Business Statistics

J. Alejandro Gelves

2024-09-06

Table of contents

Introduction

"Whatever you would make habitual, practice it; and if you would not make a thing habitual, do not practice it, but accustom yourself to something else." *Epictetus*

This course companion is designed to help you build mastery in statistics and its applications using R. Through practice, you will develop the skills and confidence needed to apply statistical concepts effectively. Each chapter begins with a list of key concepts to guide your learning, and the problems are crafted to reinforce these ideas through hands-on experience. If you need additional support while learning R, I encourage you to explore Grolemund (2014). Take your time, enjoy the process, and make practice a habit!

Why R?

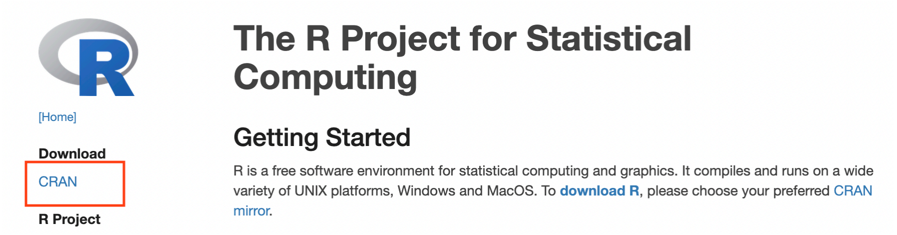
We will be using R to apply the lessons we learn in BUAD 231. R is a language and environment for statistical computing and graphics. There are several advantages to using the R software for statistical analysis and data science. Some of the main benefits include:

- R is a **powerful and flexible programming language** that allows users to manipulate and analyze data in many different ways.
- R has a large and **active community of users**, who have developed a wide range of packages and tools for data analysis and visualization.
- R is **free and open-source**, which makes it accessible to anyone who wants to use it.
- R is **widely used** in academia and industry, which means that there are many resources and tutorials available to help users learn how to use it.
- R is well-suited for working with **large and complex datasets**, and it can handle data from many different sources.
- R can be **easily integrated** with other tools and software, such as databases, visualization tools, and machine learning algorithms.

Overall, R is a powerful and versatile tool for data analysis and data science, and it offers many benefits to users who want to work with data.

Installing R.

To install R, visit the R webpage at https://www.r-project.org/. Once in the website, click on the CRAN hyperlink.

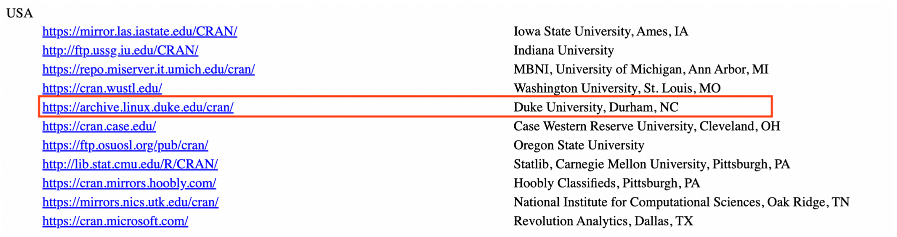


The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

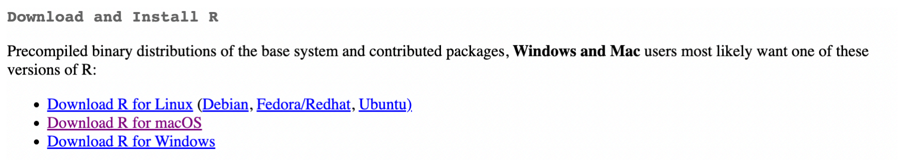
Here you can select the CRAN mirror. Scroll down until you see USA. You are free to choose any mirror you like, I recommend using the Duke University mirror.



USA

https://mirror.las.iastate.edu/CRAN/	Iowa State University, Ames, IA
http://ftp.ussg.iu.edu/CRAN/	Indiana University
https://repo.miserver.it.umich.edu/cran/	MBNI, University of Michigan, Ann Arbor, MI
https://cran.wustl.edu/	Washington University, St. Louis, MO
https://archive.linux.duke.edu/cran/	Duke University, Durham, NC
https://cran.case.edu/	Case Western Reserve University, Cleveland, OH
https://ftp.osuosl.org/pub/cran/	Oregon State University
http://lib.stat.cmu.edu/R/CRAN/	Statlib, Carnegie Mellon University, Pittsburgh, PA
https://cran.mirrors.hoobly.com/	Hoobly Classifieds, Pittsburgh, PA
https://mirrors.nics.utk.edu/cran/	National Institute for Computational Sciences, Oak Ridge, TN
https://cran.microsoft.com/	Revolution Analytics, Dallas, TX

Once you click on the hyperlink, you will be prompted to choose the download for your operating system. Depending on your operating system, choose either a Windows or Macintosh download.



Download and Install R

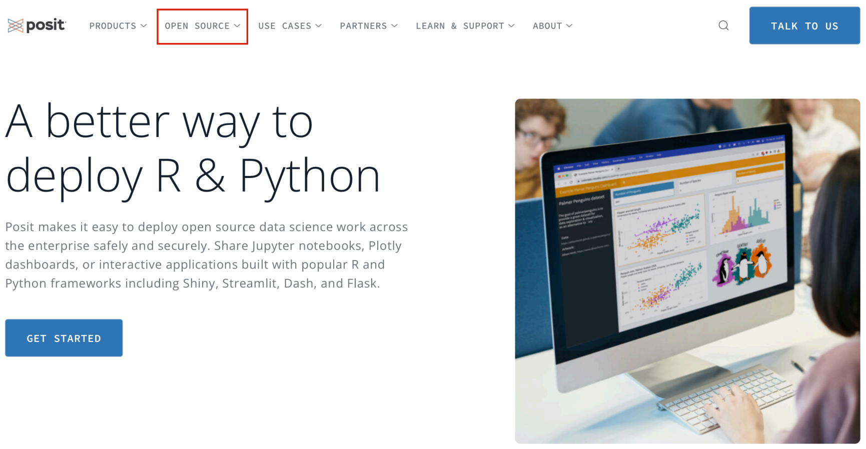
Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

Follow all prompts and complete installation.

Installing RStudio

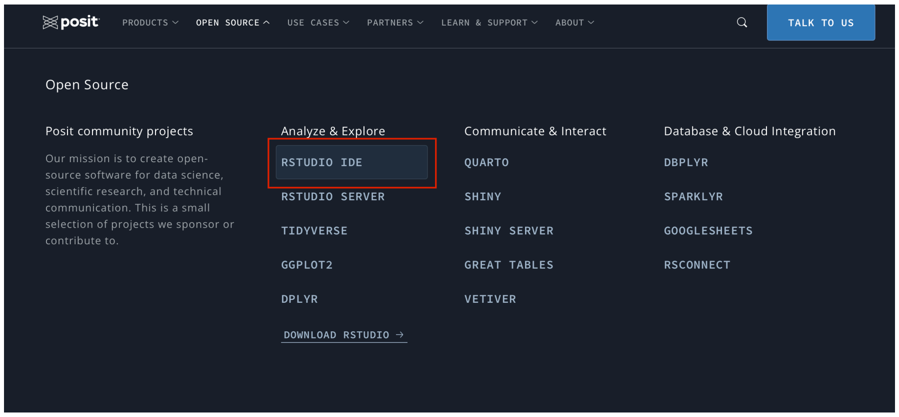
Visit the Posit website at https://posit.co. Once on the website, hover to the top of the screen and select "Open Source" from the drop down menus.



A better way to deploy R & Python

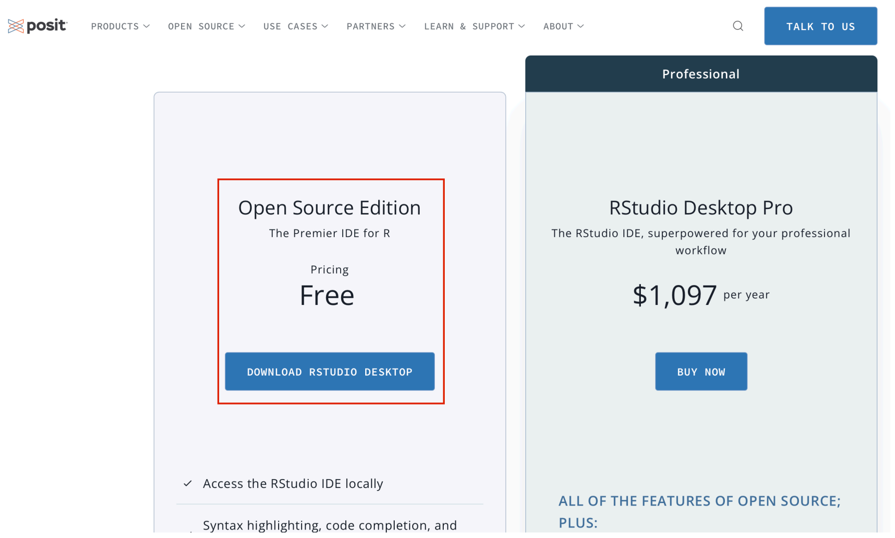
Posit makes it easy to deploy open source data science work across the enterprise safely and securely. Share Jupyter notebooks, Plotly dashboards, or interactive applications built with popular R and Python frameworks including Shiny, Streamlit, Dash, and Flask.

Next, choose "R Studio IDE".



Our mission is to create open-source software for data science, scientific research, and technical communication. This is a small selection of projects we sponsor or contribute to.

Scroll down until you see the products. You want to download "RStudio Desktop" and make sure it is the free version.



Open Source Edition

The Premier IDE for R

Pricing

Free

RStudio Desktop Pro

The RStudio IDE, superpowered for your professional workflow

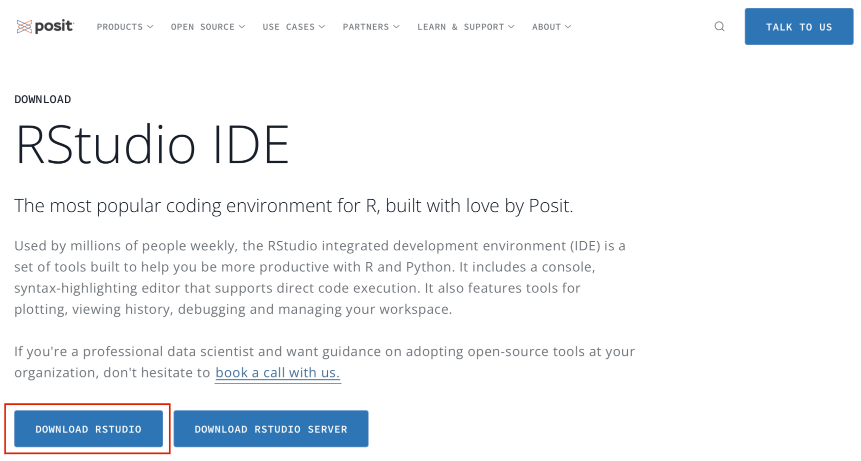
$1,097 per year

✓ Access the RStudio IDE locally

Syntax highlighting, code completion, and

ALL OF THE FEATURES OF OPEN SOURCE; PLUS:

Finally, select "Download RStudio" and follow the instructions for installation.



DOWNLOAD

RStudio IDE

The most popular coding environment for R, built with love by Posit.

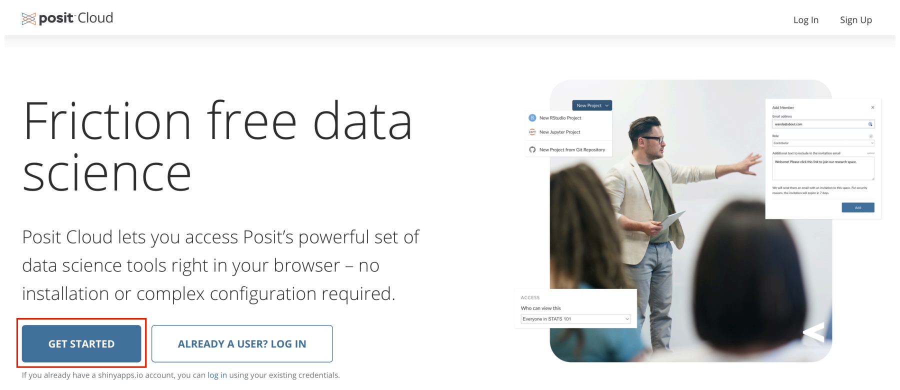
Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution. It also features tools for plotting, viewing history, debugging and managing your workspace.

If you're a professional data scientist and want guidance on adopting open-source tools at your organization, don't hesitate to book a call with us.

It is important to note that RStudio will not work if R is not installed. You can think of R as the engine and RStudio as the interface.

Posit Cloud

If you do not wish to install R, you can always use the cloud version. To do this, visit https://posit.cloud/. On the main page click on the "Get Started" button.



Friction free data science

Posit Cloud lets you access Posit's powerful set of data science tools right in your browser – no installation or complex configuration required.

If you already have a shinyapps.io account, you can log in using your existing credentials.

Choose the "Cloud Free" option and log in using your Google credentials (if you have a Google account) or sign up if you want to create a new account.

1 Descriptive Stats I

1.1 Motivation

Understanding the nature and classification of data is crucial for effective analysis and decision-making. Data are the building blocks of insights, providing a foundation for businesses, researchers, and policymakers to make informed choices. Whether capturing a snapshot of a specific moment, tracking changes over time, or organizing information in structured or unstructured formats, how data is collected and categorized significantly impacts how it is analyzed and interpreted. This overview highlights key types of data and their unique characteristics to help you better understand their application in various contexts.

1.2 Data and Types of Data

Data are facts and figures collected, analyzed and summarized for presentation and interpretation. Data can be classified as:

- **Cross Sectional Data** refers to data collected at the same (or approximately the same) point in time. *Ex: NFL standings in 1980 or Country GDP in 2015.*
- **Time Series Data** refers to data collected over several time periods. *Ex: U.S. inflation rate from 2000-2010 or Tesla deliveries from 2016-2022.*
- **Structured Data** resides in a predefined row-column format (tidy). *Ex: spreadsheet data.*
- **Unstructured Data** do not conform to a pre-defined row-column format. *Ex: Text, video, and other multimedia.*

Example: Consider a retail store analyzing its sales performance. If the store collects data on the total revenue generated *by each location* on Black Friday, it is cross-sectional data. On the other hand, if the store tracks *weekly sales for the past year* to observe trends, it is time series data. Structured data, like sales figures stored in *spreadsheets*, allows for easy comparison and analysis. Meanwhile, customer feedback gathered from *social media posts and video reviews* represents unstructured data, requiring advanced tools to extract meaningful insights.

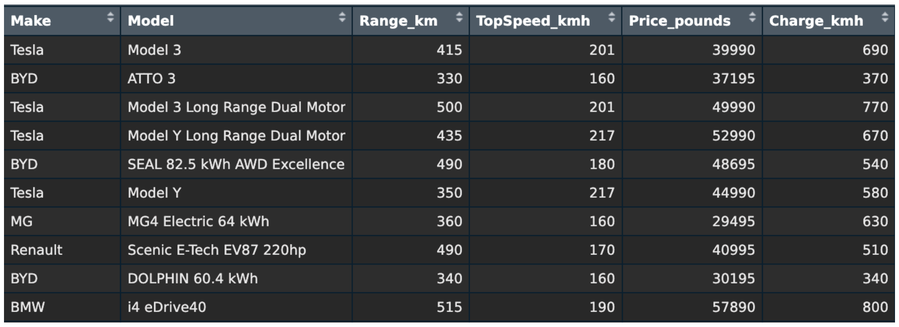
1.3 Data Sets

A **data set** contains all data collected for a particular study. Data sets are composed of:

- **Elements** are the entities on which data are collected. *Ex: Football teams, countries, and individuals.*
- **Variables** are a set of characteristics collected for each element. *Ex: Goals scored, GDP, weight.*
- **Observations** are the set of measurements obtained for a particular element. *Ex: Salah, 20 (goals), 15 (assists). US, 2.3 (inflation), 4.5% (federal interest rate).*

Elements	Variable 1	Variable 2
Element 1	#	#
Element 2	#	#
Element 3	#	#
...	...	...

Example: Consider the dataset on electric vehicles (EV's) displayed below:



Make	Model	Range_km	TopSpeed_kmh	Price_pounds	Charge_kmh
Tesla	Model 3	415	201	39990	690
BYD	ATTO 3	330	160	37195	370
Tesla	Model 3 Long Range Dual Motor	500	201	49990	770
Tesla	Model Y Long Range Dual Motor	435	217	52990	670
BYD	SEAL 82.5 kWh AWD Excellence	490	180	48695	540
Tesla	Model Y	350	217	44990	580
MG	MG4 Electric 64 kWh	360	160	29495	630
Renault	Scenic E-Tech EV87 220hp	490	170	40995	510
BYD	DOLPHIN 60.4 kWh	340	160	30195	340
BMW	i4 eDrive40	515	190	57890	800

In this dataset, each row represents an electric vehicle model, making the elements the specific EV models rather than the manufacturers. The variables collected for each model include:

- Make: The manufacturer of the EV.
- Model: The specific name of the EV model.
- Range_km: Driving range in kilometers on a full charge.
- TopSpeed_kmh: Maximum speed in km/h.
- Price_pounds: Price in pounds (£).
- Charge_kmh: Charging speed in kilometers per hour.

An example observation is "Tesla Model 3," with the following data: Make: Tesla, Model: Model 3, Range_km: 415, TopSpeed_kmh: 201, Price_pounds: 39,990, Charge_kmh: 690.

1.4 Scales of Measurement

Understanding scales of measurement is crucial for analyzing and interpreting data effectively in business. By distinguishing between categorical (e.g., marital status, satisfaction ratings) and numerical data (e.g., profits, prices), you'll know what methods to use for analysis. Knowing whether data is nominal, ordinal, interval, or ratio ensures your analysis and conclusions are accurate and relevant.

The **scales of measurements** determine the amount and type of information contained in each variable. In general, variables can be classified as **categorical** or **numerical**.

- **Categorical** (qualitative) data includes labels or names to identify an attribute of each element. Categorical data can be **nominal** or **ordinal**.
  - With **nominal** data, the order of the categories is arbitrary. *Ex: Marital Status, Race/Ethnicity, or NFL division.*
  - With **ordinal** data, the order or rank of the categories is meaningful. *Ex: Rating, Difficulty Level, or Spice Level.*
- **Numerical** (quantitative) include numerical values that indicate how many (discrete) or how much (continuous). The data can be either **interval** or **ratio**.
  - With **interval** data, the distance between values is expressed in terms of a fixed unit of measure. The zero value is arbitrary and does not represent the absence of the characteristic. Ratios are not meaningful. *Ex: Temperature or Dates.*
  - With **ratio** data, the ratio between values is meaningful. The zero value is not arbitrary and represents the absence of the characteristic. *Ex: Prices, Profits, Wins.*

Example: Let's keep using the EV example. Consider the new data set below:

Car	Brand	Range	Rating	Year
Mustang Mach-E	Ford	217	4	2021
E-Tron GT	Audi	250	3	2020
...	...	...	...	
Volt EV	Chevrolet	124	2	2021

The variables can be classified as follows: Car (Categorical - Nominal), consists of names of cars, which are labels used to identify each row. The order of these names does not matter, making it nominal data. Brand (Categorical - Nominal) represents the manufacturer of the car (e.g., Ford, Audi). These are labels with no inherent order, making it nominal data. Range

(Numerical - Ratio), refers to the car's driving range in miles. It is numerical and ratio because it has a meaningful zero (a car with zero range cannot move), and ratios are meaningful (e.g., a car with 250 miles range has double the range of one with 125 miles). Rating (Categorical - Ordinal) represents a rank or score (e.g., 4, 3, 2). The order matters, as higher ratings indicate better performance. However, the intervals between ratings are not consistent, so it is ordinal data. Year (Numerical - Interval) represents a point in time. While numerical, it is interval data because the zero point is arbitrary (e.g., year 0 does not indicate the "absence" of time), and ratios are not meaningful (e.g., 2020 is not "twice as late" as 1010).

1.5 Useful Base R Functions

Understanding and using Base R functions is essential for efficiently managing and analyzing data. Functions like `na.omit()` help clean datasets by removing rows with missing values, ensuring your analyses are accurate and complete. `nrow()` and `ncol()` quickly provide insights into the size of your dataset, while `is.na()` allows you to identify and address missing data. The `summary()` function is a powerful way to generate descriptive statistics and assess the overall structure of your data at a glance. Additionally, coercion functions like `as.integer()`, `as.factor()`, and `as.double()` enable you to convert variables to appropriate data types, ensuring compatibility with different analysis methods.

- The `na.omit()` function removes any observations that have a missing value (NA). The resulting data frame has only complete cases. *Input: A data frame (tibble) or vector.*
- The `nrow()` and `ncol()` functions return the number of rows and columns respectively from a data frame. *Input: A data frame (tibble).*
- The `is.na()` function returns a vector of *True* and *False* that specify if an entry is missing (NA) or not. *Input: A data frame (tibble) or vector.*
- The `summary()` function returns a collection of descriptive statistics from a data frame (or vector). The function also returns whether there are any missing values (NA) in a variable. *Input: A data frame (tibble) or vector.*
- The `as.integer()`, `as.factor()`, `as.double()`, are functions used to coerce your data into a different scale of measurement. *Input: A vector or column of a data frame (tibble).*

1.6 Useful DPLYR Functions

The `dplyr` package has a collection of functions that are useful for data manipulation and transformation. If you are interested in this package you can refer to Wickham (2017). To install, run the following command in the console `install.packages("dplyr")`.

- The `arrange()` function allows you to sort data frames in ascending order. Pair with the `desc()` function to sort the data in descending order.

- The `filter()` function allows you to subset the rows of your data based on a condition.
- The `select()` function allows you to select a subset of variables from your data frame.
- The `mutate()` function allows you to create a new variable.
- The `group_by()` function allows you to group your data frame by categories present in a given variable.
- The `summarise()` function allows you to summarise your data, based on groupings generated by the `goup_by()` function.

1.7 Exercises

The following exercises will help you test your knowledge on the Scales of Measurement. They will also allow you to practice some basic data "wrangling" in R. In these exercises you will:

- Identify numerical and categorical data.
- Classify data according to their scale of measurement.
- Sort and filter data in R.
- Handle missing values (NA's) in R.

Answers are provided below. Try not to peak until you have a formulated your own answer and double checked your work for any mistakes.

Exercise 1

A bookstore has compiled data set on their current inventory. A portion of the data is shown below:

Title	Price	Year Published	Rating
Frankenstein	5.49	1818	4.2
Dracula	7.60	1897	4.0
...	...	...	...
Sleepy Hollow	6.95	1820	3.8

1. Which of the above variables are categorical and which are numerical?

Suggested Answer

The "Title" variable represents the names of books. Therefore, this is a categorical variable. "Price" represents the cost of each book in a numeric format, making it a numerical variable. "Year Published" indicates the publication year of each book. It is numerical. If "Rating" represents a numerical score based on a continuous scale (e.g., average user ratings on a platform like

Goodreads), it is numerical because arithmetic operations like averaging or comparing differences are meaningful. If "Rating" represents predefined categories (e.g., "Excellent," "Good," "Fair," "Poor") or is interpreted as ranks without meaningful differences between values, it would be categorical.

2. What is the measurement scale of each of the above variable?

Suggested Answer

The measurement scale is nominal for Title since these are labels used to identify each book and do not have a numerical meaning or order. If Rating represents a score (e.g., 4.2, 4.0) given to each book, it is numerical and could be considered interval data because the scale represents a meaningful difference, but it may not have an absolute zero or meaningful ratios (e.g., a book rated 4.0 is not "twice as good" as one rated 2.0). Price is a measurable quantity with a meaningful zero (e.g., a book priced at $0 means it is free), making it ratio data. Year is interval data because the zero point is arbitrary (year 0 does not represent the absence of time) and differences between years are meaningful (e.g., 1897 - 1818 = 79 years).

Exercise 2

A car company tracks the number of deliveries every quarter. A portion of the data is shown below:

Year	Quarter	Deliveries
2016	1	14800
2016	2	14400
...	...	...
2022	3	343840

1. What is the measurement scale of the Year variable? What are the strengths and weaknesses of this type of measurement scale?

Suggested Answer

The variable Year is measured on the interval scale because the observations can be ranked, categorized and measured when using this kind of scale. However, there is no true zero point so we cannot calculate meaningful ratios between years.

2. What is the measurement scale for the Quarter variable? What is the weakness of this type of measurement scale?

Suggested Answer

The variable Quarter is measured on the ordinal scale, even though it contains numbers. It is the least sophisticated level of measurement because if we are presented with nominal data, all we can do is categorize or group the data.

3. What is the measurement scale for the Deliveries variable? What are the strengths of this type of measurement scale?

Suggested Answer

The variable Deliveries is measured on the ratio scale. It is the strongest level of measurement because it allows us to categorize and rank the data as well as find meaningful differences between observations. Also, with a true zero point, we can interpret the ratios between observations.

Exercise 3

Use the **airquality** data set included in R for this problem.

1. Sort the data by *Temp* in descending order. What is the day and month of the first observation on the sorted data?

Suggested Answer

The day and month of the first observation is August 28th.

The easiest way to sort in R is by using the `dplyr` *package. Specifically, the* `arrange()` *function within the package. Let's also use the* `desc()` *function to make sure that the data is sorted in descending order. We can use indexing to retrieve the first row of the sorted data set.*

```
library(dplyr)
SortedAQ<-arrange(airquality,desc(Temp))
SortedAQ[1,]
```

```
  Ozone Solar.R Wind Temp Month Day
1    76     203  9.7   97     8  28
```

2. Sort the data only by *Temp* in descending order. Of the 10 hottest days, how many of them were in July?

Suggested Answer

We can use the `arrange()` *function one more time for this question. Then we can use indexing to retrieve the top* 10 *observations.*

```r
SortedAQ2<-arrange(airquality,desc(Temp))
SortedAQ2[1:10,]
```

```
   Ozone Solar.R Wind Temp Month Day
1     76     203  9.7   97     8  28
2     84     237  6.3   96     8  30
3    118     225  2.3   94     8  29
4     85     188  6.3   94     8  31
5     NA     259 10.9   93     6  11
6     73     183  2.8   93     9   3
7     91     189  4.6   93     9   4
8     NA     250  9.2   92     6  12
9     97     267  6.3   92     7   8
10    97     272  5.7   92     7   9
```

3. How many missing values are there in the data set? What rows have missing values for *Solar.R*?

Suggested Answer

There are a total of 44 *missing values. Ozone has* 37 *and Solar.R has* 7*. Rows* 5, 6, 11, 27, 96, 97, 98 *are missing for Solar.R.*

We can easily identify missing values with the `summary()` *function.*

```r
summary(airquality)
```

```
     Ozone           Solar.R           Wind             Temp      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
 NA's   :37       NA's   :7                                       
     Month            Day      
 Min.   :5.000   Min.   : 1.0  
 1st Qu.:6.000   1st Qu.: 8.0  
 Median :7.000   Median :16.0  
 Mean   :6.993   Mean   :15.8  
 3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :9.000   Max.   :31.0  
```

To view the rows that have NA's in them, we can use the `is.na()` *function and indexing. Below we see that 7 values are missing for the Solar.R variable in the months 5 and 8 combined.*

```
airquality[is.na(airquality$Solar.R),]
```

```
   Ozone Solar.R Wind Temp Month Day
5     NA      NA 14.3   56     5   5
6     28      NA 14.9   66     5   6
11     7      NA  6.9   74     5  11
27    NA      NA  8.0   57     5  27
96    78      NA  6.9   86     8   4
97    35      NA  7.4   85     8   5
98    66      NA  4.6   87     8   6
```

4. Remove all observations that have a missing values. Create a new object called *CompleteAG*.

Suggested Answer

To create the new object of complete observations we can use the `na.omit()` *function.*

```
CompleteAQ<-na.omit(airquality)
```

5. When using *CompleteAG*, how many days was the temperature at least 60 degrees?

Suggested Answer

There were 107 *days where the temperature was at least* 60*.*

Using base R we have:

```
nrow(CompleteAQ[CompleteAQ$Temp>=60,])
```

```
[1] 107
```

We can also use `dplyr` *for this question. Specifically, using the* `filter()` *and* `nrow()` *functions we get:*

```
nrow(filter(CompleteAQ,Temp>=60))
```

```
[1] 107
```

6. When using *CompleteAG*, how many days was the temperature within [55,75] degrees and an *Ozone* below 20?

Suggested Answer

There were 24 *days where the temperature was between* 55 *and* 75 *and the ozone level was below* 20*.*

Using base R we have:

```
nrow(CompleteAQ[CompleteAQ$Temp>55 & CompleteAQ$Temp<75 & CompleteAQ$Ozone<20,])
```

```
[1] 24
```

Using the `filter()` function once more we get:

```
nrow(filter(CompleteAQ,Temp>55,Temp<75,Ozone<20))
```

```
[1] 24
```

Exercise 4

Use the **Packers** data set for this problem. You can find the data set at https://jagelves.github.io/Data/Packers.c

1. Remove the any observation that has a missing value with the `na.omit()` function. How many observations are left in the data set?

Suggested Answer

There are 84 *observations in the complete cases data set.*

Let's import the data to R by using the `read.csv()` function.

```
Packers<-read.csv("https://jagelves.github.io/Data/Packers.csv")
```

We can remove any missing observation by using the `na.omit()` function. We can name this new object Packers2.

```
Packers2<-na.omit(Packers)
```

To find the number of observations we can use the `dim()` function. It returns the number of observations and variables.

```r
dim(Packers2)
```

```
[1] 84  8
```

2. Determine the type of the *Experience* variable by using the `typeof()` function. What type is the variable?

Suggested Answer

The type is character.

Use the `typeof()` function on the Experience variable.

```r
typeof(Packers2$Experience)
```

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

3. Remove observations that have an "R" and coerce the *Experience* variable to an integer using the `as.integer()` function. What is the total sum of years of experience?

Suggested Answer

The total sum of experience is 288.

First, remove any observation with an R by using indexing and logicals.

```r
Packers2<-Packers2[Packers2$Experience!="R",]
```

Now we can coerce the variable to an integer by using the `as.integer()` function.

```r
Packers2$Experience<-as.integer(Packers2$Experience)
```

Lastly, calculate the sum using the `sum()` function.

```r
sum(Packers2$Experience)
```

```
[1] 288
```

2 Descriptive Stats II

2.1 Concepts

Frequency

A **frequency distribution** is a tabular summary of data showing the number of items in each of several non-overlapping classes.

- The **relative frequency** is calculated by f_i/n, where f_i is the frequency of class i and n is the total frequency.
- The **cumulative frequency** shows the number of data items with values less than or equal to the upper class limit of each class.
- The **cumulative relative frequency** is given by cf_i/n, where cf_i is the cumulative frequency of class i.

Plots

A **bar plot** illustrates the frequency distribution of qualitative data.

- Is an illustration for qualitative data.
- Includes the classes in the horizontal axis and frequencies or relative frequencies in the vertical axis.
- Has gaps between each bar.

A **histogram** illustrates the frequency distribution of quantitative data.

- Is an illustration for quantitative data.
- There are no gaps between the bars.
- The **number**, **width** and **limits** of each class must be determined.
  - The **number** of classes can be determined by the 2^k rule: select k such that 2^k is greater than the number of observations by the smallest amount.
  - The **width** of the class is approximately *range/(# of Classes)*. The value should be rounded up.

    - The **limits** should be chosen so that each point belongs to only one class.

Useful R Functions

The `table()` command generates frequency distributions or contingency tables if two variables are used.

The `prop.table()` command generates relative frequency distributions from an object that contains a table.

The `cut()` function generates class limits and bins used in frequency distributions (and histograms) for quantitative data.

Base R has the `barplot()` function for categorical variable, `histogram()` function for numerical data, and the `plot()` function for line charts or scatter plots. Below are some arguments that are helpful when plotting.

- *main*: used to set the plot's title. The title should be entered as a character.
- *col*: used to set the color of the plot. Hex and RGB values are allowed as inputs. The color should be entered as a character.
- *xlab* and *ylab*: are used to set the labels for the x and y axis respectively. The labels should be entered as characters.
- `legend()` is a function to customize the legend of a graph. This argument may be used with the `plot()`, `barplot()` or `histogram()` functions.
    - *x*: used to set the location of the legend in the plotting area. Ex: "bottomleft".
    - *legend*: a vector specifying the legend names to be included.
    - *col*: a vector specifying the color of each item in the legend.

2.2 Exercises

The following exercises will help you practice summarizing data with tables and simple graphs. In particular, the exercises work on:

- Developing frequency distributions for both categorical and numerical data.
- Constructing bar charts, histograms, and line charts.
- Creating contingency tables.

Answers are provided below. Try not to peak until you have a formulated your own answer and double checked your work for any mistakes.

Exercise 1

Install the `ISLR2` package in R. You will need the **BrainCancer** data set to answer this question.

1. Construct a frequency and relative frequency table of the *Diagnosis* variable. What was the most common diagnosis? What percentage of the sample had this diagnosis?
2. Construct a bar chart. Summarize the findings.
3. Construct a contingency table that shows the *Diagnosis* along with the *Status*. Which diagnosis had the highest number of non-survivals (0)? What was the survival rate of this diagnosis?
4. Construct a stacked column chart. Which two *Diagnosis* and *Status* combinations are the most frequent?

Exercise 2

You will need the **airquality** data set (in base R) to answer this question.

1. Construct a frequency distribution for *Temp*. Use five intervals with widths of $50 < x \leq 60$; $60 < x \leq 70$; etc. Which interval had the highest frequency? How many times was the temperature between 50 and 60 degrees?
2. Construct a relative frequency, cumulative frequency and the relative cumulative frequency distributions. What proportion of the time was *Temp* between 50 and 60 degrees? How many times was the *Temp* 70 degrees or less? What proportion of the time was *Temp* more than 70 degrees?
3. Construct the histogram. Is the distribution symmetric? If not, is it skewed to the left or right?

Exercise 3

You will need the **Portfolio** data set from the `ISLR2` package to answer this question.

1. Construct a line chart that shows the returns over time for each portfolio (X and Y) by using two lines each with a unique color. Assume the data is for the period 1901 to 2000. Include also a legend that matches colors to portfolios.

2.3 Answers

Exercise 1

1. The most common diagnosis is Meningioma, a slow-growing tumor that forms from the membranous layers surrounding the brain and spinal cord. The diagnosis represents about 48.28% of the sample.

Start by loading the `ISLR2` package. To construct the frequency distribution table, use the `table()` function.

```r
library(ISLR2)
table(BrainCancer$diagnosis)
```

```

Meningioma  LG glioma  HG glioma      Other 
        42          9         22         14 
```

The relative frequency distribution can be easily retrieved by saving the frequency table in an object and then using the `prop.table()` function.

```r
freq<-table(BrainCancer$diagnosis)
prop.table(freq)
```

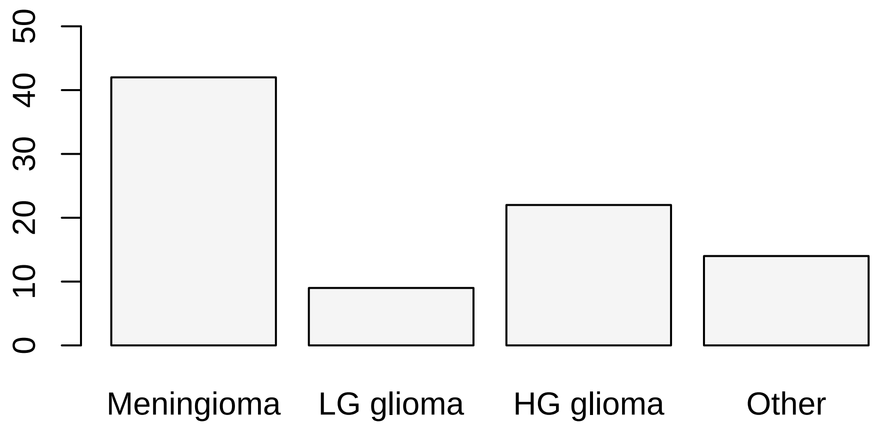
```

Meningioma  LG glioma  HG glioma      Other 
 0.4827586  0.1034483  0.2528736  0.1609195 
```

2. The majority of diagnosis are Meningioma. Low grade glioma is the least common of diagnosis. High grade glioma and other diagnosis have about the same frequency.

To construct the bar chart use the `barplot()` function in R.

```r
barplot(freq, col = "#F5F5F5", ylim=c(0,50))
```



3. 33 people did not survive Meningioma. The survival rate of Meningioma is only 21.43%.

Use the `table()` function one more time to create the contingency table for the two variables.

```r
(freq2<-table(BrainCancer$status,BrainCancer$diagnosis))
```

```
   Meningioma LG glioma HG glioma Other
  0         33         5         5     9
  1          9         4        17     5
```

To get the survival rates, we can use the `prop.table()` function once again.

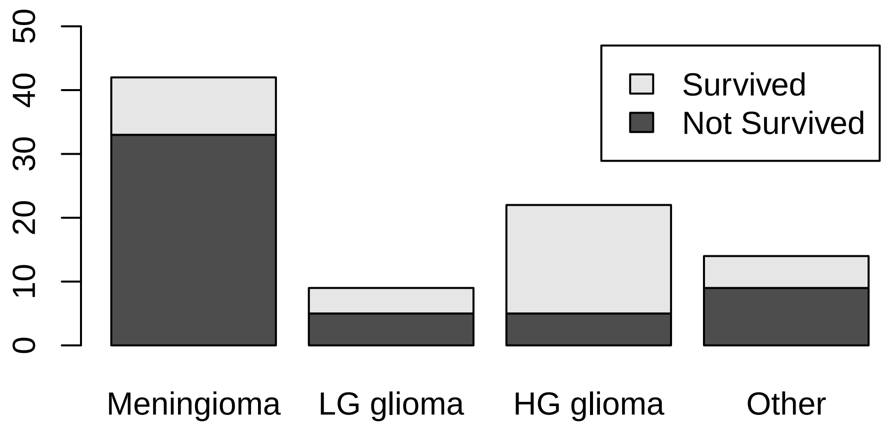
```r
prop.table(freq2,margin = 2)
```

```
   Meningioma LG glioma HG glioma     Other
  0  0.7857143 0.5555556 0.2272727 0.6428571
  1  0.2142857 0.4444444 0.7727273 0.3571429
```

4. Meningioma and not surviving is the most common with 33 occurrences. High grade glioma and surviving is the the second most common.

Use the `barplot()` function one more time to construct the stacked column chart.

```r
barplot(table(BrainCancer$status,BrainCancer$diagnosis),
        legend.text = c("Not Survived","Survived"), ylim=c(0,50))
```



Exercise 2

1. The highest frequency is in the $80 < x \leq 90$ bin. 8 temperatures were between $50 < x \leq 60$ degrees.

Create a vector containing the intervals desired by using the `seq()` function.

```
intervals <- seq(50, 100, by=10)
```

Next use the `cut()` function to create the cuts for the histogram.

```
intervals.cut <- cut(airquality$Temp, intervals, left=FALSE, right=TRUE)
```

The frequency distribution can be obtained by using the `table()` function on the *interval.cut* object created above.

```
table(intervals.cut)
```

```
intervals.cut
 (50,60]  (60,70]  (70,80]  (80,90] (90,100] 
       8       25       52       54       14 
```

2. The temperature was 5.22% of the time between 50 and 60; The temperature was 70 or less 33 times; The temperature was above 70, 78.43% of the time.

To get the relative frequency table, start by saving the proportion table into an object.Then you can use the `prop.table()` function.

```r
freq<-table(intervals.cut)
prop.table(freq)
```

```
intervals.cut
   (50,60]    (60,70]    (70,80]    (80,90]   (90,100] 
0.05228758 0.16339869 0.33986928 0.35294118 0.09150327 
```

For the cumulative distribution you can use the `cumsum()` function on the frequency distribution.

```r
cumulfreq<-cumsum(freq)
cumulfreq
```

```
 (50,60]  (60,70]  (70,80]  (80,90] (90,100] 
       8       33       85      139      153 
```

Lastly, for the relative cumulative distribution table, you can use the `cumsum()` function on the relative frequency table.

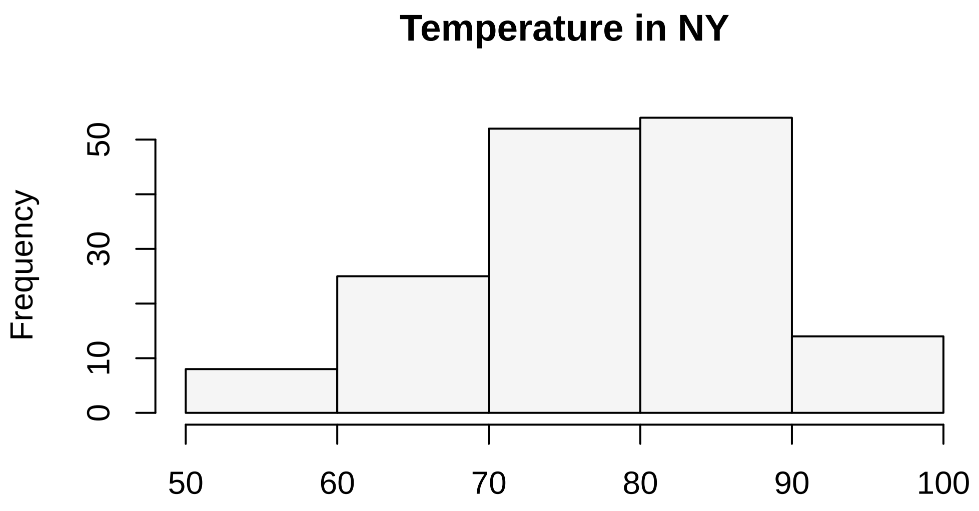
```r
cumsum(prop.table(freq))
```

```
   (50,60]    (60,70]    (70,80]    (80,90]   (90,100] 
0.05228758 0.21568627 0.55555556 0.90849673 1.00000000 
```

3. The distribution is not perfectly symmetric. It is skewed slightly to the left (see histogram.)

Use the `hist()` function to create the histogram.

```r
hist(airquality$Temp, breaks=intervals,
     right=TRUE,col="#F5F5F5", main="Temperature in NY", xlab="")
```

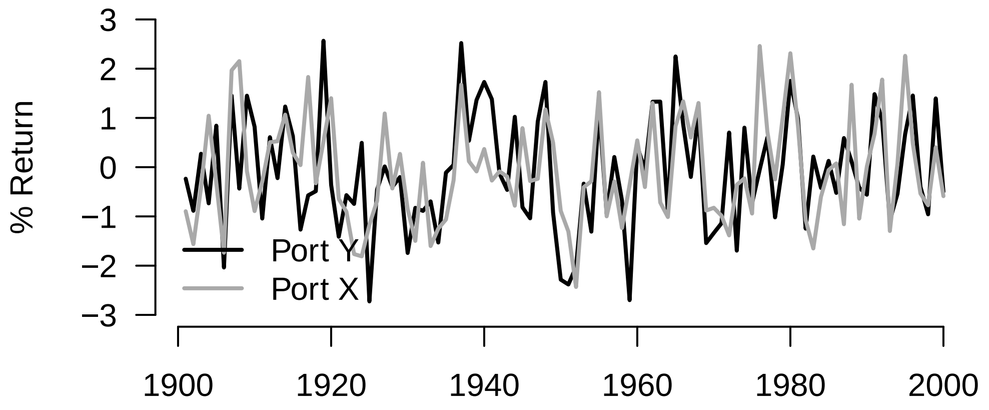


Exercise 3

1. From 1901 through 2000, both portfolios have behaved very similarly. Returns are between -3% and 3%, there is no trend, and positive (negative) returns for X seem to match with positive (negative) returns of Y.

You can use the `plot()` function to create a plot of Portfolio Y. The line for Portfolio X can be added with the `lines()` function.

```
plot(Portfolio$Y,
     x=seq(1901,2000), type="l",
     col="black", xlab="", ylab="% Return", ylim=c(-3,3),
     xlim=c(1901,2000), lwd=2, axes = F)
axis(side=1, labels=TRUE, font=1,las=1)
axis(side=2, labels=TRUE, font=1,las=1)
lines(Portfolio$X, x=seq(1901,2000), type="l",
      col="darkgrey", lwd=2)
legend(x = "bottomleft",
       legend = c("Port Y", "Port X"),
       lty = c(1, 1),
       col = c("black", "darkgrey"),
       lwd = 2,
       bty="n")
```



% Return
3
2
1
0
−1
−2
−3
Port Y
Port X
1900
1920
1940
1960
1980
2000

3 Descriptive Statistics III

3.1 Concepts

Measures of Central Location

Measures of Central Location determine where the center of a distribution lies.

- The **mean** is the average value for a numerical variable. The sample statistic is estimated by $\bar{x} = \sum x_i/n$, where x_i is observation i, and n is the number of observations. The population parameter is defined as $\mu = \sum x_i/N$.
- The **median** is the value in the middle when data is organized in ascending order. When n is even, the median is the average between the two middle values.
- The **mode** is the value with highest frequency from a set of observations.
- The **weighted mean** uses weights to determine the importance of each data point of a variable. It is calculated by $\frac{\sum w_i x_i}{\sum w_i}$, where w_i are the weights associated to the values x_i.
- The **geometric mean** is a multiplicative average that is less sensitive to outliers. It is used to average growth rates or rated of return. It is calculated by $\sqrt[n]{(1+r_1)*(1+r_2)...(1+r_n)} - 1$, where $\sqrt[n]{}$ is the n_{th} root, and r_i are the returns or growth rates.

Useful R functions

Base R has a collection of functions that calculate measures of central location.

- The `mean()` function calculates the average of a vector of values.
- The `median()` function returns the median of a vector of values.
- The `table()` function provides us with a frequency distribution. We can then identify the mode(s) of the vector provided.
- The `summary()` function returns a collection of descriptive statistics for a vector or data frame.

3.2 Exercises

The following exercises will help you practice the measures of central location. In particular, the exercises work on:

- Calculating the mean, median, and the mode.
- Calculating the weighted average.
- Applying the geometric mean for growth rates and returns.

Answers are provided below. Try not to peak until you have a formulated your own answer and double checked your work for any mistakes.

Exercise 1

For the following exercises, make your calculations by hand and verify results using R functions when possible.

1. Use the following observations to calculate the mean, the median, and the mode.

8	10	9	12	12

2. Use following observations to calculate the mean, the median, and the mode.

-4	0	-6	1	-3	-4

3. Use the following observations, calculate the mean, the median, and the mode.

20	15	25	20	10	15	25	20	15

Exercise 2

Download the `ISLR2` package. You will need the **OJ** data set to answer this question.

1. Find the mean price for Country Hill (*PriceCH*) and Minute Maid (*PriceMM*).
2. Find the mean price of Country Hill (*PriceCH*) in store 1 and store 2 (*StoreID*). Which store had the better price?
3. Find the mean price paid by Country Hill (*PriceCH*) purchasers (*Purchase*) in store 1 (*StoreID*)? How about store 2? Which store had the better price?

Exercise 3

1. Over the past year an investor bought TSLA. She made these purchases on three occasions at the prices shown in the table below. Calculate the average price per share.

Date	Price Per Share	Number of Shares
February	250.34	80
April	234.59	120
Aug	270.45	50

2. What would have been the average price per share if the investor would have bought equal amounts of shares each month?

Exercise 4

1. Consider the following observations for the consumer price index (CPI). Calculate the inflation rate (Growth Rate of the CPI) for each period.

1.0	1.3	1.6	1.8	2.1

2. Suppose that you want to invest $1000 dollars in a stock that is predicted to yield the following returns in the next four years. Calculate both the arithmetic mean and the geometric mean. Use the geometric mean to estimate how much money you would have by the end of year 4.

Year	Annual Return
1	17.3
2	19.6
3	6.8
4	8.2

3.3 Answers

Exercise 1

1. To find the mean we will use the following formula ($\frac{1}{n}\sum_{i=i}^{n} x_i$). The summation of the values is 51 and the number of observations is 5. The mean is $51/5 = 10.2$.

The median is found by locating the middle value when data is sorted in ascending order. The median in this example is 10.

The mode is the value with the highest frequency. In this example the mode is 12 since it is repeated twice and all other numbers appear only once.

The mean can be easily verified in R by using the `mean()` function:

```r
mean(c(8,10,9,12,12))
```

```
[1] 10.2
```

Similarly, the median is easily verified by using the `median()` function:

```r
median(c(8,10,9,12,12))
```

```
[1] 10
```

We can use the `table()` function to calculate frequencies and easily identify the mode.

```r
table(c(8,10,9,12,12))
```

```

 8  9 10 12 
 1  1  1  2 
```

2. The mean is -2.67, the median is -3.5, the mode is -4.

These mean is verified in R:

```r
mean(c(-4,0,-6,1,-3,-4))
```

```
[1] -2.666667
```

The median in R:

```r
median(c(-4,0,-6,1,-3,-4))
```

```
[1] -3.5
```

Finally, the mode in R:

```r
table(c(-4,0,-6,1,-3,-4))
```

```

-6 -4 -3  0  1 
 1  2  1  1  1 
```

3. The mean is 18.33, the median is 20, the data is bimodal with both 15 and 20 being modes.

These mean is verified in R:

```r
mean(c(20,15,25,20,10,15,25,20,15))
```

```
[1] 18.33333
```

The median in R:

```r
median(c(20,15,25,20,10,15,25,20,15))
```

```
[1] 20
```

The frequency distribution identifies the modes:

```r
table(c(20,15,25,20,10,15,25,20,15))
```

```

10 15 20 25 
 1  3  3  2 
```

Exercise 2

1. The mean price for Country Hill is 1.87. The mean price for Minute Maid is 2.09.

The means can be easily found with the `mean()` function:

```r
library(ISLR2)
mean(OJ$PriceCH)
```

```
[1] 1.867421
```

```r
mean(OJ$PriceMM)
```

```
[1] 2.085411
```

2. The mean price at store 1 for Country Hill is 1.80 vs. 1.84 for store 2. The juice is cheaper at store 1.

The means for each store can be found by using indexing and a logical statement. The Country Hill mean price at store 1 is given by:

```r
mean(OJ$PriceCH[OJ$StoreID==1])
```

```
[1] 1.803758
```

The Country Hill mean price at store 2 is given by:

```r
mean(OJ$PriceCH[OJ$StoreID==2])
```

```
[1] 1.841216
```

3. Purchasers of Country Hill at store 1 paid and average of 1.80 for Country Hill juice. At store 2 they paid 1.86. Once again the average price was lower at store 1.

The mean for Country Hill purchasers at store 1 is given by:

```r
mean(OJ$PriceCH[OJ$StoreID==1 & OJ$Purchase=="CH"])
```

```
[1] 1.797176
```

The mean for Country Hill purchasers at store 2 is:

```r
mean(OJ$PriceCH[OJ$StoreID==2 & OJ$Purchase=="CH"])
```

```
[1] 1.857383
```

Exercise 3

1. The average price of sale is found by using the weighted average formula. $\frac{\sum w_i x_i}{\sum w_i}$ The weights (w_i) are given by the number of shares bought and the values (x_i) are the prices. The weighted average is 246.802.

In R you can create two vectors. One holds the share price and the other one the number of shares bought.

```
PricePerShare<-c(250.34,234.59,270.45)
NumberOfShares<-c(80,120,50)
```

Next, you can multiply the *PricePerShare* and *NumberOfShares* vectors to find the numerator and then use `sum()` function to find the denominator. The weighted average is:

```
(WeightedAverage<-
  sum(PricePerShare*NumberOfShares)/sum(NumberOfShares))
```

```
[1] 246.802
```

2. The average if equal shares were bought would be 251.7933.

In R you can use the `mean()` function on the PricePerShare vector.

```
(Average<-mean(PricePerShare))
```

```
[1] 251.7933
```

Exercise 4

1. The inflation rate for each period is shown in the table below:

30%	23.08%	12.5%	16.67%

In R create an object to store the values of the CPI:

```
CPI<-c(1,1.3,1.6,1.8,2.1)
```

Next use the `diff()` function to find the difference between the end value and start value. Divide the result by a vector of starting value and multiply times 100.

```r
(Inflation<-100*diff(CPI)/CPI[1:4])
```

```
[1] 30.00000 23.07692 12.50000 16.66667
```

2. At the end of 4 years it is predicted that you would have 1621.17 dollars. Each year you would have gained 12.84% on average.

In R include the annual rates in a vector:

```r
growth<-c(0.173,0.196,0.068,0.082)
```

The arithmetic mean is:

```r
100*mean(growth)
```

```
[1] 12.975
```

The geometric mean is:

```r
(geom<-((prod(1+growth))^(1/4)-1)*100)
```

```
[1] 12.8384
```

At the end of the four years we would have:

```r
1000*(1+geom/100)^4
```

```
[1] 1621.167
```

References

Grolemund, Garret. 2014. “Hands-on Programming with r.” https://jjallaire.github.io/hopr/.
Wickham, Hadley. 2017. “R for Data Science.” https://r4ds.hadley.nz.