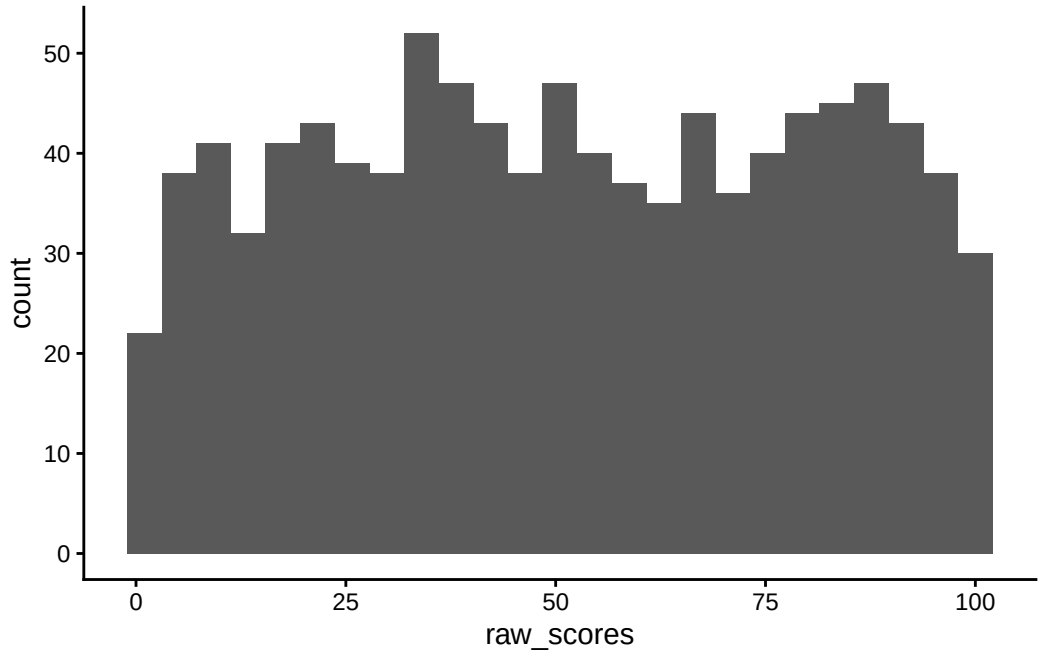
Despite very different raw values the mean for the z scores is zero and the standard deviation is 1.

```
skim(random_dat)
skim(norm_dat)
```

12.6.3 Visualize raw scores

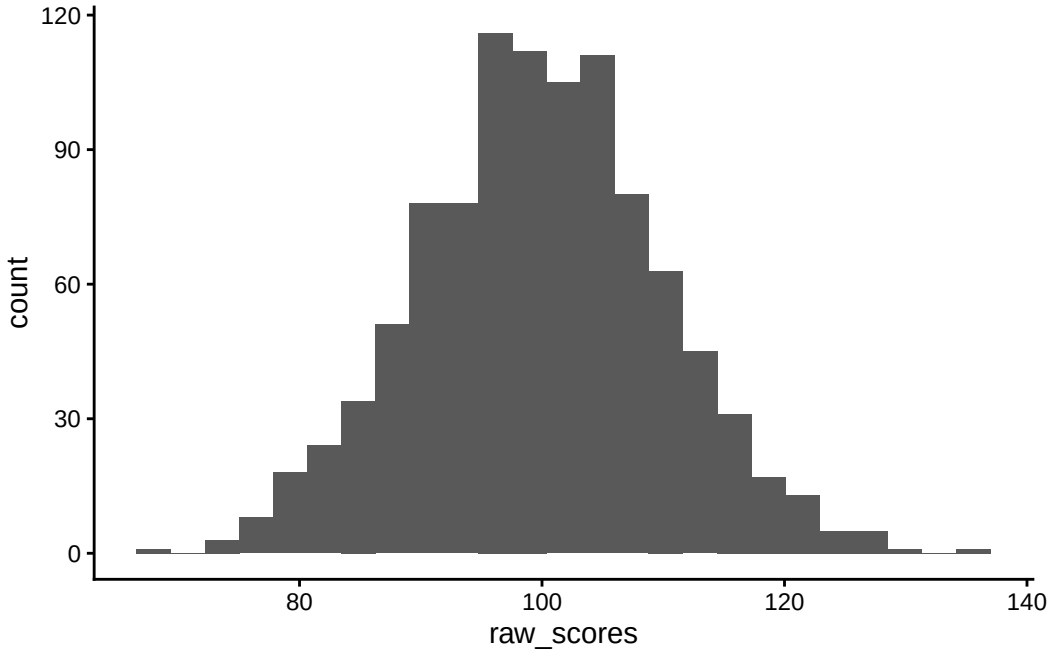
12.6.3.1 randomly distributed

```
# plot the raw score distribution

random_dat %>%
  ggplot(aes(x = raw_scores)) +
  geom_histogram(bins = 25) +
  theme_classic()
```



12.6.3.2 normally distributed

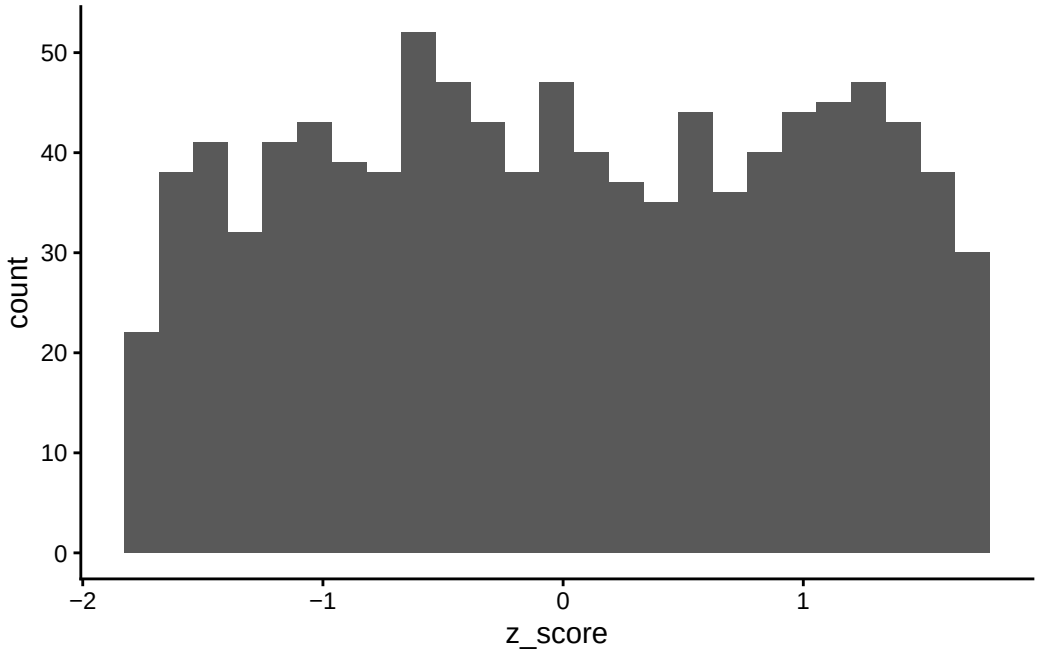
```
norm_dat %>%
  ggplot(aes(x = raw_scores)) +
  geom_histogram(bins = 25) +
  theme_classic()
```



12.6.4 Visualize z scores

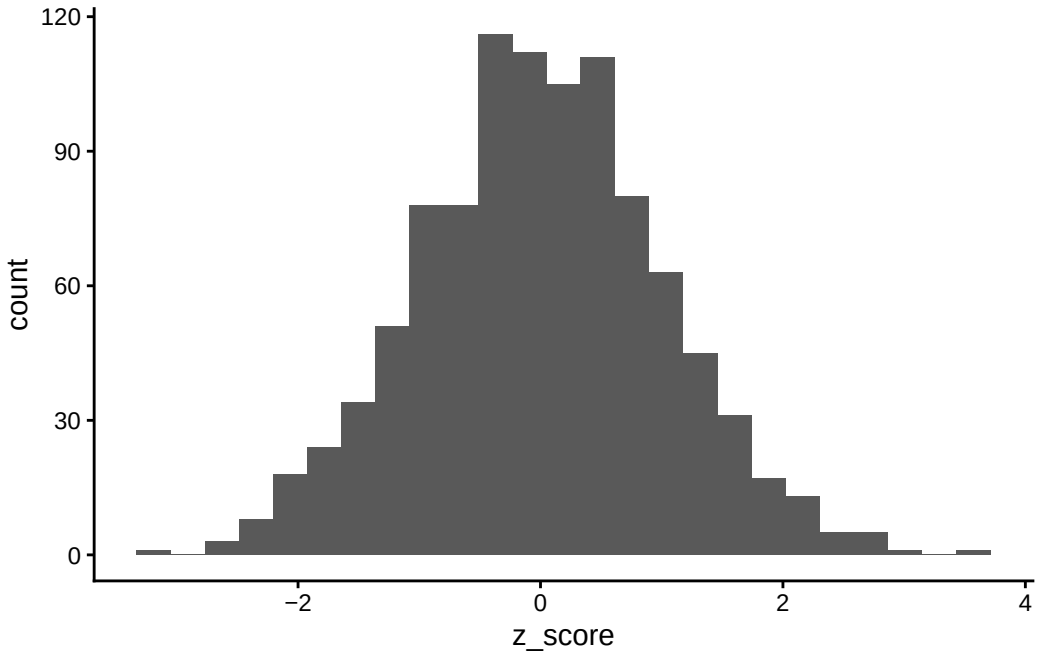
12.6.5 randomly distributed

```
random_dat %>%
  ggplot(aes(x = z_score)) +
  geom_histogram(bins = 25) +
  theme_classic()
```



12.6.6 normally distributed

```
norm_dat %>%
  ggplot(aes(x = z_score)) +
  geom_histogram(bins = 25) +
  theme_classic()
```



12.7 z scores and probability

We often need to know how likely it is to obtain a certain score. We can do this by using the `pnorm()` function. This function has four arguments we need to understand.

- `q` is the value that you want to calculate the probability of
- `mean` is the mean of the distribution
- `sd` is the standard deviation of the distribution
- `lower.tail` indicates whether we are looking at the lower tail or upper tail of the distribution. If it is FALSE, then we are looking at the upper tail.

PsyTeachR - Data Skills Chapter 13 and 14 are great for this

This code gives us the probability of obtaining a z score smaller than -1.96. We know this is a z distribution because the mean is 0 and the standard deviation is 1.

```
pnorm(q = -1.96, mean = 0, sd = 1, lower.tail = TRUE)
```

```
[1] 0.0249979
```

The probability value, or p, is $< .25$

The code below gives us the probability of randomly selecting a score of 700 or above from a dataset with a mean of 500, and a standard deviation of 100.

```
pnorm(q = 700, mean = 500, sd = 100, lower.tail = FALSE)
```

```
[1] 0.02275013
```

12.8 Z-scores and Locations

As descriptive statistics, z scores describe exactly where a score is located in a distribution.

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

13 Probability

13.1 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

13.2 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

13.3 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

13.4 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

14 Distribution of Sample Means

14.1 Samples vs. Populations

Though most behavioral science uses samples to test hypotheses, those hypotheses are almost always 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. They are approximations.

Sampling Error

The difference between sample statistics and population parameters.

14.2 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 samples are more representative of populations than smaller samples.

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

14.3 Standard Error of the Mean

The standard deviation of the distribution of sample means is the

Standard Error of the Mean

$= \sqrt{\quad} \quad M =$

n

14.4 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

```
#| standalone: true
#| viewerHeight: 600
library(shiny)
library(bslib)

ui <- page_sidebar(
  title = "Sampling Distribution Explorer",
  sidebar = sidebar(
    sliderInput("n", "Sample Size (n):", min = 2, max = 100, value = 30),
    sliderInput("sims", "Number of Simulations:", min = 5, max = 150, value = 1000, step = 5)
    actionButton("resample", "Generate New Samples", class = "btn-primary")
  ),
  card(
    card_header("Distribution of Sample Means"),
    plotOutput("distPlot")
  )
)

server <- function(input, output, session) {
  sample_means <- reactive({
    input$resample
    # Simulate drawing random samples from a normal population
    replicate(input$sims, mean(rnorm(input$n, mean = 50, sd = 10)))
  })
```

```
  output$distPlot <- renderPlot({
    means <- sample_means()
    hist(means, breaks = 30, col = "#007bc2", border = "white",
         main = paste("Distribution of", input$sims, "Means (n =", input$n, ")"),
         xlab = "Sample Mean Value")
    abline(v = 50, col = "red", lwd = 2, lty = 2) # Population Mean
  })
}

shinyApp(ui, server)
```

15 Hypothesis Testing

15.1 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.

15.2 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)

15.3 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.

15.4 Hypothesis Testing Table

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

15.5 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

16 T-tests

16.1 Introduction

T-tests compare means and help us determine if group differences we observe are statistically significant. In this chapter you will learn how to:

- Run one sample, independent, and paired t-tests in R
- Interpret output from t tests
- Test the assumptions of t tests
- Create APA style results using the `report` and `apaTables` packages

16.2 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.

```
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 becor
```

```
library(rstatix)
```

```
Attaching package: 'rstatix'

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

    filter
```

```
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.

16.3 What is a t-test?

A t-test is an 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.

16.3.1 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.

Ex. Comparing USI college athletes' (our sample) body fat percentage to the national average college student (our population) body fat percentage.

16.3.2 Independent samples t-test

Comparing two completely different groups in a between subjects design.

Ex. Comparing a group that received aromatherapy and a group that received no therapy on sinus health.

16.3.3 Paired t-test

Comparing one condition a group experiences to another condition that some group experiences.

Ex. Comparing relaxation scores before a massage and after a massage.

16.3.4 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)
)
```

16.4 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

16.4.1 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.

16.4.2 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

16.4.3 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.

16.5 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.

16.5.1 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.

16.5.2 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.

16.5.3 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 ($M = 13.61$, $SD = 2.90$), $t(16) = 12.38$, $p < .05$.

16.6 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

16.6.1 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.

16.7 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

16.7.1 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)
)
```

16.7.2 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.

16.7.3 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
```

16.7.4 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

```
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.

16.8 APA Tables

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

16.8.1 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.

16.9 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