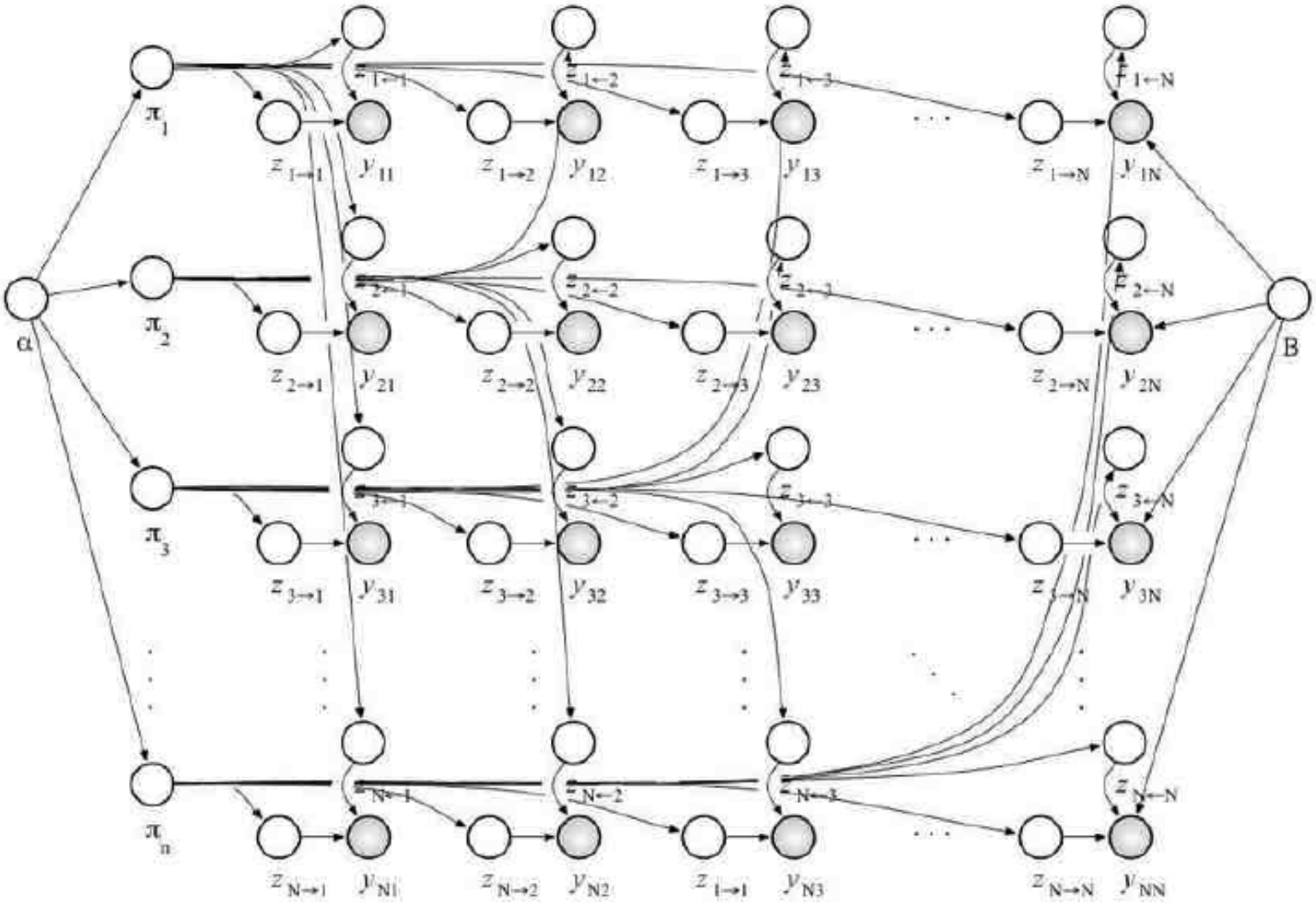


Applied Statistics and Data Science

Author: Srikar Katta

Contents

1 R Programming **2**

1.1 Introduction . . . 2
1.2 Variables and Data Types . . . 3
1.3 Conditional Logic . . . 10
1.4 Functions . . . 11
1.5 Loops . . . 15
1.6 Visualizations with ggplot2 . . . 25

2 Math Foundations **29**

2.1 Sets, Functions, and Notation . . . 29
2.2 Rates of Change and Derivatives . . . 33
2.3 Sums and Integration . . . 42

3 Probability **47**

3.1 Probability Basics . . . 47
3.2 Random Variables and Probability Distributions . . . 50
3.3 Named Distributions . . . 54
3.4 Expectation and Variance Operators . . . 62
3.5 Approximating Expectation and Variance . . . 62

4 Probabilistic Modeling: The Language of Data Science **63**

4.1 Empirical and Theoretical Distributions . . . 63
4.2 Likelihood . . . 67

Chapter 1 R Programming

1.1 Introduction

The rise of data science's popularity is in large part due to the easy use of computational techniques to simplify processes. What used to take five hours by hand can now be done in a matter of only a few seconds through the use of a computer. Throughout this book, we will use the statistical programming language R to complement the statistical concepts taught. While there are many languages that can be used for data science–including Python, Java, and Matlab–R is an easy-to-use programming language that is constantly growing, largely due to the fact that it is open-source (i.e., anyone can contribute to R's tools and functionalities). R has an incredibly strong community and is often used in scientific research and in the industry. Because R is quite expansive and has many features, it is quite difficult to understand all of R in one book. However, the content we discuss provides the reader enough information to understand the core aspects of R for statistics.

Before we dive into R, there are a few steps to tackle. First, we must install R and the R editor RStudio on our local computers by following the instructions on the RStudio website or by making an account with RStudio Cloud, the cloud version ofRStudio.

Within RStudio, there are four windows: the console tab, the workspace tab, the files tab, and the R scripts tab.

The console tab will display outputs; the workspace tab will display any variables or data created in any R scripts; the files tab will display the files in our current working directory, and the R scripts tab will open up files that we write R code in.

To open a new R script, press "File" –> "New File"–> "R Script". R scripts have the .R file extension and execute R commands. However, there are many times we want to add content to our scripts and do not want to have the code execute; rather than deleting the code, we "comment" the code out by putting a # in front of the line we do not want executed. If we are interested in commenting out multiple lines of code, we can use the RStudio shortcut, command + shift + c on Macs or control + shift + c on Linux and Windows. Comments are also incredibly useful for writing descriptions about a chunk of code's functions, allowing others (and ourselves) to quickly understand our goal when programming.

To ensure RStudio and R work, let us create our first R script that prints the statement "Hello, world!" Open a new R script and save it as \texttt{hello_world.R}. In order to get into the habit of proper coding practices, we want to add our name, date, and a short description of the script at the top as comments. For example, here is what I would write:

```
# Srikar Katta
# August 23, 2021
# My first R script
```

Now, write the statement print('Hello, world!'). While we will explain what character data types

and `print` statements are later in this chapter, at a high level, this command simply tells the computer to print out the statement inside its parentheses. In addition to the actual command, let us write a comment that explains what is happening so that we can get into the practice of describing our code:

```
# prints the statement "Hello, world!"
print('Hello, world!')
```

```
## [1] "Hello, world!"
```

To actually execute the command, we will highlight the chunk of code we are interested in running and hit `Run` in the top right of the `R` script tab or press `command + enter` on Macs or `control + enter` on Windows/Linux. In the console tab, we should see the following:

```
> print('Hello, world!')
```

```
[1] "Hello, world!"
```

1.2 Variables and Data Types

While `R` will simplify your workflow, there is a large trade-off that must be made: as the programmer, you must be very precise and very exact in your instructions. For instance, when explaining how to make a sandwich to a friend, you might say, "Take two slices of bread, put your fixens on one slice, and put the other slice on top of the fixes," and your friend would understand. However, the computer requires incredibly exact instructions. It has no idea what bread, fixens, or slices are; it does not understand what it means to "put" something on something else. Specificity is key when programming.

One frustration may be the fact that redefining "bread" for each and every instruction could be very tedious and repetitive. To overcome this, `R` (as well as other programming languages) has variables.

Definition 1.1. Variable

Variables *are spaces of memory reserved in the computer for storing specific values. The process of saving a value to a variable is known as* ***assignment****.* ♣

For instance, one variable may store the value 5 and another may store the value 6; these two variables can then be added up rather than writing "5 + 6." To assign a variable in `R`, the common convention is to write the name of the variable on the left, then "<-", and lastly the value that should be stored in the variable. For example, the following assigns the value 5 to the variable `x`:

```
x <- 5
```

Variables in `R` must follow certain naming conventions:

- The variable *must* start with a letter, but it can contain numbers, letters, underscore, and periods
- Keywords are not allowed to be defined as variable names. For example, 'for' is a keyword in `R`, so you are not allowed to save a value to the variable 'for'.
- Special characters like '¡ and '*' and white space like tabs and spaces are not allowed in the variable name

Because variables reserve an allocated amount of memory, the variable must know how much memory is required for a value, and depending on the data type, each value requires a different amount of data. While we will not discuss the exact specifics of memory allocation or even memory requirements for different data types, understanding the differences between data types is essential for a programmer.

Definition 1.2. Data type

*A **data type** is a piece of information that informs the computer of properties that can be act upon it. For instance, integer data types can be added together but character data types cannot.* ♣

The key data types for data science are logical, numeric, integer, and character.

Definition 1.3. Logical data type

*A **logical data type** has only two values: `TRUE` and `FALSE`. In R, `TRUE` and 1 are equivalent and `FALSE` and 0 are equivalent.* ♣

Definition 1.4. Numeric data type

*A **numeric data type** is any real number. The following are examples: 1, -5, 10284.99678, -901374.112. There are also keywords that represent special numeric data, like `pi` represents the number 3.14..., and `Inf` represents ∞.* ♣

Definition 1.5. Character data type

*A **character data type** is anything that resides between quotes. For example, even though 1 is a numeric, when it is stored within a pair of quotes, it becomes a character: "1". Character data can be denoted with either single (i.e., ') or double (i.e., ") quotes.* ♣

Each data type has special operations that can be performed on them. For instance, logical values have logical operators–and, or, and xor–denoted as `&`, `|`, and `xor` in R respectively. The following table describes the result of the logical operators on logical data types:

Logical Value 1	Logical Value 2	`&` outcome	`|` outcome	`xor` outcome
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

Additionally, we can apply the note operator–denoted as `!`–to logical data. This will turn a `TRUE` value to `FALSE` and vice versa.

All numeric data can be added, subtracted, multiplied, divided (unless the denominator is 0), exponentiated. The operations are shown below:

```
x <- 5.5
y <- 11

add <- x + y # use + for addition operator
sub <- x - y # use - for subtraction operator
mul <- x * y # use * for multiplication operator
```

```
div <- y / x # use / for division operator
xpo <- x ^ y # use ^ for exponentiation operator
```

In addition to operators existing for specific data types, there are some operations that exist for numerics and characters both. These generally deal with data being compared. While it may seem odd that `y` and `z` are both characters and the less than and greater than operators are being used, recall earlier that we said all data are stored as sequences of 0s and 1s in `R`. Since 0s and 1s are numeric data, they can be compared logically. While applying the greater than or less than operators on characters may not be as common as comparing numeric data in data science, it is still helpful to know that these operators exist. Essentially, the rule for characters is that special characters (e.g., *) are "smaller" than lower case letters, which are "smaller" than upper case letters. These comparisons will return logical data types (i.e., `TRUE` if the result is true and `FALSE` if the result is false).

```
x <- 5
y <- 'a'
z <- 'b'

less <- x < y # use < for less than operator
grtr <- x > y # use <= for less than or equal to operator
lseq <- x <= y # use > for greater than operator
greq <- x >= y # use >= for greater than or equal to operator
equl <- x == y # use == for equal to operator
nteq <- x != y # use != for the not equal to operator
```

While these "scalar" data types are useful, there are many instances in data science when it is useful to couple many different values together. For instance, suppose we have a sequence of 10,000 product prices off of an online marketplace. It is likely that we will apply the same operations to all 10,000 data, so it makes sense to have a data type to save all the values together. In `R`, multiple values can be saved in a vector.

Definition 1.6. Vector data type

*A **vector** is a data type in which multiple values can be stored together. These are denoted by `c([insert data here])` with commas separating values.* ♣

It is important to note that vectors do not have to consist of the same data types. Suppose we are trying to capture information about products in an online marketplace. While product prices will be numeric, the product's name or brand or supplier will most likely be character data types; using vectors, it is possible to store these data together. It is also possible to combine vectors together by creating a new vector where the two elements are the old vector. Additionally, we may be interested in accessing a specific value of the vector, which we can do by indexing the vector; this is accomplished by calling the variable storing the list, placing square brackets after the variable name, and placing the position of the element of interest inside the square brackets. It is important to note that counting starts from one in `R`. So the first element in the vector will be called using the index 1.

```
vector1 <- c('apple', 1, 'red')
vector2 <- c('banana', 0.5, 'yellow')

c(vector1, vector2) # displays vector1 and vector2 together
```

```
## [1] "apple"  "1"      "red"    "banana" "0.5"    "yellow"
```

```
vector2[2] # accesses the second element of vector2: 0.5
```

```
## [1] "0.5"
```

One limitation of vectors however is that they can only store one dimensional data. In the previous example, while we may have been able to intuit that `'apple'` represents the product, it may be better to explicitly store what each value represents, which is possible with a list. We can access data within a list by either providing the index of element of interest or by writing the name of the element of interest as a character if the name is provided. We can also access named elements by writing '[insert name of list variable]$[insert name of data in the vector]'.

Definition 1.7. List data type

*A **list** is a data type in which multiple values can be stored together and the values can be named. Lists can also store other lists or vectors inside of them. These are denoted by `list([insert data here])` with commas separating values. To name the values of a list, the name of the data is followed by `=` and then the data to be stored: `list([insert name] = [insert data], [insert name] = [insert data], ...)`.* ♣

```
list1 <- list(fruit = 'apple', price = 1, color = 'red')
list2 <- list(fruit = 'banana', price = 0.5, color = 'yellow')

list1 # displays list 1
```

```
## $fruit
## [1] "apple"
##
## $price
## [1] 1
##
## $color
## [1] "red"
```

```
list(list1, list2) # displays list1 and list2 together
```

```
## [[1]]
## [[1]]$fruit
## [1] "apple"
##
## [[1]]$price
## [1] 1
```

```
##
## [[1]]$color
## [1] "red"
##
##
## [[2]]
## [[2]]$fruit
## [1] "banana"
##
## [[2]]$price
## [1] 0.5
##
## [[2]]$color
## [1] "yellow"
```

```
list1['fruit'] # accesses the 'fruit' element of list1
```

```
## $fruit
## [1] "apple"
```

```
list2[2] # accesses the second element of list2: 0.5
```

```
## $price
## [1] 0.5
```

```
list2$color
```

```
## [1] "yellow"
```

There are also many cases in which a data scientist may be interested in representing multidimensional data more succinctly. For example, `list1` and `list2` both have the same named data, so it may be better to store the data as a matrix instead, a two-dimensional data type. Generally, matrix columns represent the same value of interest (e.g., price) while rows represent data for each observation (e.g., data on each product). Data can be accessed in matrixes by specifying the row of the element of interest followed by the column of the element of interest within square brackets, similar to a vector. If the matrix has names, the data can be accessed using the names of the row and column the element is located in. It is also possible for a matrix to access more than one element through a single call by providing a vector of row indexes and column indexes that are both aligned to the elements of interest.

Definition 1.8. Matrix data type

*A **matrix** is a data type that has both rows and columns, with entries of all the same data type. Unlike the other data types, matrices take in more information; specifically, they require a vector of data, the number of rows, the number of columns, and the manner in which the vector should be reshaped. They can be formed by calling* `matrix(data = [insert vector], nrow = [insert integer representing number of rows], ncol = [inser integer representing number of columns], byrow = [insert logical representing whether the matrix should be formed rowise or not], dimnames`

= [insert list of length two gicing the row and column names respectively or NULL if not necessary]. ♣

```
vector1 <- c('apple', 1, 'red')
vector2 <- c('banana', 0.5, 'yellow')
# make matrix row-wise
m1 <- matrix(data = c(vector1, vector2),
       nrow = 2,
       ncol = 3,
       byrow = TRUE,
       dimnames = NULL)
# make matrix column-wise
m2 <- matrix(data = c(vector1, vector2),
       nrow = 3,
       ncol = 2,
       byrow = FALSE,
       dimnames = NULL)
# make matrix row-wise with names
m3 <- matrix(data = c(vector1, vector2),
       nrow = 2,
       ncol = 3,
       byrow = TRUE,
       dimnames = list(c('product1', 'product2'),
                       c('product', 'price', 'color')))
m1
```

```
##      [,1]     [,2]  [,3]
## [1,] "apple"  "1"   "red"
## [2,] "banana" "0.5" "yellow"
```

```
m2
```

```
##      [,1]    [,2]
## [1,] "apple" "banana"
## [2,] "1"     "0.5"
## [3,] "red"   "yellow"
```

```
m3
```

```
##          product  price color
## product1 "apple"  "1"   "red"
## product2 "banana" "0.5" "yellow"
```

```
# access data in m1
m1[1, 3] # row 1, column 3 is 'yellow'
```

```
## [1] "red"
```

```
m3['product2', 'product'] # 'banana'
```

```
## [1] "banana"
```

```
m2[c(1, 2), c(1, 2)]
```

```
##      [,1]    [,2]
## [1,] "apple" "banana"
## [2,] "1"     "0.5"
```

If a function or operation can be applied to all values in a vector, list, or matrix, then an operation can be applied by treating the variable just like another variable. For example, if we have a vector `x` of numeric data, then we could add 1 to all the values in `x` by calling `x + 1`:

```
vector1 <- c(1.5, 2, 2.5)
vector1 + 1
```

```
## [1] 2.5 3.0 3.5
```

However, if all the values are not of the same data type a function cannot be applied to that data, then there will be an error:

A dataframe data type is very similar to a matrix in the sense that it is two dimensional and has rows and columns. However, dataframes are a generalized form of a matrix and allows for different data types: each column is a vector of values of the same data types but the rows can differ. For example, notice that even though 1 in `R` is a numeric, it is a character in the matrix `m1`, as evidenced by the quotes around its entry. However, in the `dataframe`, 1 is a numeric.

\begin{defintion}{Dataframe data type}{} A **dataframe** is a generalized matrix whose columns may be of different data types. \end{definition}

`data.frame` constructions take the following form:

```
[insert variable name] <- data.frame(
  [insert column name] = [insert vector of data],
  [insert column name] = [insert vector of data],
  ...
  [insert column name] = [insert vector of data]
)
```

Because the columns are vectors of the same data type, we must first create vectors representing data we want in our columns; keep in mind that all vectors must be the same length or else `R` will throw an error.

```
vector1 <- c('apple', 'banana')
vector2 <- c(1, 0.5)
vector3 <- c('red', 'yellow')

df1 <- data.frame(
```

```
  product = vector1,
  price = vector2,
  color = vector3
)

print(df1)
```

```
##   product price  color
## 1   apple   1.0    red
## 2  banana   0.5 yellow
```

1.3 Conditional Logic

In addition to variables and data types, conditional statements are essential for data science. Conditionals often follow the form of "if. . . then." For instance, "if a person is from Philadelphia, then they are likely fans of the Eagles," would be a conditional statement because it follows an "if. . . then" style statement.

Definition 1.9. Conditional

*A **conditional** is a statement that follows the form of "if [insert condition], then [insert conclusion]". The condition must be a logical type: `TRUE` or `FALSE`.*

♣

Suppose we are creating a simulation of the real estate market, which vary significantly by geographic location: real estate in Manhattan is different than real estate in Boston, which is very different than real estate in Wichita, Kansas. These conditional statements can be very helpful in these situations. The "if" portion of the conditional will be denoted by `if([insert logical statement])` in `R`. Within the `if` statement must go a logical statement (i.e., a statement that returns `TRUE` or `FALSE`). The "if" portion is then followed by the conclusion, which is contained within brackets. To create multi-level conditionals, we use the `else` operator.

Note that it is possible for for two conditional to be true; in the real estate market, a house will be located in Boston and the northeast. However, if both conditions are used in the same if-else logic, then only the first true condition will be considered. In other words, as soon as the first true condition occurs, the computer stops evaluating all other else statements.

```
location <- 'Boston'

# simple conditional statement
if(location == 'Boston') {
  mean_house_price <- 800000
}

mean_house_price
```

```
## [1] 8e+05
```

```
# if-else conditional statement
if(location != 'Boston') {
  mean_house_price <- 300000
} else {
  mean_house_price <- 800000
}
mean_house_price
```

```
## [1] 8e+05
```

```
# multi-level if-else conditional statement
if(location == 'Boston') {
  mean_house_price <- 800000
} else if(location == 'Manhattan') {
  mean_house_price <- 1500000
} else if(location == 'Northeast') {
  mean_house_price <- 500000
} else {
  mean_house_price <- 250000
}
mean_house_price
```

```
## [1] 8e+05
```

1.4 Functions

Returning to the example in the introduction where we discussed "making a sandwich," variables allow you to define what "bread" and "fixens" are; however, the computer still may not know what it means to "put" fixens on bread. Functions are actions in computer programming that allow us to overcome the problems of repeating ourselves; rather than continuously telling the computer what "put" means, we can define a function to perform the action for us.

Definition 1.10. Functions in programming

Functions *are blocks of codes that perform actions in programming that take in inputs and provide zero or one outputs.* ♣

Functions follow a particular construction: functions take in inputs, called arguments, that can be data structures that the function acts upon (e.g., take the mean of a vector) or it may be a way to alter the method in which a function acts (e.g., return a different mean for "Northeast" and "Boston" inputs). However, functions can also have zero inputs and always perform the same action (i.e., execute the same block of code) by calling the function. It is important to note that arguments often require a specific data structure, or else the code will throw an error or not execute properly. Ensuring the proper data type is being input can save us from headaches down the road.

Additionally, functions in programming return either one or zero data structures that can then be saved to other variables. However, if we would like to return two values–the mean and the variance, for example–we could save the results in a named list and access the values by returning the list.

The following are a few examples of commonly used functions in R:

```
# print function
#   input: character
#   returns: nothing
#   action: displays character to screen
print('Hello, world')
```

```
## [1] "Hello, world"
```

```
# mean function
#   input: vector of numeric data
#   output: numeric
#   action: calculates the mean of the numeric data
mean(c(1, 2, 3))
```

```
## [1] 2
```

```
mean('Hello, world') # using the wrong input data type can lead to errors
```

```
## Warning in mean.default("Hello, world"): argument is not numeric or logical:
## returning NA
```

```
## [1] NA
```

If using RStudio, we can find information on functions by typing `?[insert function name]`, which will then prompt the "Help" tab to display information on the function.

Functions can also have default arguments. For instance, in the `mean` function, it actually takes in multiple arguments: a vector of numerics `x`, a numeric scalar `trim` that represents "the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint," and a logical `na.rm` that "[indicates] whether NA values should be stripped before the computation proceeds." However, we did not need to specify the `trim` and `na.rm` arguments when previously calling mean because they have defaults `trim = 0` and `na.rm = FALSE`. The following displays the differences in calling the `mean` function when changing defaults:

```
vector1 <- c(1, 2, 2, 4)
print(mean(vector1)) # mean with trim = 0 default
```

```
## [1] 2.25
```

```
print(mean(vector1, trim = 0.25)) # mean when removing top and bottom 25% of values
```

```
## [1] 2
```

We can also create our own functions by calling

```
[insert name] <- function([insert arguments], [insert default argument] = [insert default input])
  [insert code]

  return([insert data structure to return])
}
```

As we discussed earlier, functions do not need to take in any arguments or return any values. So, we can define a function with neither of these as such:

```
[insert name] <- function() {
  [insert code]
}
```

Additionally, function names follow the same rules/conventions as variable names:

- The variable *must* start with a letter, but it can contain numbers, letters, underscore, and periods
- Keywords are not allowed to be defined as variable names. For example, 'for' is a keyword in R, so you are not allowed to save a value to the variable 'for'.
- Special characters like '¡ and '*' and white space like tabs and spaces are not allowed in the variable name

It is very common to see functions within functions as well. Suppose we will be printing out three lines: the first statement being a person's name, the second being a person's age, and the third being the number of classes a person has taken. For the letter grades, we will ask how many classes did they receive with that letter grade. We can do so by defining the function \texttt{print_person}:

```
print_person <- function(name, age, n_classes) {
  print(paste('Name: ', name))
  print(paste('Age: ', age))
  print(paste('Number of classes: ', n_classes))
}
print_person(
  name = 'Srikar',
  age = '21',
  n_classes = 25
)
```

```
## [1] "Name:  Srikar"
## [1] "Age:  21"
## [1] "Number of classes:  25"
```

Now, we can extend this to also print out someone's GPA. However, in order to do this, we need someone's letter grades, which we will obtain by creating a new function called \texttt{print_person_plus_gpa} that also has parameters \texttt{a_grade, b_grade, c_grade, d_grade, f_grade} that each take in a numeric that details the number of classes a person has earned that letter grade in. First, we need to convert the \texttt{a_grade, b_grade, c_grade, d_grade, f_grade} numerics into GPA points: each A is worth 4 points, each B is worth 3, each C is worth 2, each D is worth 1, and Fs are worth 0. We then divide the GPA points

by the number of classes taken

```
calc_gpa <- function(a_grade, b_grade, c_grade, d_grade, f_grade) {
  gpa_points <-
    (a_grade * 4) + (b_grade * 3) + (c_grade * 2) + (d_grade * 1) + (f_grade * 0)
  n_classes <- sum(c(a_grade, b_grade, c_grade, d_grade, f_grade))
  gpa <- gpa_points/n_classes
  return(gpa)
}
```

Then, we can put the \texttt{print_person} and \texttt{print_gpa} statements together.

```
print_person_plus <- function(name, age, a_grade, b_grade, c_grade, d_grade, f_grade) {

  # calculate GPA
  gpa <- calc_gpa(a_grade = a_grade,
                  b_grade = b_grade,
                  c_grade = c_grade,
                  d_grade = d_grade,
                  f_grade = f_grade)

  # calculate number of classes taken
  n_classes <- sum(c(a_grade, b_grade, c_grade, d_grade, f_grade))

  # print information
  print_person(name = name,
               age = age,
               n_classes = n_classes)
  print(paste('GPA: ', gpa))
}

print_person_plus(name = 'Srikar',
                  age = 21,
                  a_grade = 25,
                  b_grade = 0,
                  c_grade = 0,
                  d_grade = 0,
                  f_grade = 0
                  )
```

```
## [1] "Name:  Srikar"
## [1] "Age:  21"
## [1] "Number of classes:  25"
## [1] "GPA:  4"
```

1.5 Loops

Functions are very useful for modularizing code and removing redundancies. In a similar vein, loops allow us to remove repetition in our code by executing the same block of code multiple times.

Definition 1.11. Loops in programming

*A **loop** is a control sequence that runs a block of code multiple times.* ♣

To illustrate its use case, suppose we were asked to find the GPA of 25 students. While we have a matrix with each row representing the following data for an individual student: \texttt{name, age, a_grade, b_grade, c_grade, d_grade, f_grade}, where each column name represents the same value as its corresponding parameters in the \texttt{print_person_plus} function.

```
##        names ages a_grades b_grades c_grades d_grades f_grades
## 1       Liam   27        9       12       20        2        7
## 2     Olivia   19        3       17       20        4        6
## 3      Kanye   18       10       12       11        0       17
## 4       Noah   25       11       15       20        3        1
## 5      Rohan   31       14       14       15        4        3
## 6       Emma   18       11       12       11        5       11
## 7     Oliver   20       10       16       18        7        0
## 8        Ava   23       11       14        9        5       11
## 9     Elijah   23       10       11       14        5       10
## 10 Charlotte   22        5       15       14        2       14
## 11   William   20       12        6       14        6       12
## 12    Sophia   27        9       17       25        4        0
## 13     James   26        8        8       17        3       14
## 14       Jun   20        9       10       14        4       13
## 15    Amelia   15       14       11       15        6        4
## 16  Benjamin   21       10       15       21        5        0
## 17  Isabella   15        9       20       15        4        2
## 18     Lucas   19       11       18       17        6        0
## 19       Mia   26        7       11       15        5       12
## 20    Tashin   28        9       13       17        2        9
## 21     Henry   15       13       12       12        3       10
## 22    Evelyn   25        7       13       12        9        9
## 23    Miriam   17       13       15       12        5        5
## 24 Alexander   18       11       16       23        9        0
## 25    Harper   14        5       14       25        7        0
```

While the \texttt{print_person_plus} function allows us to avoid retyping all of the code multiple times, we still have to write the \texttt{print_person_plus} function each and every time. For example,

```
# info for person 1
print_person_plus(name = grades_table$names[1],
```

```
                  age = grades_table$ages[1],
                  a_grade = grades_table$a_grades[1],
                  b_grade = grades_table$b_grades[1],
                  c_grade = grades_table$c_grades[1],
                  d_grade = grades_table$d_grades[1],
                  f_grade = grades_table$f_grades[1]
                  )
```

```
## [1] "Name:  Liam"
## [1] "Age:  27"
## [1] "Number of classes:  50"
## [1] "GPA:  2.28"
```

```
# info for person 2
print_person_plus(name = grades_table$names[2],
                  age = grades_table$ages[2],
                  a_grade = grades_table$a_grades[2],
                  b_grade = grades_table$b_grades[2],
                  c_grade = grades_table$c_grades[2],
                  d_grade = grades_table$d_grades[2],
                  f_grade = grades_table$f_grades[2]
                  )
```

```
## [1] "Name:  Olivia"
## [1] "Age:  19"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
```

Instead we can use a for loop, which we usually use to iterate over a sequence of some sort; the loop stops executing after it has iterated through all the elements in the sequence.

Definition 1.12. For loop in programming

*A **for loop** is a control sequence that iterates through a vector of elements (generally a sequence) and stops once ever element in the vector has been iterated through. The syntax is usually* `for([insert variable name] in [insert vector]) { [insert code] }` ♣

In this case, we will create a sequence from 1 to 25 so that we can go through each and every observation in the dataframe, using `i` as our indexing variable.

```
for(i in 1:25) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
```

Next, we will replace the inner code chunk with the function of interest:

```
for(i in 1:25) {
  print_person_plus(name = grades_table$names[i],
                    age = grades_table$ages[i],
                    a_grade = grades_table$a_grades[i],
                    b_grade = grades_table$b_grades[i],
                    c_grade = grades_table$c_grades[i],
                    d_grade = grades_table$d_grades[i],
                    f_grade = grades_table$f_grades[i]
                    )
}
```

```
## [1] "Name:  Liam"
## [1] "Age:  27"
## [1] "Number of classes:  50"
## [1] "GPA:  2.28"
## [1] "Name:  Olivia"
## [1] "Age:  19"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
## [1] "Name:  Kanye"
```

```
## [1] "Age:  18"
## [1] "Number of classes:  50"
## [1] "GPA:  1.96"
## [1] "Name:  Noah"
## [1] "Age:  25"
## [1] "Number of classes:  50"
## [1] "GPA:  2.64"
## [1] "Name:  Rohan"
## [1] "Age:  31"
## [1] "Number of classes:  50"
## [1] "GPA:  2.64"
## [1] "Name:  Emma"
## [1] "Age:  18"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
## [1] "Name:  Oliver"
## [1] "Age:  20"
## [1] "Number of classes:  51"
## [1] "GPA:  2.56862745098039"
## [1] "Name:  Ava"
## [1] "Age:  23"
## [1] "Number of classes:  50"
## [1] "GPA:  2.18"
## [1] "Name:  Elijah"
## [1] "Age:  23"
## [1] "Number of classes:  50"
## [1] "GPA:  2.12"
## [1] "Name:  Charlotte"
## [1] "Age:  22"
## [1] "Number of classes:  50"
## [1] "GPA:  1.9"
## [1] "Name:  William"
## [1] "Age:  20"
## [1] "Number of classes:  50"
## [1] "GPA:  2"
## [1] "Name:  Sophia"
## [1] "Age:  27"
## [1] "Number of classes:  55"
## [1] "GPA:  2.56363636363636"
## [1] "Name:  James"
## [1] "Age:  26"
## [1] "Number of classes:  50"
## [1] "GPA:  1.86"
```

```
## [1] "Name:  Jun"
## [1] "Age:  20"
## [1] "Number of classes:  50"
## [1] "GPA:  1.96"
## [1] "Name:  Amelia"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.5"
## [1] "Name:  Benjamin"
## [1] "Age:  21"
## [1] "Number of classes:  51"
## [1] "GPA:  2.58823529411765"
## [1] "Name:  Isabella"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.6"
## [1] "Name:  Lucas"
## [1] "Age:  19"
## [1] "Number of classes:  52"
## [1] "GPA:  2.65384615384615"
## [1] "Name:  Mia"
## [1] "Age:  26"
## [1] "Number of classes:  50"
## [1] "GPA:  1.92"
## [1] "Name:  Tashin"
## [1] "Age:  28"
## [1] "Number of classes:  50"
## [1] "GPA:  2.22"
## [1] "Name:  Henry"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.3"
## [1] "Name:  Evelyn"
## [1] "Age:  25"
## [1] "Number of classes:  50"
## [1] "GPA:  2"
## [1] "Name:  Miriam"
## [1] "Age:  17"
## [1] "Number of classes:  50"
## [1] "GPA:  2.52"
## [1] "Name:  Alexander"
## [1] "Age:  18"
## [1] "Number of classes:  59"
```

```
## [1] "GPA:  2.49152542372881"
## [1] "Name:  Harper"
## [1] "Age:  14"
## [1] "Number of classes:  51"
## [1] "GPA:  2.33333333333333"
```

Suppose that instead of indexing until 25, we indexed until 26. Then, we would get the following output:

```
for(i in 1:26) {
  print_person_plus(name = grades_table$names[i],
                    age = grades_table$ages[i],
                    a_grade = grades_table$a_grades[i],
                    b_grade = grades_table$b_grades[i],
                    c_grade = grades_table$c_grades[i],
                    d_grade = grades_table$d_grades[i],
                    f_grade = grades_table$f_grades[i]
                    )
}
```

```
## [1] "Name:  Liam"
## [1] "Age:  27"
## [1] "Number of classes:  50"
## [1] "GPA:  2.28"
## [1] "Name:  Olivia"
## [1] "Age:  19"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
## [1] "Name:  Kanye"
## [1] "Age:  18"
## [1] "Number of classes:  50"
## [1] "GPA:  1.96"
## [1] "Name:  Noah"
## [1] "Age:  25"
## [1] "Number of classes:  50"
## [1] "GPA:  2.64"
## [1] "Name:  Rohan"
## [1] "Age:  31"
## [1] "Number of classes:  50"
## [1] "GPA:  2.64"
## [1] "Name:  Emma"
## [1] "Age:  18"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
## [1] "Name:  Oliver"
## [1] "Age:  20"
```

```
## [1] "Number of classes:  51"
## [1] "GPA:  2.56862745098039"
## [1] "Name:  Ava"
## [1] "Age:  23"
## [1] "Number of classes:  50"
## [1] "GPA:  2.18"
## [1] "Name:  Elijah"
## [1] "Age:  23"
## [1] "Number of classes:  50"
## [1] "GPA:  2.12"
## [1] "Name:  Charlotte"
## [1] "Age:  22"
## [1] "Number of classes:  50"
## [1] "GPA:  1.9"
## [1] "Name:  William"
## [1] "Age:  20"
## [1] "Number of classes:  50"
## [1] "GPA:  2"
## [1] "Name:  Sophia"
## [1] "Age:  27"
## [1] "Number of classes:  55"
## [1] "GPA:  2.56363636363636"
## [1] "Name:  James"
## [1] "Age:  26"
## [1] "Number of classes:  50"
## [1] "GPA:  1.86"
## [1] "Name:  Jun"
## [1] "Age:  20"
## [1] "Number of classes:  50"
## [1] "GPA:  1.96"
## [1] "Name:  Amelia"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.5"
## [1] "Name:  Benjamin"
## [1] "Age:  21"
## [1] "Number of classes:  51"
## [1] "GPA:  2.58823529411765"
## [1] "Name:  Isabella"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.6"
## [1] "Name:  Lucas"
```

```
## [1] "Age:  19"
## [1] "Number of classes:  52"
## [1] "GPA:  2.65384615384615"
## [1] "Name:  Mia"
## [1] "Age:  26"
## [1] "Number of classes:  50"
## [1] "GPA:  1.92"
## [1] "Name:  Tashin"
## [1] "Age:  28"
## [1] "Number of classes:  50"
## [1] "GPA:  2.22"
## [1] "Name:  Henry"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.3"
## [1] "Name:  Evelyn"
## [1] "Age:  25"
## [1] "Number of classes:  50"
## [1] "GPA:  2"
## [1] "Name:  Miriam"
## [1] "Age:  17"
## [1] "Number of classes:  50"
## [1] "GPA:  2.52"
## [1] "Name:  Alexander"
## [1] "Age:  18"
## [1] "Number of classes:  59"
## [1] "GPA:  2.49152542372881"
## [1] "Name:  Harper"
## [1] "Age:  14"
## [1] "Number of classes:  51"
## [1] "GPA:  2.33333333333333"
## [1] "Name:  NA"
## [1] "Age:  NA"
## [1] "Number of classes:  NA"
## [1] "GPA:  NA"
```

Notice that the code does not break; instead, it prints `NA` values, which would hurt our analyses. To overcome this issue, we will instead replace the "to" index with an automatic calculation of the number of rows in the dataframe, using the `nrow` function:

```
for(i in 1:nrow(grades_table)) {
  print_person_plus(name = grades_table$names[i],
                    age = grades_table$ages[i],
                    a_grade = grades_table$a_grades[i],
```

```
                    b_grade = grades_table$b_grades[i],
                    c_grade = grades_table$c_grades[i],
                    d_grade = grades_table$d_grades[i],
                    f_grade = grades_table$f_grades[i]
                    )
}
```

```
## [1] "Name:  Liam"
## [1] "Age:  27"
## [1] "Number of classes:  50"
## [1] "GPA:  2.28"
## [1] "Name:  Olivia"
## [1] "Age:  19"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
## [1] "Name:  Kanye"
## [1] "Age:  18"
## [1] "Number of classes:  50"
## [1] "GPA:  1.96"
## [1] "Name:  Noah"
## [1] "Age:  25"
## [1] "Number of classes:  50"
## [1] "GPA:  2.64"
## [1] "Name:  Rohan"
## [1] "Age:  31"
## [1] "Number of classes:  50"
## [1] "GPA:  2.64"
## [1] "Name:  Emma"
## [1] "Age:  18"
## [1] "Number of classes:  50"
## [1] "GPA:  2.14"
## [1] "Name:  Oliver"
## [1] "Age:  20"
## [1] "Number of classes:  51"
## [1] "GPA:  2.56862745098039"
## [1] "Name:  Ava"
## [1] "Age:  23"
## [1] "Number of classes:  50"
## [1] "GPA:  2.18"
## [1] "Name:  Elijah"
## [1] "Age:  23"
## [1] "Number of classes:  50"
## [1] "GPA:  2.12"
```

```
## [1] "Name:  Charlotte"
## [1] "Age:  22"
## [1] "Number of classes:  50"
## [1] "GPA:  1.9"
## [1] "Name:  William"
## [1] "Age:  20"
## [1] "Number of classes:  50"
## [1] "GPA:  2"
## [1] "Name:  Sophia"
## [1] "Age:  27"
## [1] "Number of classes:  55"
## [1] "GPA:  2.56363636363636"
## [1] "Name:  James"
## [1] "Age:  26"
## [1] "Number of classes:  50"
## [1] "GPA:  1.86"
## [1] "Name:  Jun"
## [1] "Age:  20"
## [1] "Number of classes:  50"
## [1] "GPA:  1.96"
## [1] "Name:  Amelia"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.5"
## [1] "Name:  Benjamin"
## [1] "Age:  21"
## [1] "Number of classes:  51"
## [1] "GPA:  2.58823529411765"
## [1] "Name:  Isabella"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.6"
## [1] "Name:  Lucas"
## [1] "Age:  19"
## [1] "Number of classes:  52"
## [1] "GPA:  2.65384615384615"
## [1] "Name:  Mia"
## [1] "Age:  26"
## [1] "Number of classes:  50"
## [1] "GPA:  1.92"
## [1] "Name:  Tashin"
## [1] "Age:  28"
## [1] "Number of classes:  50"
```

```
## [1] "GPA:  2.22"
## [1] "Name:  Henry"
## [1] "Age:  15"
## [1] "Number of classes:  50"
## [1] "GPA:  2.3"
## [1] "Name:  Evelyn"
## [1] "Age:  25"
## [1] "Number of classes:  50"
## [1] "GPA:  2"
## [1] "Name:  Miriam"
## [1] "Age:  17"
## [1] "Number of classes:  50"
## [1] "GPA:  2.52"
## [1] "Name:  Alexander"
## [1] "Age:  18"
## [1] "Number of classes:  59"
## [1] "GPA:  2.49152542372881"
## [1] "Name:  Harper"
## [1] "Age:  14"
## [1] "Number of classes:  51"
## [1] "GPA:  2.33333333333333"
```

Our code can now also account for a new 26th student that may join the dataset.

1.6 Visualizations with ggplot2

R has become one of the most popular languages for data science because of its strong community of independent developers, which means that many people contribute to developing R's functionality. R's extensions come from packages.

Definition 1.13. Packages in R

Packages *are collections of code, tests, data, objects, and documentation that are assembled in a specific format to allow for easy installation.* ♣

To install a package that is listed on R's public repository, CRAN, we simply have to type

```
# package name must be within quotes
install.packages('[insert package name]')
```

Once the package is installed, we can load it in by typing

```
# package name can have quotes or not -- it does not matter here
library([insert package name])
```

One of the most popular packages in R is the ggplot2 package, which creates great visualizations built

on the idea of the "Grammar of Graphics". First we will install ggplot2 and then load it into our R environment.

```
# package name must be within quotes
install.packages('ggplot2')
```

```
library(ggplot2)
```

While the ggplot2 syntax may be difficult when first starting, it will become second nature in the long run. Unlike other plotting libraries we may run into in R, ggplot2 takes in dataframes and then layers other features on top.

Let us first load in data to use from the ggplot2 library:

```
# load data
data('diamonds', package = 'ggplot2')
```

In the "Environment" tab of RStudio, we should now see an object called diamonds, the dataframe we will use. To explore what the columns represent, we can use ?diamonds because it is an object in the ggplot2 package and will have documentation for it.

```
# show the first five rows of the data
print(head(diamonds))
```

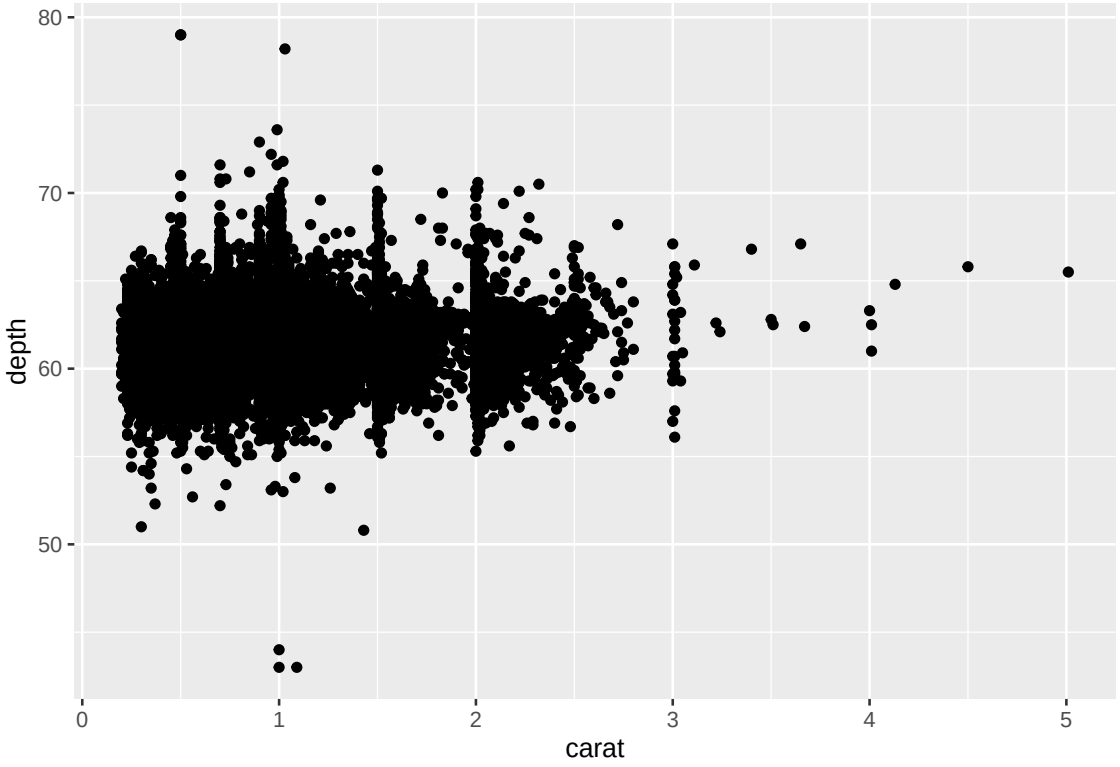
```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
# show details about the diamonds dataset in the Help tab
?diamonds
```

The first layer of the ggplot2 visualization specifies where the data comes from, which in this case is the diamonds dataset. The next layers decide what data we will display and how to display it. For instance, to make a bar plot, we use \texttt{geom_bar}; to make a scatter plot, we use \texttt{geom_point}. Within the layer function, we use the aes function to specify which vectors to map onto each axis.

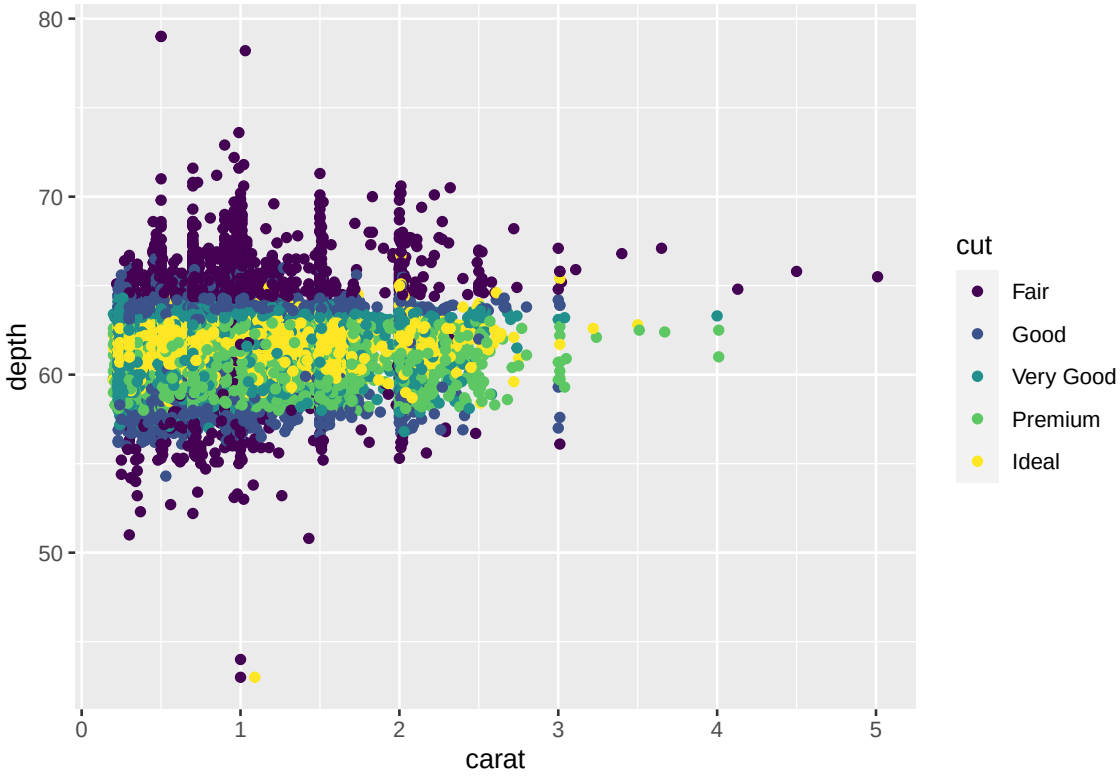
While plotting may seem complicated, perhaps an example will make more sense. Let us create a scatter plot with the number of carats on the x-axis and the depth on the y-axis:

```
ggplot(diamonds) +
  geom_point(aes(x = carat, y = depth))
```



Notice that for each carat and depth observation, there is a cut associated with those values; we can take advantage of this mapping and further customize our visualization by coloring our points to display the cut of each observations:

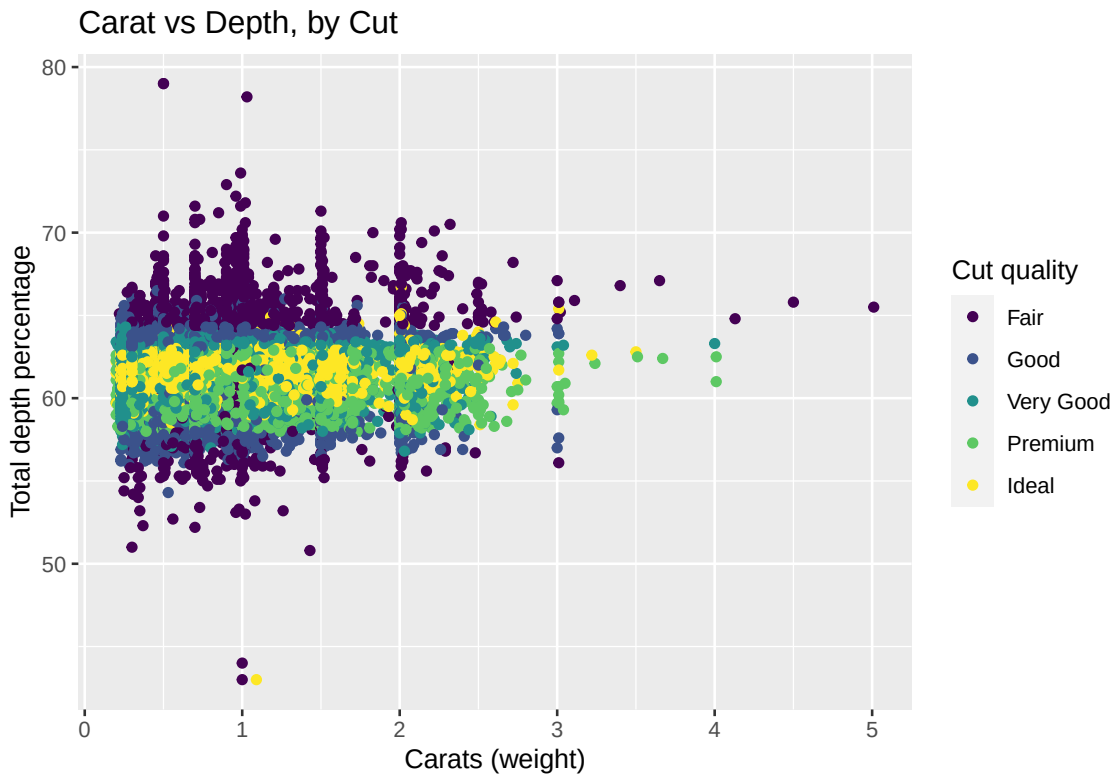
```
ggplot(diamonds) +
  geom_point(aes(x = carat, y = depth, color = cut))
```



We can also add titles using the `labs` layer. Each aesthetic in the `aes` function can be relabeled in the

`labs` function, as can the `title`, and the `subtitle`.

```
ggplot(diamonds) +
  geom_point(aes(x = carat, y = depth, color = cut)) +
  labs(title = 'Carat vs Depth, by Cut',
       x = 'Carats (weight)',
       y = 'Total depth percentage',
       color = 'Cut quality')
```



While `ggplot2` has many features, these basics should guide us through the rest of the content in this textbook.

Chapter 2 Math Foundations

2.1 Sets, Functions, and Notation

Sets are at the heart of mathematics, applied statistics, and data science. The mathematical realm that we most often work in – the real numbers – is a set, and understanding the ideas behind sets can help a data scientist understand

1. the terminology used in this book and by other empirical researchers
2. the mathematical foundations of probability/statistics.

While the foundations of statistics from a mathematical perspective is far beyond the scope of this book, it is nevertheless useful to understand the basic idea behind sets.

Definition 2.1. Set

*A **set** is a collection of unique objects (elements), such as numbers, and is often denoted by braces (i.e., $\{\ldots\}$) or capital letters (e.g., A). The set with no elements is called the **empty set** and is denoted by $\emptyset$.*

♣

For example, {1, 2, 3, 4} and {statistics, computer science, data science} are considered to be sets because they are both collections of distinct elements. Even though the second example is not mathematical, it satisfies the basic definition of a set and is therefore a set. However, {1, 1, 2, 3, 4} is not a set because it contains 1 twice.

2.1.1 Set Operations

Just as numbers can be added and multiplied, sets have their own operations. We are concerned with set *union* and set *intersection* as these show up in the basic probability theory that is required for data science.

Definition 2.2. Set Union

*The **union** of a collection of sets (denoted by $\cup$) is the set of all elements in the collection.*

♣

Definition 2.3. Set Intersection

*The **intersection** of a collection of sets (denoted by $\cap$) is the set of all elements shared by every set in the collection.*

♣

For example, suppose we have sets $A = \{1, 2, 3\}$ and $B = \{3, 4, 5\}$. The union of A and B is $A \cup B = \{1, 2, 3, 4, 5\}$ since the union contains all the elements in A and B. On the other hand, the intersection of A and B is $A \cap B = \{3\}$ since 3 is the only element in both sets.

While set operators are useful, very rarely do people say “take the intersection of sets A and B." We implicitly use these set operators in our everyday lives. Suppose someone is majoring in mathematics and statistics and needs to identify what courses they need to take. For their math curriculum, they take

courses like calculus, probability theory, algebra, analysis, etc. And for their statistics curriculum, they take courses like calculus, probability theory, regression analysis, applied statistics, etc. The math and statistics curricula are each their own sets because they are collections of distinct courses.

Suppose someone asks, “What are the required courses in math **and** statistics?" The use of “and" in that statement suggests the intersection of the sets of courses because they want courses shared by both sets of courses. If someone wanted to find out what all courses they need to take, then that is the same as finding the union. In everyday English, that is equivalent to saying, “What are the courses in math **or** statistics?" The use of “or" in that statement suggests the union of the sets of courses because they want the set of all courses. Understanding the relationship between “and"/“or" and sets allows a data scientist to translate domain problems into math.

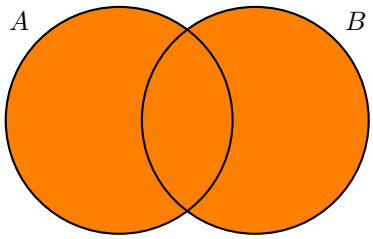


Figure 2.1: $A \cup B$

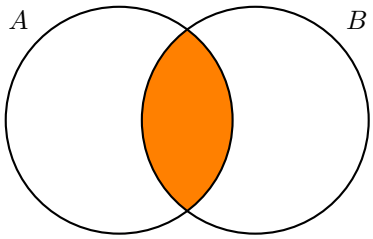


Figure 2.2: $A \cap B$

2.1.2 Functions

While sets on their own are very powerful tools, sets can be related to one another through **functions**. Secondary school algebra introduces functions to students, but it is not as rigorously defined as it is in this section. In applied statistics, understanding how functions work can guide the practitioner to decide what probability distributions most likely fit with certain quantities. This will be expanded upon more in later sections, but functions are essential to mathematics and statistics.

Definition 2.4. Function

*Suppose we have two sets A and B. A **function** is a process that associates each element A – referred to as the **domain** – to a single value of the set B – called the **co-domain**. In other words, if f is a function, then for all elements a in A, there exists one and only one element b contained in the set B such that $f(a) = b$.*

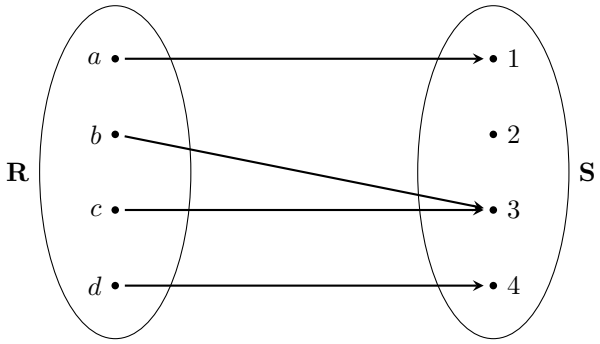
♣

Suppose we have a relationship f that maps values from the set $R = \{\text{a, b, c, d}\}$ to the set $S = \{1, 2, 3, 4, 5\}$

defined by

$$f(r) = \begin{cases} 1 \text{ if} & r = a \\ 3 \text{ if} & r = b \text{ or } r = c \\ 4 \text{ if} & r = d. \end{cases}$$

Graphically, the function would look as follows:

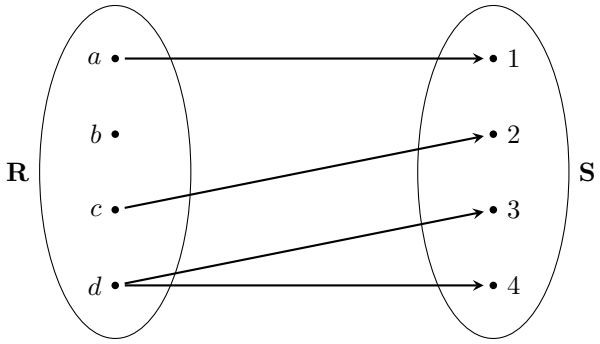


Notice that *every* element of R maps to *one and only one* value of S, so f is a function. Also, in this example, there is no element r in R such that $f(r) = 2$. People often make the mistake of saying that since nothing in R maps onto 2 that f is not a function. However, this is not a requirement of functions and is therefore not a problem. Additionally, notice that $f(b) = 3$ and $f(c) = 3$. Even though both values map into 2, this is not a problem.

Now consider the relation g that maps values R to S defined by

$$g(r) = \begin{cases} 1 \text{ if} & r = a \\ 2 \text{ if} & r = a \\ 3 \text{ if} & r = b \\ 4 \text{ if} & r = d. \end{cases}$$

Graphically, this relationship would look as follows:



This relation g is *not* a function for two reasons:

1. Even though b is an element of R, it is not mapped to any value in S
2. An element of the domain, d, maps to *two* values and it can map to one and only one for g to be considered a function.

Another key concept of functions is the **image** of the function. Suppose f maps values from arbitrary sets

A to B; this does not mean that all the values in B must have some corresponding value in A (i.e., for each element b in B, there does not need to exist a value in A such that $f(a) = b$). Because of this, there may be some values in B that are not mapped to. In order to "get rid of these," the **image** is the subset of B that contains all values that f maps values of A onto.

Definition 2.5. Image of a Function

*Let f be a function from A to B. The **image** of f is all the output values f may produce.* ♣

Example 2.1 Let f be defined from $(-\infty, \infty)$ to $(-\infty, \infty)$ defined by $f(x) = 0$. The image of f is the set of output values of f; since 0 is the only output value of f, the image of f is the set $\{0\}$.

Example 2.2 Let f be defined from $(-\infty, \infty)$ to $[0, \infty)$ defined by $f(x) = x^2$. The image of f is the set of output values of f. Since x^2 has output values $[0, \infty)$, the image of f is its co-domain.

Example 2.3 Let f be defined from $[10, \infty)$ to $[0, \infty)$ defined by $f(x) = x^2$. The image of f is the set of output values of f. While this looks very similar to the previous example, there is one key difference: the domain is now restricted to be greater than or equal to 10. Because of this, f is always greater than or equal to $10^2 = 100$. So, the image of f is $[100, \infty)$.

The distinction between the image and co-domain may seem quite tedious, but it is crucial in defining other ideas. Suppose someone is traveling from the United States to Japan to India. From the United States to Japan, the traveler exchanges the US dollar to Japanese yen; for each amount in US dollars, there is one and only one equivalent in Japanese yen and all US dollar conversions are considered. So this conversion rate is a function. From Japanese yen to Indian rupees, there is another conversion rate, which is also a function. So, the output of the US-Japan function is the input of the Japan-India function. Such an idea is referred to as the **composition of functions**.

Definition 2.6. Function Compositions

*A **function composition** is an operation that takes two functions f and g and creates a new function h such that $h(x) = g(f(x))$. In other words, the output of one function is the input of another. From this, it is easy to recognize that the domain of h is equivalent to the domain of f and the co-domain of h is the same as the codomain of g. This is often denoted as $(g \circ f)(x) = g(f(x))$.* ♣

Because the output of one function is the input of another, key restrictions must be placed. Namely, in order for the composition of functions to be a function itself, the image of the input function must itself be a subset of the domain of the output function. So for example, suppose a function f is to be the input function of the function g. Then, the composition of these functions would be $g(f(x))$. In order for $g(f(x))$ to be a function, all of the values in its domain must map to one and only one value in its co-domain. Suppose the image of f is not a subset of the domain of g. Then, there exists some element in f's domain, call it x_0, whose output that has no mapping in g. So, that means that $g(f(x_0))$ does not exist, and thus, $(g \circ f)(x)$ is not a function. Thus, in order for the composition itself to be a function, the image of f must be a subset of

the domain of g. The following examples can further illustrate these points.

Example 2.4 Let f map values from $R = \{\text{a, b, c, d}\}$ to $S = \{1, 2, 3, 4\}$ defined by

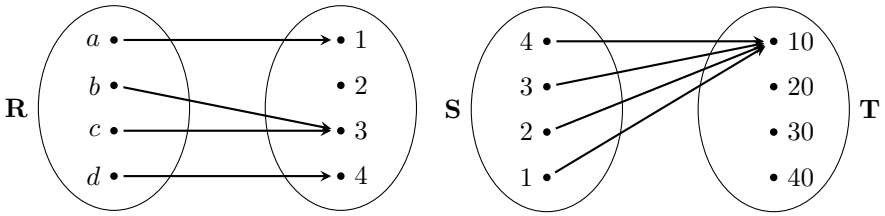
$$f(r) = \begin{cases} 1 \text{ if} & r = a \\ 3 \text{ if} & r = b \text{ or } r = c \\ 4 \text{ if} & r = d. \end{cases}$$

Let g map values from S to $T = \{10, 20, 30, 40\}$ defined by

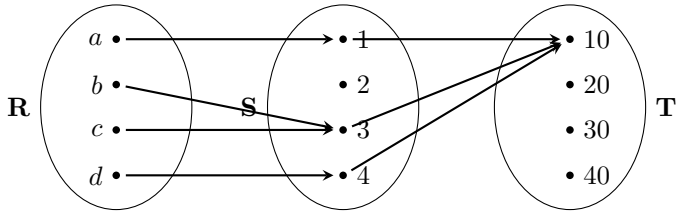
$$g(s) = 10.$$

Is it possible to create $(g \circ f)(x)$? Is it possible to create $(f \circ g)(x)$?

It would be useful to consider these functions graphically. The function f is on the left while g is on the right. Here, it is easy to recognize that the image of f is $\{1, 3, 4\}$, which is a subset of the domain of g, $S = \{1, 2, 3, 4\}$. However, the image of g, which is $\{10\}$, is not an input of f.



So, for the composition $(g \circ f)(x)$, the values in R all have somewhere to map (visually represented below). Even though the image of f and the domain of g are not equivalent, that is okay; the fact that the image of f is a subset of the domain of g is enough. Since each of the values in R is mapped to one and only one value of T, $(g \circ f)(x)$ is a valid function. On the other hand, because the image of g is not a subset of the domain of f (since 10 is not an element of R), it is not possible to create the composition $(f \circ g)(x)$.



2.2 Rates of Change and Derivatives

In high school algebra, students often discuss *rates of change*, a concept that defines how a change in one quantity relates to a change in another quantity. This idea is linked very closely to sets and functions as the most common use of rates of change is with functions: how does a change in inputs impact the output of a function? In earlier math classes, the phrase “rise over run" is an intuitive way of calculating rates of change. The “run"

Definition 2.7. Rate of Change

Suppose f is a function. Let x_0 and x_1 be elements of the domain of f. The rate of change of $f(x)$ from x_0 to x_1 is then defined as

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

since that is change in f (i.e., $f(x_1) - f(x_0)$) with respect to the change in x (i.e., $x_1 - x_0$). ♣

This is perhaps best explained through an example.

Example 2.5 Let f be a function from $[0, 1]$ (i.e., all the values between 0 and 1, inclusive) to $[0, 2]$ defined by

$$f(x) = 2x.$$

So for example, $f(0.5) = 2(0.5) = 1$. In other words, the output is just equal to twice the input.

This is a linear function, so the rate of change is equivalent to the slope. Intuitively, since 2 is the coefficient on x, the rate of change is 2. However, we can check this using definition **??**. Let $x_0 = 0$ and $x_1 = 1$. The choice of values is arbitrary since the function is linear, so all that matters is that x_0 and x_1 both belong to the domain of f. Then, the rate of change of f is

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{2 - 0}{1 - 0} = \frac{2}{1} = 2,$$

which is also the slope of $f(x)$.

2.2.1 The Derivative

Notice that the formula outlined in Definition **??** works for arbitrary x_0 and x_1 values, *unless* $x_0 = x_1$ because it falls victim to the “divide by zero" problem (i.e., 0 can never be in the denominator). So, instead of calculating

$$\frac{f(x_0) - f(x_1)}{x_0 - x_1}, \text{ where } x_0 = x_1,$$

calculate

$$\lim_{x_1 \to x_0} \frac{f(x_0) - f(x_1)}{x_0 - x_1}.$$

Then, it is possible to find the instantaneous rate of change.

Definition 2.8. Instantaneous Rate of Change

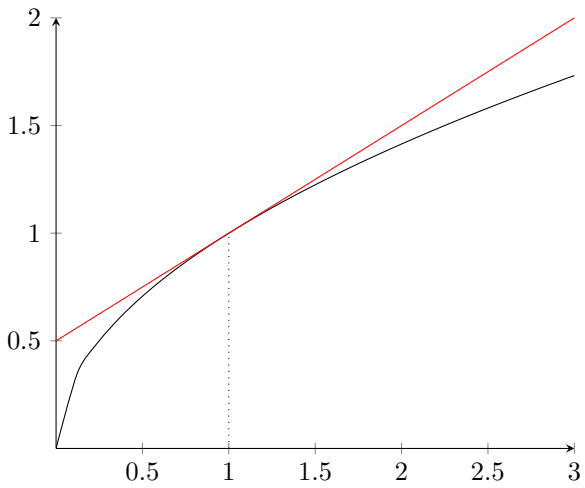
*The **instantaneous rate of change** is the rate of change at a single point x_0. For a function f, it is calculated as*

$$\lim_{x_1 \to x_0} \frac{f(x_0) - f(x_1)}{x_0 - x_1},$$

often denoted as $f'(x)$ or $\frac{df}{dx}$. ♣

Notice that there is nothing limiting this definition to only linear functions. The instantaneous rate of change can be calculated for polynomials (e.g., $f(x) = 5x^5 + 2x^3 + 9x + 6$), exponential functions (e.g., $f(x) = e^x$), logarithms (e.g., $f(x) = log_2((x))$, and several others. The instantaneous rate of change describes the slope of the line that “just touches" the function at a given point. For example, the red line

in the following graph “just touches" the black line at the dotted line; this is the instantaneous rate of change/derivative.



Definition 2.9. Tangent line

*Geometrically, the derivative at a point x_0 is interpreted as the rate of change of the line that “just touches" x_0, also known as the **tangent line**. If the derivative exists, then there is only one possible tangent line.*

♣

Unlike the rate of change calculation in Definition **??**, the derivative has two restrictions:

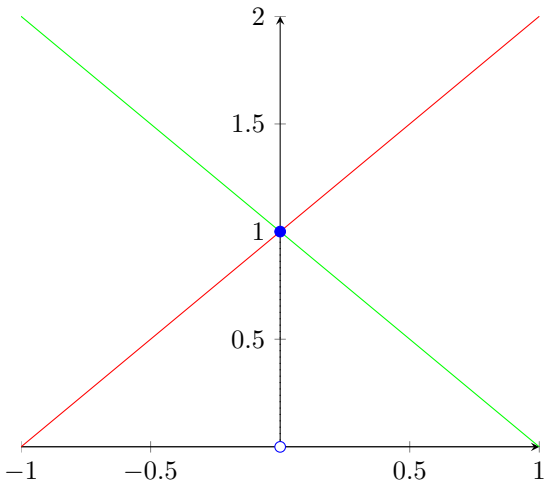
1. The function must not have an abrupt change in values at the point of interest (i.e., the function must be continuous at the point for which one wants to find the instantaneous rate of change)
2. The function at the point of interest cannot be a cusp or kink

To illustrate the need for these, consider the following examples.

Example 2.6 Consider the function g from $(-\infty, \infty)$ to $(-\infty, \infty)$ defined by

$$g(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise.} \end{cases}$$

Obviously, this function is not continuous at $x = 0$ because there is an abrupt change in values. Intuitively, consider what the tangent line may look like at $g(0)$. Because $g(0)$ is a single point, both the red and green lines “just touch" the blue dot, so $g(0)$ has more than one tangent line, which means its instantaneous rate of change is not unique and therefore does not have a derivative.



Example 2.7 Consider the function g from $(-\infty, \infty)$ to $(-\infty, \infty)$ defined by

$$g(x) = |x| = \begin{cases} x, & \text{if } x \geq 0 \\ -x, & \text{otherwise.} \end{cases}$$

What is the instantaneous rate of change at 0? Apply the formula from Definition 2.2.1:

$$\lim_{x_1 \to 0} \frac{g(0) - g(x_1)}{0 - x_1} = \lim_{x_1 \to 0} \frac{-g(x_1)}{-x_1}.$$

Consider the values $-1, \frac{-1}{2}, \frac{-1}{3}, \frac{-1}{4}, \ldots$. Notice that this sequence of values goes to 0 since the numerator is fixed at -1 but the denominator increases to ∞. Let x_1 be an arbitrary value selected from this set. Since all the values in this set are less than 0, $g(x_1) = -x_1$. So,

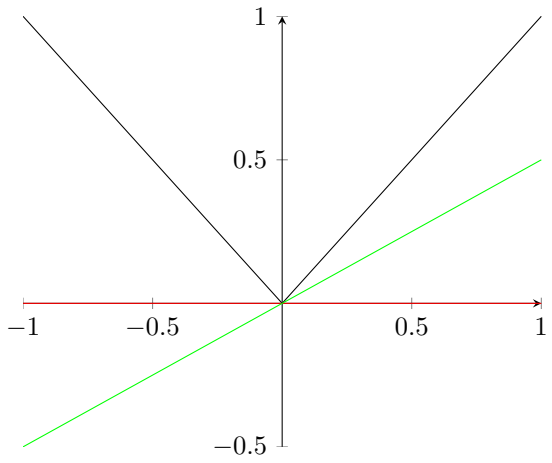
$$\begin{aligned} \lim_{x_1 \to 0} \frac{-g(x_1)}{-x_1} &= \lim_{x_1 \to 0} \frac{-(-x_1)}{-x_1} \\ &= \lim_{x_1 \to 0} \frac{x_1}{-x_1} \\ &= \lim_{x_1 \to 0} -1 \\ &= -1, \end{aligned}$$

so the instantaneous rate of change must be -1. However, now consider the values $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$. This sequence of values als goes to 0 since the numerator is fixed at 1 but the denominator increases to ∞. Let x_1 be an arbitrary value selected from this set. Since all the values in this set are less than 0, $g(x_1) = x_1$. So,

$$\begin{aligned} \lim_{x_1 \to 0} \frac{-g(x_1)}{-x_1} &= \lim_{x_1 \to 0} \frac{-x_1}{-x_1} \\ &= \lim_{x_1 \to 0} 1 \\ &= 1, \end{aligned}$$

so the instantaneous rate of change must be 1. The instantaneous rate of change cannot be two values at the same time, so it does not exist. This is because g is a kink at 0.

Geometrically, there can again be multiple tangent lines with different rates of change at $g(0)$, so no derivative exists for $g(0)$.



If a function f is continuous and has no kinks across its domain, often times, data scientists define a function f' that maps all the values in the domain to the instantaneous rate of change at each point.

Definition 2.10. Derivative of a Function

*Let f be a function from some set A to $(-\infty, \infty)$ that is both continuous and has no kinks. Then, if x is an element of the set A, there exists an instantaneous rate of change for x. So, the **derivative of a function** is a function, denoted as f', itself from A to $(-\infty, \infty)$ that maps each value x in A to the instantaneous rate of change of $f(x)$, often denoted as f'.*

♣

2.2.2 Derivative "Shortcuts"

While the derivative definition is often very useful for solidifying theoretical interpretations of the derivative, using the limit for actual calculations is very tedious. Mathematicians discovered derivative rules that are often used in calculus. Proving these rules is left to the reader.

Theorem 2.1. Derivative Rules

Let f and g be differentiable functions from some set A to $(-\infty, \infty)$. Then,

1. *If f is a polynomial of order n (i.e., $f(x) = x^n$, then $f'(x) = nx^{n-1}$*
2. *If f is of the form $f(x) = e^x$, then $f'(x) = e^x$*
3. *If f is of the form $f(x) = ln(x)$, then $f'(x) = \frac{1}{x}$*
4. *$(f+g)'(x) = f'(x) + g'(x)$: the derivative of a sum of functions, is the sum of the functions' derivatives*
5. *$(cf)'(x) = cf'(x)$: the derivative of a constant times a function is the constant times the function's derivative*
6. *$(f \cdot g)'(x) = f'(x)g(x) + f(x)g'(x)$: the derivative of a product of functions is equal to the sum of the derivative of the first function times the second function and the derivative of the second function times the first function*
7. *$\left(\frac{f}{g}\right)'(x) = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$: the derivative of the quotient of two functions is equal to the bottom times the derivative of the top minus the top times the derivative of the bottom divided by the bottom squared*

♡

Example 2.8

Example 2.9

2.2.3 The Chain Rule

Along with the arithmetic derivative rules comes the property of the **chain rule**, a formula to compute the derivative of a composition of functions. While neural networks – a famous and commonly used computing system in artificial intelligence – are not discussed in this textbook, the chain rule has guided the development of research in that area of research. It is useful even in the techniques laid out here, especially in discussions on the transformation theorem.

Definition 2.11. Chain Rule

Let f and g be differentiable functions such that $f(g(x))$ exists. Because f and g are both differentiable, $(f \circ g)$ must be differentiable. Then, $(f \circ g)'(x) = f'(g(x))g'(x)$.

♣

Example 2.10 Let f be a function from $(-\infty, \infty)$ to $[0, \infty)$ defined by $f(x) = x^2$. Let g be a function from $[0, \infty)$ to $[0, \infty)$ defined by $g(x) = 2x + 2$. Find $(f \circ g)'(x)$.

First, recognize that $(f \circ g)(x) = f(g(x))$ and this can be seen as a chain rule problem. So both $f'(g(x)) = \frac{d}{dx}(g(x))^2$ and $g'(x) = \frac{d}{dx}(2x + 2)$ are needed. To find $f'(g(x))$, recognize that this is a polynomial, so using the polynomial derivative rule, $f'(g(x)) = 2g(x) = 2(2x + 2)$. And to find $g'(x)$, recognize that g is the sum of two individual functions: $2x$ and 2, so $g'(x)$ can be found by summing the derivatives of $2x$ and 2, which are 2 and 0 respectively because of the polynomial rules. So, $g'(x) = 2 + 0 = 2$. Thus, $(f \circ g)'(x) = f'(g(x))g'(x) = (2(2x + 2))(2) = 4(2x + 2) = 8x + 8$.

This can also be computed without the chain rule, by taking advantage of the fact that $(f \circ g)(x) = f(g(x)) = (2x + 2)^2 = 4x^2 + 8x + 4$. Since this is function is a sum of polynomials, $\frac{d}{dx}(4x^2 + 8x + 4)$ simplifies to $8x + 8 + 0 = 8x + 8$. Thus, $(f \circ g)'(x) = 8x + 8$.

2.2.4 Finding Minima/Maxima

One of the key applications of the derivative is its ability to help practitioners find the points at which a function achieves minimum and maximum points. For example, consider the function $f(x) = x^2$. It is quite easy to recognize that the minimum of f occurs at $x = 0$, but for functions like $f(x) = x^6 + 4x^4 - 12x^3 + x^2 - 1$, finding minima/maxima is no longer a simple task. The derivative can help simplify this process. But first, consider the following definitions of minima and maxima.

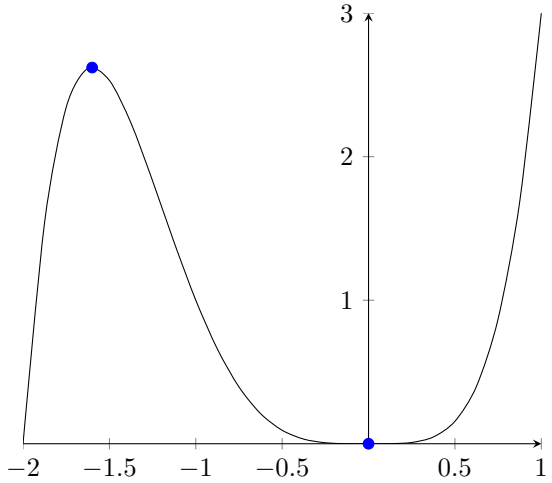
Definition 2.12. Global Minimum

*Let f be a function from some set A to $(-\infty, \infty)$. If f has a **global minimum**, then there exists some element of A denoted as x_0 such that $f(x_0) \leq f(x)$ for all elements x in A.* ♣

Definition 2.13. Global Maximum

*Let f be a function from some set A to $(-\infty, \infty)$. If f has a **global maximum**, then there exists some element of A denoted as x_0 such that $f(x_0) \geq f(x)$ for all elements x in A.* ♣

Often, people discuss global minima/maxima, but there are other types of minima and maxima that may be useful. Consider the function f from $(-\infty, \infty)$ to $(-\infty, \infty)$ defined by $f(x) = x^5 + 2x^4$. As x approaches $-\infty$, f goes to $-\infty$' and as x approaches ∞, f goes to ∞. So the global minimum/maximum do not exist since there is no single point x_0 such that $f(x_0)$ is less than/greater than all other points in the domain of f. However, in the following graph of f, it is quite easy to recognize that there are values that can be considered minima/maxima from $(-2, 1)$, a subset of the domain, highlighted in blue:



Such minima/maxima are referred to as **local minima/maxima**.

Definition 2.14. Local Minimum

*Let f be a function from some set A to $(-\infty, \infty)$. If f has a **local minimum**, then there exists some element of A denoted as x_0 such that $f(x_0) < f(x)$ for all elements x in some connected subset of A where x and x_0 are distinct.* ♣

Definition 2.15. Local Maximum

*Let f be a function from some set A to $(-\infty, \infty)$. If f has a **local maximum**, then there exists some element of A denoted as x_0 such that $f(x_0) > f(x)$ for all elements x in some connected subset of A where x and x_0 are distinct.*

♣

2.2.4.1 Finding Minima/Maxima

Consider some arbitrary function that is both continuous and un-kinked. Suppose the goal is to find a local minimum, which is known to exist. The local minimum (call it *min*) is, by definition, smaller than all the values in its neighborhood, so the function would look like a “U" around *min*. Notice that the points smaller than *min* all must have a decreasing rate of change since they must all be decreasing to approach *min*; and the points greater than *min* all must have an increasing rate of change since they must all be increasing “away from" *min*. In other words, the points smaller than *min* must have a *negative derivative*, while the points greater than *min* must have a *positive derivative*. That means that the derivative at *min* must be 0.

However, just finding where the function’s derivative is 0 is not enough to find the local minimum. Let *max* be some local maximum. By definition, *max* is greater than all the values in its neighborhood, so the function would look like a “$\cap$" around *max*. Using a similar argument as before, this would then yield the following result: the derivative at *max* must be equal to 0.

Furthermore, *max* and *min* are not the only values at which the derivative is equal to 0. Consider the function f from $(-\infty, \infty)$ defined by $f(x) = x^3$. Graphically, it is easy to recognize at 0, $f'(0) = 0$ because the rate of change of the tangent line is 0. However, 0 is certainly not a local minimum or maximum because the points directly smaller than 0 are all less than $f(0)$ and the points directly greater than 0 are all greater than $f(0)$. Thus, $f'(0) = 0$ but 0 is not a local maximum/minimum for f. Such a point is known as a **saddle point.**

Definition 2.16. Saddle Point

*Let f be a differentiable function (i.e., continuous and un-kinked). Let x_0 be an element in the domain of f. If $f'(x_0) = 0$ and x_0 is not a local minimum/maximum, then x_0 is known as a **saddle point**.*

♣

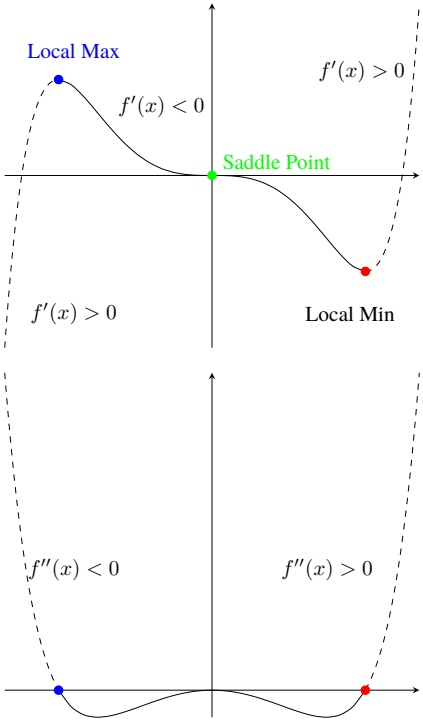
While finding the points at which the derivative is equal to 0 seems like a simple solution to finding local extrema, a little more analytical work is required. Notice that for a local maximum, the points directly to less than *max* are increasing towards *max*; in other words the derivative is positive there. Additionally, the points directly greater than *max* are decreasing away from *max*, so the derivative is negative there. On the other hand, for a local minimum, the points directly less than *min* are decreasing towards *min*, so the derivative is negative at those values. Furthermore, the points directly greater than *max* are decreasing away from *max*, so the derivative is positive there. For a saddle point, the derivative of the values directly before and after are the same sign. Using this result, one can classify local minima/maxima and saddle points.

So, in order to classify a point as a saddle point or extremum, the sign of the first derivative is used. If the derivative itself is a differentiable function, then it is possible to further simplify the process by using the **second derivative**, the derivative of the derivative.

Definition 2.17. The Second Derivative

Let f be a differentiable function. If f' is differentiable, then the ***second derivative*** *of f is the derivative of f', often denoted as f'' or $\frac{d^2f}{dx^2}$.* ♣

When the point is a local maximum, the sign of the first derivative changes from positive to negative, so the first derivative must be decreasing; in other words, the second derivative must be less than 0. On the other hand, when the point is a local minimum, the sign of the first derivative changes from negative to positive, so the first derivative must be increasing; so, the second derivative must be greater than 0. And when the sign of the first derivative is constant, there is a saddle point. So, when the second derivative is equal to 0, there is a saddle point. Using this test, classification of points as minima/maxima/saddle points is possible.



Example 2.11 Let f be a function from $(-\infty, \infty)$ to $(-\infty, \infty)$ defined by

$$f(x) = -x^7 + x^5 + x^3.$$

Find the local extrema and classify them as minima or maxima.

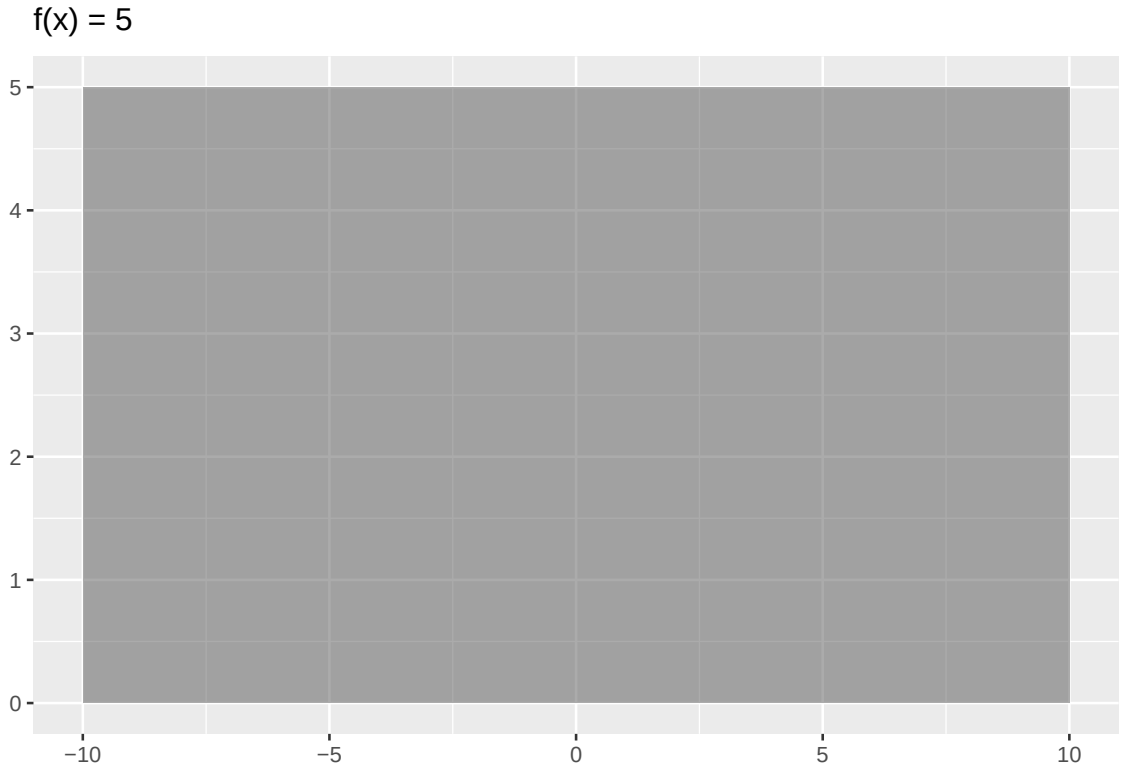
Notice that f is the sum of polynomials, $-x^7$, x^5, and x^3. So by combining the addition and polynomial arithmetic derivative rules, $f'(x) = -7x^6 + 5x^4 + 3x^2$. It is possible to find the local extrema by finding the zeros of the first derivative. It is expected that reader understands how to isolate the different values of x at which $f'(x) = 0$, so these steps are not provided. It is then found that $f'(x) = 0$ when x is one of $-1.05, 0, 1.05$. Because f' is a polynomial, it is continuous and un-kinked and is therefore differentiable. So classifying these values can utilize the second derivative. Notice, $f''(x) = \frac{d}{dx}(-7x^6) + \frac{d}{dx}(5x^4) + \frac{d}{dx}(3x^2)$. Each of these calculations utilize the fact that a constant term can be pulled out of a derivative and the polynomial rule. So, $f''(x) = -42x^5 + 20x^3 + 6x^2$. Notice, $f''(1.05) \approx -24.15 < 0$, so 1.05 is a local maximum. Additionally, $f''(-1.05) \approx 24.15 > 0$, so -1.05 is a local minimum. And $f''(0) = 0$, so 0 is a saddle point. Now, the last step is to understand whether 1.05 and -1.05 are local or global extrema; the only way they would not be global extrema is if the function tends towards $-\infty$ and ∞. So,

$\lim_{x\to-\infty} f(x) \to \infty$, so -1.05 is not a global maximum. And $\lim_{x\to\infty} f(x) \to -\infty$, so 1..05 is not a global minimum. Thus, 1.05 and -1.05 are local minimum and maximum respectively.

2.3 Sums and Integration

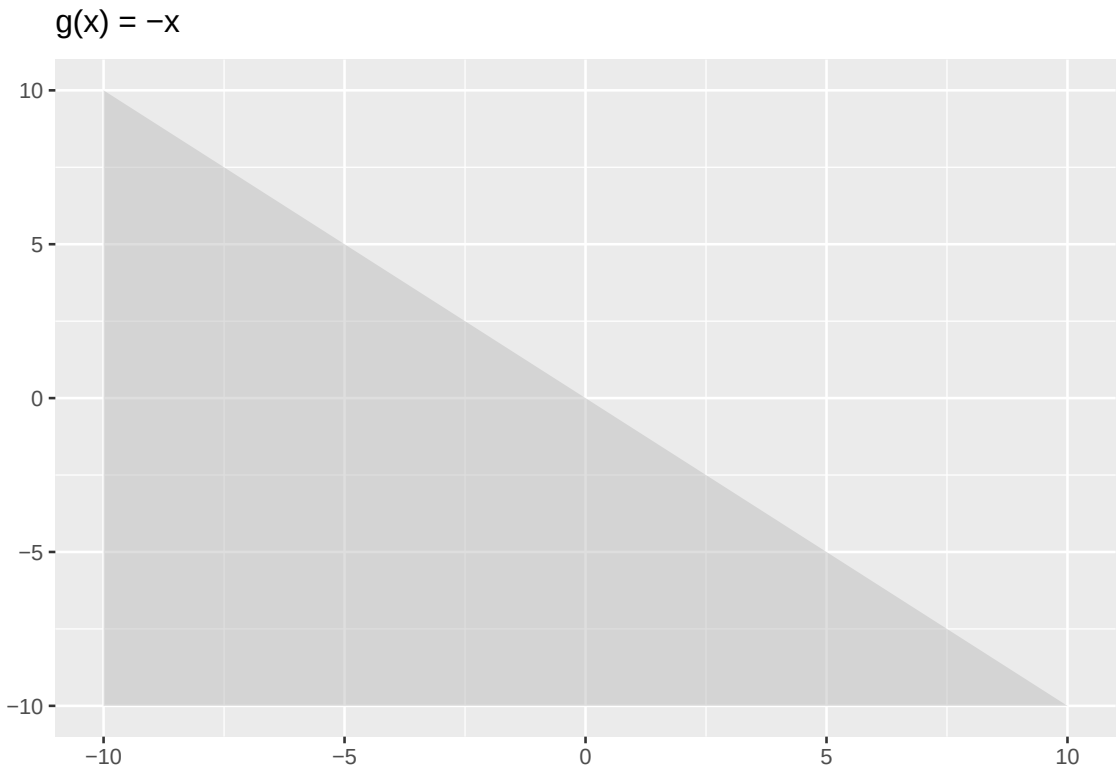
In addition to calculating rates of change, we often care about understanding areas under functions. % insert integration examples

We can start thinking about this intuitively: suppose we wanted to approximate the area under the function $f : [-10, 10] \to (-\infty, \infty)$ defined by $f(x) = 5$:

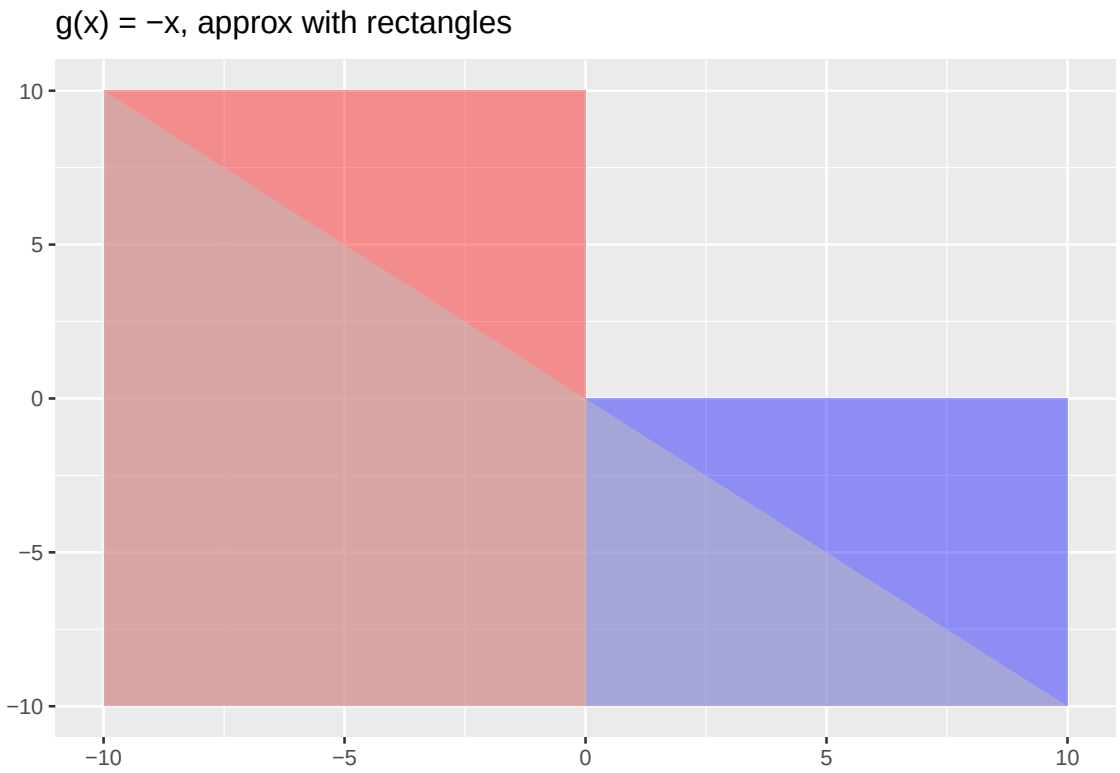


As we know from basic geometry, this is a rectangle and its area can be calculated as $base * height$. In this case, $base = 10 - (-10) = 20$ and $height = 5 - 0 = 5$, so the area under the function is $20 * 5 = 100$.

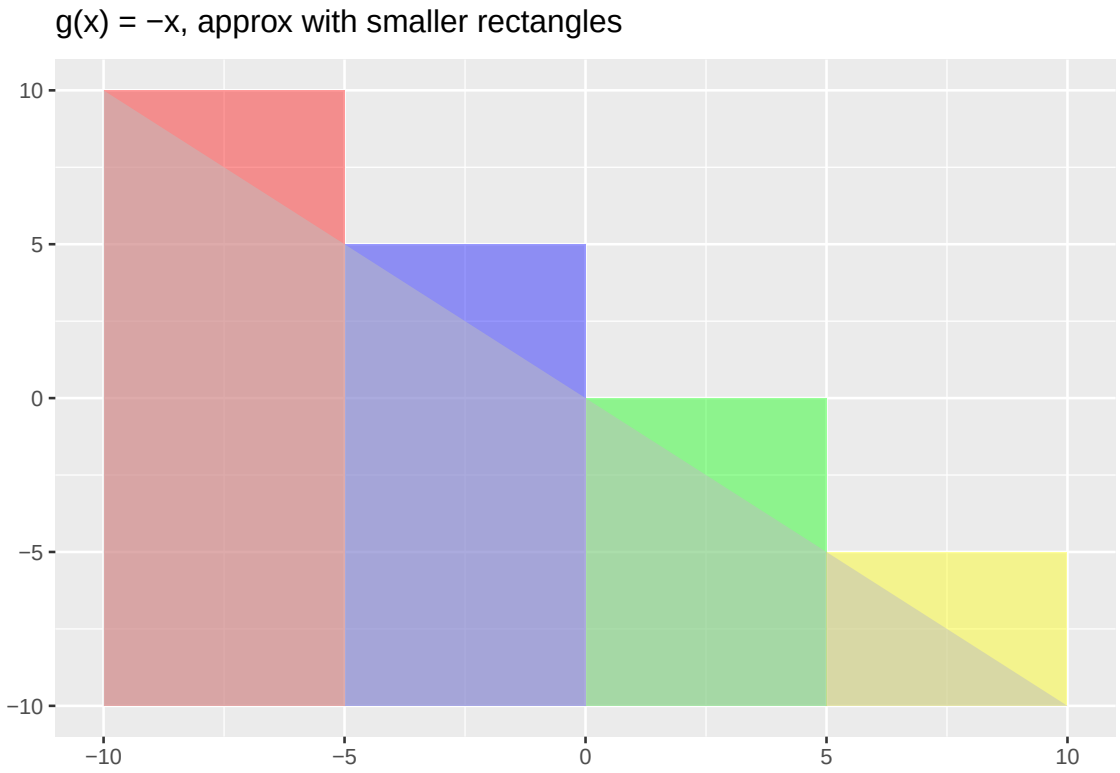
Now, consider the function $g : [-10, 10] \to (-\infty, \infty)$ defined by $g(x) = -x$:



While we know we can calculate this function's area because it is a triangle ($base * height$), we can also approximate the function's area using triangles:



The area of the red rectangle is $10 * 20 = 200$ and the area of the blue rectangle is $10 * 10 = 100$, so the approximated area of the triangle would be $200 + 100 = 300$, which is quite different than the true area of 200 However, we can get an even closer approximation by introducing more rectangles:



Here, the red rectangle has an area of $5 * 20 = 100$, the blue an area of $5 * 15 = 75$, the green an area of $5 * 10 = 50$, and the yellow an area of $5 * 5 = 25$; so the triangle's area approximation is $100 + 75 + 50 + 25 = 250$, which is closer to the true area of 200 than the earlier approximation with only two rectangles, suggesting that by reducing the width of the rectangles and introducing more, we can more closely approximate the area under the function.

However, let us more first define notation in order to allow us to write this information more compactly. Each rectangle will have the same width, or change in x, which we will refer to as Δx. Each rectangle's height is the value of g evaluated at the start of the rectangle. So the area of one rectangle that starts at the value a between -10 and 10 would be $\Delta a * g(a)$. To approximate the area of the function, we would then sum the areas of each individual rectangle. Suppose we are using natural number (i.e., 1, 2, 3, . . .) n number of rectangles to approximate the function. Then the width of each rectangle would be $\frac{10-(-10)}{n} = \frac{20}{n}$.

Then, the area of the triangle's approximation would be,
$\frac{20}{n}g(-10)+\frac{20}{n}g(-10+\frac{20}{n})+\frac{20}{n}g(-10+2\frac{20}{n})\ldots\frac{20}{n}g(-10+(n-2)\frac{20}{n})+\frac{20}{n}g(-10+(n-1)\frac{20}{n})$.
Notice, we can actually factor out $\frac{20}{n}$ since it is common in each term, which becomes

$$\frac{20}{n}\left[g\left(-10\right)+g\left(-10+\frac{20}{n}\right)+g\left(-10+2\frac{20}{n}\right)\ldots g\left(-10+\left(n-2\right)\frac{20}{n}\right)+g\left(-10+\left(n-1\right)\frac{20}{n}\right)\right].$$

In mathematics notation, we compress the summation using the $\sum$ symbol. $\sum_{i=1}^{n-1} i$ reads as "take the sum from i equals 1 to i equals n of i." So i becomes an indexing variable. The summation from the earlier equation now simplifies to

$$\frac{20}{n}\sum_{i=0}^{n-1} g\left(-10+\frac{20i}{n}\right),$$

which reads as "multiply the quotient of twenty divided by n by the sum from i equals 1 to $n-1$ of g of

negative ten plus twenty times i divided by n." As we discussed earlier, $\frac{20}{n}$ is simply the width of all of our rectangles, and because it is shared by all rectangle area calculations, we can factor it out; additionally, $-10 + \frac{20i}{n}$ means that we shift the starting point of the rectangle over by $\frac{20}{n}$ at each calculation; and $g\left(-10 + \frac{20i}{n}\right)$ is simply the height of each rectangle.

As we increase n, we approximate the area of the triangle more and more closely. Since we discussed limits already, we can apply a similar idea and get an exact solution for the area by calculating the limit of the sum as $n \to \infty$:

$$\lim_{n\to\infty} \frac{20}{n} \sum_{i=0}^{n-1} g\left(-10 + \frac{20i}{n}\right).$$

We can generalize the idea of approximating areas for any function with a bounded domain. Suppose we have a function $f : A \to B$ with sets A and B. Assume that A is bounded below by real number a and above by real number $\bar{a}$. Then the area under f will be

$$\lim_{n\to\infty} \frac{\bar{a} - a}{n} \sum_{i=0}^{n-1} f\left(a + \frac{\bar{a} - a}{n} i\right).$$

Recall that the goal of increasing n is to reduce the width of each of our rectangles, so we can actually rewrite this formula using Δx, the density of the rectangles:

$$\lim_{\Delta x\to 0} \Delta x \sum_{i=0}^{\frac{\bar{a}-a}{\Delta x}} f(a + i\Delta x).$$

The idea of using an infinite number of rectanlges to approximate the area under the curve of a bounded interval is known as a Reimann Integral.

Definition 2.18. Reimann Integral

*The **Reimann integral** is an approximation of the area under the curve of a bounded function using the infinite sum of infinitesimally dense rectangles. Let $f : A \to B$, with sets A and B, be a function with A being bounded below by a and above by $\bar{a}$. Then, the integral is denoted by $\int$ and the area under the function is written as $\int_a^{\bar{a}} f(x)dx$, where dx represents the variable that should be used for the density of the rectangles.*

♣

One limitation of the Reimann integral, however, is that it can only find the area over a bounded set. There are often instances when we are interested in finding the area over infinite spaces. For instance, while we will review this in our discussions on probability, one of the basic axioms of probability is that the probability over the entire space of the probability distribution is 1. When we have a distribution whose space is $(-\infty, \infty)$, like the normal distribution, we cannot know that the probability of the entire space is 1 if we cannot calculate the area under the distribution's curve using the Reimann Integral.

To solve this problem, we often using the Lebesgue integral. While we will not cover its derivation or even the intuition because it requires its own textbook and background, it is important to understand that we are able to solve integrals for unbounded sets. This will be the integral we refer to throughout this textbook, but the notation will remain consistent with the Reimann Integral problems, as will any techniques for actually solving integrals.

Theorem 2.2. Integration Rules

Let f and g be integrable functions over a bounded interval $[a, b]$. Then,

1. $\int_a^b [f(x) + g(x)]dx = \int_a^b f(x)dx + \int_a^b g(x)dx$
2. $\int_a^b f(x)dx = -\int_b^a f(x)dx$
3. $\int_a^a f(x)dx = 0$
4. *If f is a polynomial of order n (i.e., $f(x) = x^n$), then* $\int_a^b f(x)dx = \frac{b^{n+1} - a^{n+1}}{n+1}$
5. *If f is of the form $f(x) = e^x$, then* $\int_a^b f(x)dx = e^b - e^a$
6. *If f is of the form $f(x) = \frac{1}{x}$, then* $\int_a^b f(x)dx = ln(|b|) - ln(|a|)$.

♡

While derivatives and integrals may seem like disjoint ideas, they are actually importantly linked together. Intuitively, the integral represents the sum of the parts of some function. The derivative, on the other hand, represents the instantaneous changes in the function. So, the integral of the derivative will provide the sum of the minute changes of the original function, which is actually the total change in the total function. In other words, the integral and derivative can be seen as inverses of one another.

Mathematically, suppose there exists a function f that is differentiable over the interval $[a, b]$. Then,

$$\int_a^b f'(x)dx = f(b) - f(a).$$

Chapter 3 Probability

The basics of statistics and, by extension, data science are rooted in probability, the study of likelihoods and chances of outcomes and realizations. Using probability, we can predict the likelihood of future events or understand the effect of some treatment or interpolate missing values across space. Probability depends heavily on sets since we are usually interested in understand the chance of a set of outcomes occurring. The set of all possible outcomes is known as the sample space. For example, the sample space of a coin flip is $\{heads, tails\}$ because a coin can either come up as heads or tails. An event is some subset of the same space. For example, $\{heads\}, \{tails\}, \{heads, tails\}, \emptyset$ are the possible events of a coin flipping game. The probability of an event is a representation of the chance of that event occurring, with a probability of 1 meaning that the event is guaranteed while a probability of 0 meaning that the event is guaranteed to not happen.

Definition 3.1. Sample space

*A **sample space** is the set (i.e., all elements are unique) of all possible outcomes in a probability space. It is often denoted by* Ω*.* ♣

Definition 3.2. Event

*An **event** is some subset of the sample space.* ♣

There are some aspects of probability theory that we take as truth and based on these assumptions build the rest of probability. These are known as the axioms of probability. Consider the sets of events A, B, a subset of the sample space Ω, with $\mathbb{P}(A)$ denotes the probability of A. The axioms state

1. Nonnegativity: $\mathbb{P}(A) \geq 0$
2. Normalization: $\mathbb{P}(\Omega) = 1$
3. Finite Additivity: If A and B are disjoint (i.e., $A \cap B$) $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$.

As a consequence of these axioms, $\mathbb{P}(A) \leq 1$, $\mathbb{P}(\emptyset) = 0$, $P(A) + P(A^c) = 1$–where A^c represents the complement of A (i.e., all the elements not in A but in Ω)–$B \subset A$ (i.e., B is a strict subset of A) implies $\mathbb{P}(B) < \mathbb{P}(A)$.

3.1 Probability Basics

We often run into situations when we are interested in the probability of the union of two non-disjoint sets, A and B. Then,

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B).$$

While the $\mathbb{P}(A \cup B)$ may seem unintuitive, recall that sets cannot have duplicated elements. So, since A and B are non-disjoint sets, they share elements. So, $A \cup B$ will have less elements that the sum of the number of elements in A and B individually. By including the $\mathbb{P}(A \cap B)$ term, we account for the redundancy in elements between the two sets. And when A and B are disjoint, $A \cap B = \emptyset$; since $\mathbb{P}(\emptyset) = 0$,

$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$.

Example 3.1 Suppose we have 100 students in a school: 50 students who have science majors, 60 students who have liberal arts majors, and 30 are both. What are the chances that we randomly select a student who has a science and/or liberal arts major?

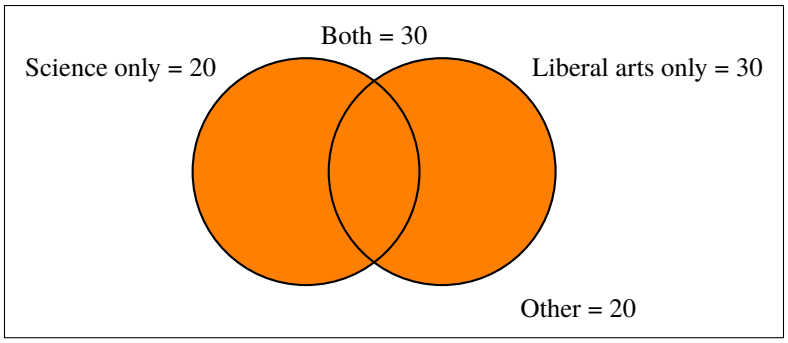


Figure 3.1: Example **??** Venn Diagram

First, recognize that since we have 100 students, the chances of selecting any one student, irrespective of their major, is $\frac{1}{100}$. Now, because we have 50 science majors (set A in this situation), the chances of randomly selecting a student who has a science major is $50 * \frac{1}{100} = 0.5$. Similarly, because we have 60 liberal arts majors (set B in this situation), the chances of selecting a student who has a liberal arts major is $60 * \frac{1}{100} = 0.6$. Lastly, because we have 30 students who are majoring in both science and liberal arts ($A \cap B$), the probability of choosing someone who is majoring in both is $30 * \frac{1}{100} = 0.3$. So, the probability of choosing any individual student who has a science or liberal arts major is $0.5 + 0.6 - 0.3 = 0.8$. Again, because we are "double counting" students across both majors, we must remove the redundancy, which is why we subtract 0.3.

We can also validate our answer by understanding the role of the science and liberal arts majors in the whole sample space. In this situation, students can either be science majors, liberal arts majors, both, or neither: there is no other possibility. So if we are interested in the population of science, liberal arts, or both, that means that the only remaining possibility is that a student is majoring in something else. Because we know that $\mathbb{P}(A) = 1 - \mathbb{P}(A^c)$ for some set A, a student majoring in something else is the complement of majoring in science, liberal arts, or both. Since there are 20 students majoring in something else, $\mathbb{P}(science \cup liberal\ arts) = 1 - \mathbb{P}(other) = 1 - 0.2 = 0.8$, the answer we arrived at earlier.

In addition to the inclusion-exclusion principle, we often need to consider conditional probabilities. Considering the setup in Example 3.1, suppose we are working for the liberal arts school. The school is interested in creating a scholarship for its students and wants to know the probability that a double major will win the award randomly. While it may seem as though there is a 30% chance because 30 of the 100 students in the university are double majors, the liberal arts school only cares about students who have liberal arts majors, which is actually only 60 students. So the chances of awarding the scholarship to a liberal arts student is 50%, not 30%.

We often denote conditional probabilities with the $|$ sign. Consider two sets A and B. If we want to know the probability of the events in A occurring, knowing that B is true, then we write this as $\mathbb{P}(A|B)$ and it is calculated as $\frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$. Let us break this down more: because B is guaranteed to be true, we only care about the outcomes in A that also coincide with outcomes in B, which is the $A \cap B$ term. And because B

is true, we are no longer working with the entire population – only the population in B.

Definition 3.3. Conditional Probability

The chance of the set A occurring, conditional upon set B being true, is known as ***conditional probability*** *and is calculated as* $\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$.

♣

From conditional probabilities comes the important idea of independence. In probability, we say two events are independent if one event occurring does not change the outcome of another event. So, for example, when flipping a fair coin, the outcome from one flip will not impact the outcome from a subsequent flip: these are independent events. Mathematically, we know the event A is independent from the event B if $\mathbb{P}(A|B) = \mathbb{P}(A)$: the chances of A occurring do not depend on B.

Definition 3.4. Law of Total Probability

Suppose we have two events A and B. Partition B into n disjoint sets: $\{B_1, B_2, \ldots, B_n\}$ The ***Law of Total Probability*** *states that* $\mathbb{P}(A) = \sum_{i=1}^{n} \mathbb{P}(A|B_i)\mathbb{P}(B_i)$.

♣

From the concept of independence comes an important relationship about set intersections. Suppose the sets A and B are independent of one another. Then, $\mathbb{P}(A|B) = \mathbb{P}(A)$. Recall also that $\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$. So, that implies that $\mathbb{P}(A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$, which in turn can be written as $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$. So, if two events are independent, then the probability of their intersection is equivalent to the product of their individual probabilities.

Definition 3.5. Independence

Two events A and B are independent if $\mathbb{P}(A|B) = \mathbb{P}(A)$. *From this, we know that* $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$ *if A and B are independent.*

♣

Because we are discussing calculating the probability of two events co-occurring, notice that $\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$, so $\mathbb{P}(A \cap B) = \mathbb{P}(A|B)\mathbb{P}(B)$. Since $\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}$, we can now replace $\mathbb{P}(A \cap B)$ with $\mathbb{P}(A|B)\mathbb{P}(B)$. So,

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

This idea, known as Bayes' Rule, is really just converting a realization of an event (B) into the actual occurrence of an event (A).

Definition 3.6. Bayes' Rule

Bayes' Rule *states that, given two events A and B,*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

♣

Example 3.2 We are testing students for COVID-19. There are four possible outcomes: the student tests positive and the student actually has COVID-19, the student tests negative and the student actually does not have COVID-19, the student tests positive and the student does not actually have COVID-19, and the student tests negative and the student does actually have COVID-19. The following table illustrates the probability of each of these events occurring.

We are given the following information: students have a 20% chance of having COVID-19. However,

conditional upon having COVID-19, a student has a 98% chance of testing positive and a 2% chance of testing negative. And conditional upon not having COVID-19, a student has a 1% chance of testing positive and a 99% chance of testing negative.

We know that students will test positive or negative, but using this information, can we backtrack and understand the chances that a student actually has COVID-19 given that they tested positive?

From conditional probability rules, we know that

$$\mathbb{P}(\text{has COVID-19}|\text{test positive}) = \frac{\mathbb{P}(\text{has COVID-19, test positive})}{\mathbb{P}(\text{test positive})}.$$

However, we have no information on $\mathbb{P}(\text{has COVID-19, test positive})$. Instead, we can take advantage of Bayes' Rule to backtrack this information:

$$\mathbb{P}(\text{has COVID-19}|\text{test positive}) = \frac{\mathbb{P}(\text{test positive | has COVID-19})\mathbb{P}(\text{has COVID-19})}{\mathbb{P}(\text{test positive})}.$$

We know that $\mathbb{P}(\text{test positive | has COVID-19}) = 0.98$ and $\mathbb{P}(\text{has COVID-19}) = 0.2$. However, what is the probability of a student testing positive? From the law of total probability, we know that $\mathbb{P}(\text{test positive}) = \mathbb{P}(\text{test positive | has COVID-19})\mathbb{P}(\text{has COVID-19}) + \mathbb{P}(\text{test positive | does not have COVID-19})\mathbb{P}(\text{does not have COVID-19})$. So, $\mathbb{P}(\text{test positive}) = 0.98(0.2) + 0.01(0.2) = 0.196 + 0.002 = 0.198$.

Putting all of the factors together, $\mathbb{P}(\text{has COVID-19}|\text{test positive}) = \frac{0.98(0.2)}{0.198} = \frac{0.196}{0.198} \approx 0.99$. So, the chance that a student actually has COVID-19 given that they test positive is 99%.

3.2 Random Variables and Probability Distributions

If we think back to elementary mathematics, we first learned about basic mathematical operations on the integers (e.g., addition and multiplication) and then discussed algebra and solving for missing values. Likewise, now that we understand the basic set theoretic operations for probability, we can expand our toolkit to discuss probabilistic variables, often referred to as random variables.

Definition 3.7. Random variables

*A **random variable** is a function that maps the possible values in a sample space to numerical representations. Random variables are typically denoted with capital letters.* ♣

Consider the following example for the outcome of a fair coin toss: let X be a random variable for the outcome of a fair coin toss. Then, $X = 1$ if we flip heads up and $X = 0$ if we flip tails up. While random variables are mappings of possible values to numerical representations, they do not detail the exact value of a coin toss, for example. After we toss the coin, the coin takes on some value (i.e., heads or tails) and there is some numeric representation for that outcome (i.e., 0 or 1). The numeric representation of the actual outcome is known as the realization.

Definition 3.8. Realization

*The value that a random variable actually takes on is called the **realization**.* ♣

So far, we have discussed random variables as simply mappings from the sample space to a numerical representation; however, we know that in a probability space, we have more than just the sample space. We also have the probability function. For example, in our discussion of coin tosses, the probability function maps heads to 50% and tails to 50%. We extend this probability function to now work for random variables; namely, the probability distribution is the mapping of a random variable's outcomes to a probability. Reconsidering the random variable X for a coin toss (i.e.,$X = 1$ for heads up and $X = 0$ for tails up), the probability that $X = 1$ is the same as saying the probability of flipping a coin heads up. The probability that $X = 0$ is the same as saying the probability of flipping tails up.

Definition 3.9. Probability Distribution

*A **probability distribution** is a function that maps the outcomes of a random variable to the probability of observing that outcome.*

♣

3.2.1 Discrete Random Variables

There are two types of random variables: discrete and continuous. Discrete random variables typically deal with sample spaces whose possible outcomes we can write in a list, and their probability distribution functions are called probability mass functions. For example, let X be a random variable representing the face of a single die roll. There are six possible values of X: $\{1, 2, 3, 4, 5, 6\}$. Because there is a finite number of outcomes, the random variable of X is a discrete distribution. Additionally, suppose Y is the number of times we flip a coin heads up in an infinite number of tries. Then, the possible values of Y are $\{0, 1, 2, 3, \dots \infty\}$. Even though there is an infinite number of possible values for Y, we can write all of the outcomes in a list. So, Y is a discrete random variable. Now, assume that Z is a random variable that selects a real number in the space $[0, 1]$. No matter how hard we try, we can never list out all the possible outcomes of Z. By way of contradiction, let us assume that we can list out the possible outcomes of Z: $\{a_1, a_2, \dots, a_\infty\}$, where a_i is some possible outcome for Z. Because Z's domain is over the space $[0, 1]$, there will definitely be a number between any two consecutive elements, i.e. $\frac{a_i + a_{i+1}}{2}$ is also in Z. So no matter how hard we try, we will have an uncountably infinite number of elements in Z's co-domain, so Z is not a discrete random variables.

Definition 3.10. Probability Mass Function

*A **probabilty mass function** (pmf) is the probability distribution function for a discrete random variable. Let X be a discrete random variable. The pmf of X, which we call $f(x)$, is given by $f_X(x) = \mathbb{P}(X = x)$; in other words, it describes the probability that the disrecete random variable is equal to a specific value.*

♣

Let us consider a few examples of discrete random variables.

Example 3.3 Suppose we are rolling a six-sided, fair die. Let N be a random variable representing the number of times we roll the die until we roll our first 6. We want to find the support and the probability mass function of N. Let us first recall what the support and pmf of a random variable are. The pmf is the probability of realizing an event c, i.e. $\mathbb{P}(N = n)$. And the support is all values of n such that $\mathbb{P}(N = c) > 0$. Because we can roll $1, 2, 3, \dots, \infty$ number of times before rolling a 6, the values of N that have probabilities over 0 are $\{1, 2, \dots, \infty\}$. These values are known as the support of a random variable.

Definition 3.11. Support

*Let X be a random variable with probability distribution function $f_X(x)$. The **support** of X is all the possible values of X such that $f_X(x) > 0$.*

♣

Now, finding the pmf is a little trickier but doable. To help us keep track of which roll we are discussing, let us introduce new random variables $X_1, X_2, \ldots, X_n$, each of which represent the outcome of a single roll; for any random variable in this set, represented by X_i,

$$X_i = \begin{cases} 1 & \text{if we roll a 6} \\ 0 & \text{if we do not roll a 6.} \end{cases}$$

Then, the pmf of X_i would be given by

$$f_X(x) = \begin{cases} \frac{1}{6} & \text{if } x = 1 \\ \frac{5}{6} & \text{if } x = 0. \end{cases}$$

First, realize that $\mathbb{P}(N = n)$ is the same as saying that we do not roll a 6 for $n - 1$ times and roll a 6 the last time. In other words, $\mathbb{P}(N = n) = \mathbb{P}(X_1 = 0, X_2 = 0, \ldots, X_{n-1} = 0, X_n = 1)$. While the joint probability may seem intimidating, we can easily simplify it by recalling that because the outcome of one roll does not impact the probability of another roll, each of the X_i's are independent of one another. This means that for two events A and B, $\mathbb{P}(A, B) = \mathbb{P}(A)\mathbb{P}(B)$. So, because they are independent,

$$\begin{aligned} \mathbb{P}(N = n) &= \mathbb{P}(X_1 = 0, X_2 = 0, \ldots, X_{n-1} = 0, X_n = 1) \\ &= \mathbb{P}(X_1 = 0)\mathbb{P}(X_2 = 0)...\mathbb{P}(X_{n-1} = 0)\mathbb{P}(X_n = 1) \\ &= \mathbb{P}(X_n = 1) \prod_{i=1}^{n-1} \mathbb{P}(X_i = 0) \\ &= \frac{1}{6} \prod_{i=1}^{n-1} \frac{5}{6} \\ &= \frac{1}{6} \left(\frac{5}{6}\right)^{n-1}. \end{aligned}$$

So, the probability mass function of N is

$$f_N(n) = \frac{1}{6} \left(\frac{5}{6}\right)^{n-1}.$$

This example covers a few important topics. Besides offering an example of discrete random variables, we also introduced the idea of the support of a distribution. Implicitly, we also discussed independently and identically distribution, or IID for short, variables. When we have a collection of random variables that all follow the same probability distribution function and are independent of one another, we call them independently and identically distributed or IID for short. The idea of IID variables is crucial in applied statistics and data science. <!– First, the probability mass function of the form $p(1-p)^{n-1}$ for some p between $[0, 1]$ and a random variable with the support being the whole numbers (i.e., $0, 1, 2, \ldots, \infty$) is known as the geometric distribution. These are typically used to model "time-to-event" problems in discrete settings, like the number of times we need to roll a die to see a 6. –>

<!– Implicitly through the problem, we also learned about the Bernoulli distribution: when the probability

mass function is of the form $\mathbb{P}(X = 1) = p$ and $\mathbb{P}(X = 0) = 1 - p$ for some p in $[0, 1]$ and a random variable with support $\{0, 1\}$, the pmf is called a Bernoulli distribution. These are typically used to model "indicator" problems that deal with binary responses. –>

Definition 3.12. IID

*A collection of random variables are called independently and identically distributed, or **IID** for short, if all the random variables follow the same probability distribution and are independent of one another.*

♣

3.2.2 Continuous Random Variables

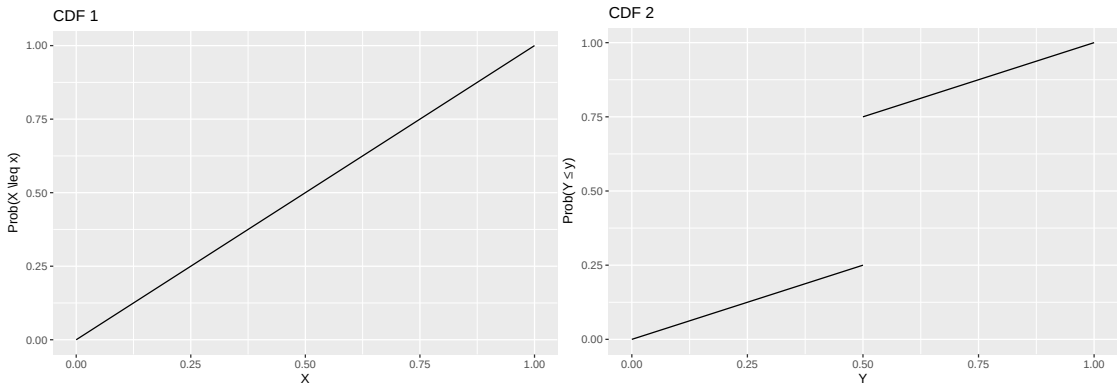
To understand continuous random variables, we first need to introduce the cumulative distribution function, a function that maps the possible outcomes of a random variable to values between 0 and 1. The cumulative distribution function asks "what is the probability that the realization of some random variable is less than or equal to a specified value?" While it is not important to our discussion now, because the CDF only looks at "less than or equal to" values, the CDF is an increasing function. In other words, let x_1, x_2 be elements of the sample space of the random variable X. If $x_1 < x_2$, then $F_X(x_1) \leq F_X(x_2)$ where $F_X(x)$ represents the CDF of X.

Definition 3.13. Cumulative Distribution Function

*Let X be a random variable. Then, the **cumulative distribution function** (CDF) $F_X(x)$ is a function that maps X to $[0, 1]$ defined by $F_X(x) = \mathbb{P}(X \leq x)$. The general notation of the CDF is $F_{[\textit{random variable}]}(\textit{realization})$.*

♣

Now, the formal definition of a continuous probability distribution is a random variable whose cumulative distribution function is differentiable. Consider the following cumulative distribution functions:



First, it is easy to recognize that the cumulative distribution function of X is differentiable because it is smooth and there are no breaks or kinks in the function. So, X's probability distribution is continuous. On the other hand, notice that CDF2 has a sudden jump at $Y = 0.5$, so its CDF is not differentiable and Y is therefore not a continuous random variable. However, not continuous does not imply discrete. Because we cannot "list out" the possible values of Y, Y's probability distribution is not discrete either.

While probability mass functions calculate $\mathbb{P}(X = x)$ for a discrete random variable X, probability density functions are not as straightforward because in an uncountably infinite set, the possibility of choosing exactly one value is 0. Let Y be a random variable chosen between $[0, 1]$, and suppose we were interested

in the probability that Y is 0.3; then, $\mathbb{P}(Y = 0.3) = \frac{1}{|[0,1]|}$, where $|A|$ represents the number of units in the set A. Because $[0, 1]$ is continuous, it has an uncountably infinite number of values. So, $\mathbb{P}(Y = 0.3) = 0$, which would mean that it is impossible for us to observe 0.3. Instead, we will ask what is the probability that Y lies in the range $[0.3, 0.3 + \delta]$, where δ is some infinitesimal number. For the same reason that $[0, 1]$ has an uncountably infinite number of values, ($[0.3, 0.3 + \delta]$ has an uncountably infinite number of possible values; $[0.3, 0.3 + \delta]$ simply has a smaller uncountably infinite number of values. So we should be able to quantify the probability that $Y = 0.3$.

We know that $F_Y(0.3) = \mathbb{P}(Y \leq 0.3)$ and $F_Y(0.3 + \delta) = \mathbb{Y} \leq \nvdash \nVdash + \delta$. So, we should be able to find $\mathbb{P}(0.3 \leq Y \leq 0.3 + \delta)$ with $F_Y(0.3 + \delta) - F_Y(0.3)$, which is actually a calculation for the change in F_Y from 0.3 to $0.3+\delta$. Now, we can get even more precise values by finding $\lim \delta \to 0 F_Y(0.3+\delta) - F_Y(0.3)$, which is just the instantaneous rate of change, which is just the derivative. So, we define the probability density function of Y as $f_Y(y) = \frac{d}{dy} F_Y(y)$.

Definition 3.14. Probability density function

*Let Y be a continuous random variable with CDF $F_Y(y)$. Then, the **probability density function** of Y is given by $f_Y(y) = \frac{d}{dy} F_Y(y)$.*

♣

We know that probability distributions are functions with domains and co-domains. However, we often care about something beyond the domain in probability: the support of a probability distribution. As an illustration, suppose we are trying to relate income to a random variable. We can define a variable Z that maps the sample space $(-\infty, \infty)$ to (∞, ∞); because it maps the entire sample space to the real numbers, Z is a random variable. However, income (in this very simplistic setting) cannot be negative. So, $\mathbb{P}(Z < 0) = 0$. Even though, $(-\infty, 0)$ is a part of the domain of Z's probability distribution, we care more about $[0, \infty)$ because those values are possible.

3.3 Named Distributions

Every probability distribution has its own functional form, but some are so well known and so widely used that they have their own names. In this portion, we will discuss a few of the main distributions and demonstrate how to sample values from the distribution in `R`.

3.3.1 Named Discrete Random Variables and Distributions

3.3.1.1 Bernoulli Random Variable

If a random variable X's support is $\{0, 1\}$ (i.e., $\mathbb{P}(X = x) > 0$ if and only if $x = 0$ or $x = 1$), then X is said to follow the Bernoulli distribution.

Definition 3.15. Bernoulli distribution

Let X be a random variable that follows the ***Bernoulli distribution****. Then, the probability mass function of X is*

$$f_X(x) = \begin{cases} p & \text{if } x = 1 \\ 1 - p & \text{if } x = 0 \end{cases} \tag{3.1}$$

for some p in the range $(0, 1)$. We denote X as a Bernoulli random variable with parameter p by writing $X \ Bernoulli(p)$. We read the symbol as "is distributed as."

♣

We often use Bernoulli random variables to indicate whether an outcome occurred or not. The outcome of interest occurring is represented by the random variable taking on a value of 1 and a value of 0 if the outcome did not occur. We also use Bernoulli random variables in experiments where the only outcomes are either "successes" or "failures" but not both. In this type of experiment, known as a Bernoulli trial, we can consider the Bernoulli random variable to be an indicator of success, such that the random variable is 1 if the success occurs or the random variable is 0 if the success does not occur. Because the random variable is 1 with probability p, the parameter p is sometimes referred to as the "success probability."

We actually saw Bernoulli random variables in Example 3.3. We used them to denote whether the outcome of a single die roll was a six or not. In this example too, the Bernoulli random variable was used as an indicator variable of sorts.

3.3.1.2 Binomial Random Variable

Let N be some positive integer and p be a real number in the range $(0, 1)$. Let $X_1, \ldots, X_N$ be N random variables all simulated from $Bernoulli(p)$. Let $X = \sum_{i=1}^{N} X_i$; in other words, X represents the number of successes in N independent Bernoulli trials with the same success probability p. X follows the Binomial distribution, and we therefore call it a binomial random variable.

We know that probability mass functions have two major components: the support (i.e., values of X such that $\mathbb{P}(X = x) > 0$) and the functional form (i.e., $\mathbb{P}(X = x)$). So let us intuitively devise both of these components.

Starting with the support, we know that we have N independent trials and that X represents the number of successes. The lower bound for X is having no trials succeed, which would mean $X = 0$. And the upper bound for X is having all the trials succeed, which would mean $X = N$. Additionally, it is impossible for us to have a fractional number of successes (e.g., 2.5 successes), so the support of X is all integer values between 0 and N, inclusive.

The pmf is a little more difficult. But suppose $X = 0$; this would imply that $X_1 = 0, X_2 = 0, \ldots, X_N = 0$. And because $X_1, \ldots, X_N$ are IID random variables, they are independent, which means that the joint probability of all these events occurring is the product of the probabilities of the individual events occurring:

$$\begin{aligned}\mathbb{P}(X_1 = 0, X_2 = 0, \ldots, X_N = 0) &= \mathbb{P}(X_1 = 0)\mathbb{P}(X_2 = 0)\ldots\mathbb{P}(X_N = 0)\\ &= (1-p)(1-p)\ldots(1-p)\\ &= (1-p)^N,\end{aligned}$$

because the probability of any single trial failing is $(1-p)$, and we multiply $(1-p)$ N times. So, $\mathbb{P}(X = 0) = (1-p)^N$.

Now, suppose $X = 1$. Then, that means exactly 1 of the X_i's is 1. This could be when $X_1 = 1$, all else 0; or $X_2 = 1$, all else 0; or $X_3 = 1$, all else 0; etc. Notice, we have a sequence of *or* statements, which means we can apply the inclusion-exclusion principle (i.e., given events A and B, $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$). Notice, we can never realize two sequences together. For example, the events "$X_1 = 1$, all else 0" and "$X_2 = 1$, all else 0" can never occur together because they are contradictory (X_2 cannot be 1 if all else, which includes X_2, is 0 when $X_1 = 1$). So in this case, we can just sum the probability of each sequence of outcomes together. Now suppose we are looking at one individual sequence: $X_i = 1$, all else 0. Then, because each of the X_i are independent, we have $(1-p)$ multiplied $N-1$ times for the $N-1$ failures multiplied by p for the 1 success: $p(1-p)^{N-1}$. Now, realize that we have N possible sequences because each of the N random variables can be 1. So, we sum $p(1-p)^{N-1}$ together N times. This means, $\mathbb{P}(X = 1) = Np(1-p)^{N-1}$.

Now that we have gone through a couple specific examples, we can intuit the general functional form of the pmf. Suppose $X = m$ for some integer m between 0 and N, inclusive. Let us first find the probability of a single sequence occurring: the first m terms are successes and the last \$N - m \$ terms are failures. First recall that since the X_i's are independent of one another, the joint probability is the product of the probabilities of the individual events. Now, since the probability of a single success is p and the probability of a single failure is $1-p$, that means we take the product of p, m times, and $(1-p)$, $N-m$ times. So, $\mathbb{P}$(the first m terms are successes and the last $N-m$ terms are failures) $= p^m(1-p)^{N-m}$. Now, we have to figure out how many ways can we have m successes in N possible trials.

First, recognize that each of the X_i are independent of one another, so $\mathbb{P}(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} \mathbb{P}(X_i)$. Since there are m successes and $N-m$ failures, the probability of this occurring is $p^m(1-p)^{N-m}$. Now, we have to consider how many possible ways we can have m successes and $N-m$ failures though.

We know that there are N possible places to keep 1 success. Let us fix where that is. Now, we have $N-1$ random variables that could be the 2nd success. And then we have $N-2$ random variables that could be the 3rd success. And now notice a pattern emerging: we have $N-k+1$ possible places to keep our k^{th} success. So, when we consider all possible permutations of m successes and $N-m$ failures, we have $N(N-1)(N-2)\ldots(N-m+1)$ arrangements. But now, we have to consider that the order in which we place the successes actually does not matter. Whether X_2 was labeled as a success first or second is of no relevance to our problem since we simply care about the number of successes. We have m possible orderings for the first success, $m-1$ possible orderings for the second success, $m-2$ possible orderings for the third success, and $m-k+1$ orderings for the k^{th} success. So, the total number of combinations is actually $\frac{N(N-1)(N-2)\ldots(N-m)}{m(m-1)(m-2)\ldots1}$.

To make this more compact, let us introduce some notation on the idea of the factorial.

Definition 3.16. Factorial

*Let n be an integer. The **factorial** of n, written as $n!$, is the product of all positive integers less than or equal to n:*

$$\begin{aligned} n! &= n(n-1)(n-2)\dots 1 \\ &= \prod_{i=1}^{n} i. \end{aligned}$$

♣

Now, we can rewrite the total number of combinations as

$$\begin{aligned} \frac{N(N-1)(N-2)\dots(N-m)}{m(m-1)(m-2)\dots 1} &= \frac{N(N-1)(N-2)\dots(N-m)(N-m-1)\dots(3)(2)(1)}{m(m-1)(m-2)\dots(1)(N-m)(N-m-1)\dots(3)(2)(1)} \\ &= \frac{N!}{m!(N-m)!}. \end{aligned}$$

So, we can write the pmf of X as

$$\mathbb{P}(X=m) = \frac{N!}{m!(N-m)!} p^m (1-p)^{(N-m)}.$$

Definition 3.17. Binomial random variable

*A **binomial random variable** represents the sum of N independent and identically distributed Bernoulli trials for some integer N. We would denote X being a binomial random variable with success probability p for N independent trials as $X \sim Binomial(N,p)$. And the pmf of X is given by*

$$\mathbb{P}(X=x) = \frac{N!}{x!(N-x)!} p^x (1-p)^{N-x}.$$

♣

Let us consider a few examples to illustrate the practicality of binomial random variables.

Example 3.4 Suppose we are flipping a fair coin three times. What are the chances that we flip two heads?

First, let X be a random variable denoting the number of times we flip the coin heads-up. Because the coin flips are independent (i.e., the outcome of one coin flip does not affect the outcome of another), we have 3 independent Bernoulli trials, each with success probability $\frac{1}{2}$, where success is given by flipping the coin heads-up. Because X is the sum of the outcomes, $X \sim Binomial(3, \frac{1}{2})$. So, the probability of flipping two heads is given by

$$\begin{aligned} \mathbb{P}(X=2) &= \frac{3!}{2!(3-2)!} \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^{(3-2)} \\ &= \frac{3!}{2!1!} \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^1 \\ &= \frac{3\cdot 2\cdot 1}{2\cdot 1\cdot 1} \left(\frac{1}{2}\right)^3 \\ &= 3\left(\frac{1}{8}\right) \\ &= \frac{3}{8}. \end{aligned}$$

Now, we can confirm this answer by simplifying the problem. Let H mean that we flipped the coin

heads-up and T mean that we flipped the coin tails-up. Let a triplet of letters represent the outcome of three coin flips; for example, we would represent flipping heads then tails then heads as HTH. Now, our sample space for this problem is $\{HHH, HHT, HTH, THH, TTH, THT, HTT, TTT\}$. Notice, there are only three possibilities in our sample space of eight elements in which we have exactly two heads-up: $\{HHT, HTH, THH\}$. So, the probability of us seeing two heads is $\frac{3}{8}$ from this approach as well.

Example 3.5 Suppose we own an e-commerce business and each product at the store can be rated with a thumbs-up or a thumbs-down. If everyone has the same chance of giving a product a thumbs-up at $\frac{7}{10}$, what are the chances that at least 60 out of the 100 people who bought the product will give the product a thumbs-up? To simplify the problem, assume that one person's vote does not impact another person's review.

Let X represent the number of people who rated the product with a thumbs-up. Because we assume that votes do not affect each other and that each vote can either be a thumbs-up or not, the votes are independent Bernoulli trials. And since the probability of anyone giving a product a thumbs-up is $\frac{7}{10}$, $X \sim Binomial\left(100, \frac{7}{10}\right)$.

Now, we need to know the probability that at least 60 people give the product a thumbs-up: $\mathbb{P}(X \geq 60)$. Notice, if $X = x_1$ and $X = x_2$, then $x_1 = x_2$ because it is impossible for us to have both x_1 and x_2 thumbs-up at the same time. So,

$$\begin{aligned}\mathbb{P}(X \geq 60) &= \mathbb{P}(X = 60 \cap X = 61 \cap \ldots \cap X = 100) \\ &= \mathbb{P}(X = 60) + \mathbb{P}(X = 61) + \ldots + \mathbb{P}(X = 100).\end{aligned}$$

Now, we could either calculate the probability of each of these, or we could use `R` and simplify our work. There is a function called `pbinom` that returns the output of the cumulative distribution function (i.e., $\mathbb{P}(X \leq x)$, if given x, the success probability, and the number of trials. While it may be tempting to simply to take the answer from `pbinom`, it would be incorrect because `pbinom(60, ...)` would return $\mathbb{P}(X \leq 60)$, when we are actually interested in $\mathbb{P}(X \geq 60)$. To over come this issue, we can realize that because the probability of the entire sample space is 1,

$$\begin{aligned}\mathbb{P}(X \geq 60) &= 1 - \mathbb{P}(X < 60) \\ &= 1 - \mathbb{P}(X \leq 59).\end{aligned}$$

So, we can find the solution by calculating

So, $\mathbb{P}(X \geq 60) \approx 0.988$, which means we are almost guaranteed for at least 60 of 100 people to rate the problem with a thumbs up if our success probability is 0.7.

Now, we can also confirm our `R` answer. We can create a function that calculates the pmf $\mathbb{P}(X = x)$ and then take the sum over a sequence:

We have confirmed our answer.

```
binomial_pmf <- function(N, p, x) {
  # N represents the number of trials
  # p represents the success probability
  # x is the value at which we are evaluating the pmf: P(X = x)
```

```
  # calculate (N!)/[x!(N - x)!] * p^x * (1-p)^(N - x)
  return_value <- (factorial(N)/(factorial(x) * factorial(N - x))) *
    (p^x) * ((1 - p)^(N - x))

  return(return_value)
}

sum_pmf_60_to_100 <- 0
for(x in 60:100) {
  sum_pmf_60_to_100 <- sum_pmf_60_to_100 + binomial_pmf(N = 100, p = 0.7, x = x)
}
print(sum_pmf_60_to_100)
```

```
## [1] 0.9875016
```

3.3.1.3 Multinomial Random Variables

We can also generalize the idea motivating binomial random variables to cases in which we have more than just two outcomes. For example, suppose we are interested in not just the number of 6s and non-6s rolled in a six-sided die in N trials; what would be the pmf of a random variable that accounted for the the number of times each side was rolled? The multinomial distribution generalizes the binomial distribution to answer exactly those questions.

Recall that we first introduced the idea of the Bernoulli distribution to motivate the source of the binomial distribution. The multinomial distribution has its own version of a Bernoulli trial, known as the categorical distribution.

Definition 3.18. Categorical distribution

*Let X be a random variable following the **categorical distribution** with k outcomes for some positive integer k. Let $p_i = \mathbb{P}(X = i)$. The pmf of X is then given by*

$$\mathbb{P}(X = x) = p_1^{\mathbb{1}(x=1)} p_2^{\mathbb{1}(x=2)} \dots p_k^{\mathbb{1}(x=k)},$$

where $\mathbb{1}(\cdot)$ is the indicator function, which returns 1 if its argument is true and 0 otherwise. Let j be a possible outcome. Then, $\mathbb{1}(x = 1) = \mathbb{1}(x = 2) = \ldots = \mathbb{1}(x = j - 1) = \mathbb{1}(x = j + 1) = \ldots = \mathbb{1}(x = k - 1) = \mathbb{1}(x = k) = 0$ and $\mathbb{1}(x = j) = 1$. Since anything to the power of 0 is 1, $\mathbb{P}(X = j) = p_j$. ♣

Let X be a multinomial random variable with k categories and N independent trials. We now let X_i represent the number of outcomes in which category i was realized. In the other distributions we have seen, $\mathbb{P}(X = x)$ refers to the probability that a random variable is a scalar (e.g., $\mathbb{P}(X = 2)$). However, in the multinomial distribution, $\mathbb{P}(X = x)$ is equivalent to $\mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k)$. And similar to the categorical distribution, p_i represents the success probability for a single category i. We

leave the derivation of the pmf to the reader, but it follows a similar logic to the the binomial distribution:

$$\begin{aligned}\mathbb{P}(X = x) &= \mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) \\ &= \frac{n!}{x_1!x_2!\ldots x_k!} p_1^{x_1} p_2^{x_2} \ldots p_k^{x_k}.\end{aligned}$$

Definition 3.19. Multinomial random variable

Let X be a multinomial random variable with N independent trials and k categories such that the probability of realizing each category i is p_i. Then,

1. $\sum_{i=1}^{k} p_i = 1$*: the sum of the probabilities of class realizations must be 1*
2. $\sum_{i=1}^{k} X_i = N$*: the total number of outcomes across all classes must be equal to the number of trials*
3. *The pmf is given by*

$$\begin{aligned}\mathbb{P}(X = x) &= \mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) \\ &= \frac{N!}{x_1!x_2!\ldots x_k!} p_1^{x_1} p_2^{x_2} \ldots p_k^{x_k}.\end{aligned}$$

Notationally, X is distributed multinomially is written as

$$X \sim Multinomial(p_1, p_2, \ldots, p_k, N)$$

♣

Example 3.6 Let $D \sim Multinomial\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, 100\right)$. How many categories does this problem have? How many independent trials do we run? And what is $\mathbb{P}(D_1 = 20, D_2 = 20, D_3 = 25, D_4 = 10, D_5 = 5, D_6 = 20)$?

In order to first calculate the number of categories, we have to recall what each of the parameters of the multinomial distribution represent. We first have $p_1, p_2, \ldots, p_k$, which represent the probability of each class being realized in one trial. Since we have 6 class realization probabilities, we have 6 classes.

In the multinomial distribution's construction, after we specify the class realization probabilities, we then have the number of independent trials. In this case, we have 100 independent trials.

Lastly, to evaluate $\mathbb{P}(D_1 = 20, D_2 = 20, D_3 = 25, D_4 = 10, D_5 = 5, D_6 = 20)$, we have to find the pmf of D:

$$\begin{aligned}\mathbb{P}(D = d) &= \frac{N!}{d_1!d_2!\ldots d_6!} p_1^{d_1} p_2^{d_2} \ldots p_k^{d_k} \\ &= \frac{100!}{20!20!25!10!5!20!} \left(\frac{1}{6}\right)^{20} \left(\frac{1}{6}\right)^{20} \left(\frac{1}{6}\right)^{25} \left(\frac{1}{6}\right)^{10} \left(\frac{1}{6}\right)^{5} \left(\frac{1}{6}\right)^{20} \\ &= \frac{100!}{20!20!25!10!5!20!} \left(\frac{1}{6}\right)^{100}.\end{aligned}$$

We can now calculate this value in R:

```
print(factorial(100)/(factorial(20)*factorial(20)*factorial(25)*factorial(10)*factorial(5)*factor
```

```
## [1] 1.468638e-09
```

So there is a near 0 chance of these outcomes occurring.

Example 3.7 Suppose we are rolling a fair, six-sided die. What is the probability that in 100 rolls, we roll twenty 1s, twenty 2s, twenty-five 3s, ten 4s, five 5s, and twenty 6s?

Because we are dealing with the number of outcomes of multiple classes across independent trials, we should model this problem using the multinomial distribution. First, recall that the parameters of the multinomial distribution are the probabilities of each class being realized in one trial and the number of independent trials. Since we are working with a fair, six-sided die, we have six classes, each of which has a class realization probability of $\frac{1}{6}$. And since we roll the die 100 times, we have 100 independent trials. So, let X be the random variable we use in this problem. Then,

$$X \sim Multinomial\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, 100\right).$$

So, we want to know

$$\mathbb{P}(X_1 = 20, X_2 = 20, X_3 = 25, X_4 = 10, X_5 = 5, X_6 = 20).$$

Notice, this problem has the same exact form as Example 3.6. So, we already know the answer: the probability of these outcomes occurring is near 0. However, we will confirm this answer using R.

We previously discussed the `pbinom` function in example 3.5. There is another, very similar function for the multinomial distribution: `dmultinom`. `dmultinom` has arguments `x` and `prob`: `x` is a list that contains the number of times each class was observed in all the independent trials while `prob` contains the class realizations for the corresponding class in `x`. So, in R, we can easily find the probability of our outcome occurring:

```
print(dmultinom(x = c(20, 20, 25, 10, 5, 20), prob = c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)))
```

```
## [1] 1.468638e-09
```

The result from R is the same as what we had calculated previously in Example 3.6, confirming our computation.

Example 3.8 ?? Similar to Example 3.5, we will consider another example studying the probability of a product having certain ratings. Rather than having thumbs-up or thumbs-down ratings, we have five stars. We want to know the probability that out of 2021 ratings, we will have 103 one-star ratings, 200 two-star ratings, 505 three-star ratings, 507 four-star ratings, and 706 five-star ratings. The probability of anyone rating the product with x stars is given below:

Table 3.1: Multinomial Ratings Probability

Rating	Probability
1	0.050
2	0.100
3	0.250
4	0.250
5	0.350

Let us assume that one person's rating does not affect another's. Since the ratings are independent of one another and we have multiple categories, it makes sense to model this problem as a multinomial distribution.

Let $R \sim Multinomial(0.05, 0.1, 0.25, 0.25, 0.35, 2021)$ represent the ratings results. Using, R, we can calculate $\mathbb{P}(R_1 = 103, R_2 = 200, R_3 = 505, R_4 = 507, R_5 = 706)$:

```
print(dmultinom(x = c(103, 200, 505, 507, 706), prob = c(0.05, 0.1, 0.25, 0.25, 0.35)))
```

```
## [1] 5.694475e-07
```

So, the probability that out of 2021 ratings, we will have 103 one-star ratings, 200 two-star ratings, 505 three-star ratings, 507 four-star ratings, and 706 five-star ratings is about $5.6 \cdot 10^{-7}$.

3.3.1.4 Geometric Random Variables

3.3.1.5 Poisson Random Variables

3.3.1.6 Negative Binomial Random Variables

3.3.1.7 Normal Random Variables

3.3.1.8 log-Normal Random Variables

3.3.1.9 Exponential Random Variables

3.3.1.10 Gamma Random Variables

3.4 Expectation and Variance Operators

3.5 Approximating Expectation and Variance

Chapter 4 Probabilistic Modeling: The Language of Data Science

Data science is an interdisciplinary field that is incredibly collaborative. Given the great heterogeneity between disciplines, many times "modeling" in one field is different than "modeling" in another, so we must translate this into the language that data scientists, statisticians, and machine learners use: probabilistic models. Using, this information, we can construct the data science pipeline.

Definition 4.1. Scientific and Probabilistic Models

*A **scientific model** is a physical, conceptual, or mathematical representation of a real world system, process, or event.*

*As compared to scientific models, **probabilistic models** specifically deal with the study of how data was generated, taking advantage of randomness and variability through the assignment of probability distributions to different quantities.* ♣

Most real world problems typically deal with estimating some quantity given a set of other quantities. After gathering the data and positing a process by which the real data was produced, we calculate the likelihood function; we then maximize the likelihood function to find unknown quantities. Using our guesses for the unknown quantities, we then estimate our objective.

In general, variables are quantities that have a theoretical variation; that is, in our probabilistic model, these values come from a probability distribution. On the other hand, constants are quantities that do not have theoretical variation: they are fixed. We classify our quantities into four groups:

1. Known constants, values that are observed and have no theoretical variation
2. Unknown constants, values that are unobserved and have no theoretical variation
3. Latent/omitted variables, values that are unobserved and have no theoretical variation
4. Observations/data/random variables, values that are observed and have theoretical variation

We often refer to the theoretical results as the model statement, which describes the probabilistic process by which our data was generated and informs us on which quantities are variables and constants. The empirical information is given through the problem statement and describes what information is actually observed. Combininig the information from both sources, we will estimate our quantities.

4.1 Empirical and Theoretical Distributions

In statistical modeling, there are two ideas: what we *believe* will happen – the theoretical – and what will actually happen – the empirical. In the assumption/model statement phase of any situation, we must assume whether a quantity has variability or not. While that sounds simple enough, there may be issues that arise. Take the following example: we assume that a fair, six-sided die has a uniform distribution, so the theoretical variation is greater than 0. However, suppose we roll only 1s, a completely possible outcome. Should we classify this as a known constant or a known variable? The model statement will

guide the researcher, not the data itself. So because the theoretical variation is greater than zero, this is known variable – the empirical result has no impact on our modeling.

Consider the following two scenarios:

Let X_i be a random variable that represents the outcome of rolling a 6-sided die with 6 on **all sides**. That means that $x_i = 6$ with probability 1 (recall that the big letter means a random variable while the little letter means the realization of the random variable). Because $\mathbb{P}(X_i = 6) = 1$, X_i has no theoretical variation and is a constant. When we look at the data (i.e., the empirical distribution) and our theoretical distribution (i.e., $X_i = 6$ with probability 1), we should see that the two align.

In another scenario, suppose we have a random variable Y_i representing the outcome of rolling a 6-sided die with $1, 2, ..., 6$ on its sides, so each outcome has probability $\frac{1}{6}$. Then, $\mathbb{P}(Y_i = 1) = \mathbb{P}(Y_i = 2) = \ldots \mathbb{P}(Y_i = 6) = \frac{1}{6}$. When we actually roll the die, it is possible that we roll only a sequence of 1s (i.e. $y_1 = 1, y_2 = 1, \ldots$); or we could roll another sample like $y_1 = 1, y_2 = 2, y_3 = 1, y_4 = 6, \ldots$, in which there is empirical variation. The empirical and theoretical distributions do not necessarily have to be equivalent to one another.

The first example is a probabilistic representation of a constant: x_i will always have the same outcome. On the other hand, in the second situation, y_i is not guaranteed to be the same outcome each time *empirically*. The distinction between the empirical and theoretical variability comes from the sample outcome versus the expected outcome.

So, we want to estimate something, given that only a subset of quantities and given some assumptions about how those quantities relate. The problem and model statements will help us categorize our variables into the following table:

	Observed	**Unobserved**
Variable		
Constant		

Example 4.1 Suppose we want to estimate some quantity, τ, which is a function of $y_1 ... y_n, x_1 ... x_n, \theta$, where each i is a user and $y_1 ... y_n$ is a series of expenditures that each i has made, x_i is a bunch of covariates for each user i (i.e., age, gender, race, etc), and θ is a set of scalar quantities for the model. Our goal is to now estimate θ given $y_1 ... y_n, x_1 ... x_n$.

First, we know θ is unknown. We must ask another question: "is θ an unknown constant or an unknown variable?" This information comes from the model statement.

Suppose our probabilistic model reads as follows:

$$\text{for } i = 1, \ldots, n : y_i = \theta x_i$$

So, if given an age bracket (x_i), then y_i–the amount that user i will spend–is just some scalar multiple of their age bracket. In this model, there is no variability because we did not make any assumptions about variables having a distribution. Because we never explicitly stated which distribution θ comes from, we classify θ as a constant. The 2x2 table for this problem reads as follows:

	Observed	**Unobserved**
Variable	NA	NA
Constant	$x_1...x_n, y_1...y_n$	θ

Now, let us try a new example with the same problem setting but that will lead to a different two-by-two table.

Example 4.2 Suppose we want to estimate some quantity τ that is a function of $y_1...y_n, x_1...x_n, \theta, \sigma$, where each i is a user, y_i is a series of expenditures that each i has made, x_i is a bunch of covariates for each user i (i.e., age, gender, race, etc), and θ is a set of scalar quantities for the model. Similar to the earlier example, our goal is to estimate θ given $y_1...y_n, x_1...x_n$.

Now, suppose we posit the following model assumptions:

$$x_i \sim \mathcal{N}(0, \sigma^2)$$
$$y_i = \theta x_i$$

We know that x_i is a random quantity that has a normal distribution with population variance σ^2, which is now different from the earlier problem. Let us classify our variables as before. Because θ does not explicitly come from a probability distribution, θ is a constant.

Before, we had assumed that our x_i was not distributed. But our x_i are still observed. So, we can classify them as observed variables. Now, notice that we have a new quantity: σ^2 an observed value. Since we don't make any assumptions about its distribution, it is an observed constant.

Classifying our y_i is a little bit more complicated than before, but we will learn later on that because x_i has some variability, y_i must also have some variability, even if θ is a constant. So, y_i is an *observed variable* also.

	Observed	**Unobserved**
Variable	$x_1...x_n, y_1...y_n$	NA
Constant	σ^2	θ

We will now outline how the same *problem* statement but with different *model* statements can lead to different 2x2 tables.

Example 4.3

Consider the following problem/model statement:

$$y_i = \text{number of days } i \text{ purchases something}$$
$$z_i = \begin{cases} 1 & age(i) \geq 30 \\ 0 & age(i) < 30 \end{cases}$$

In this problem, we have four settings, each of which will lead to a different 2x2 table.

- **Example 4.4**: z_i is observed and $z_i \sim iid\ Bernoulli(p)$
- z_i is observed and no assumptions about distribution
- z_i is not given and $z_i \sim iid\ Bernoulli(p)$
- z_i is not given and no assumptions about distribution

Let's think about each of these settings individually.

Example 4.4 Consider the first problem/model statement from Example 4.3. We are given $y_1...y_n, z_1...z_n$, and our model assumptions are as follows:

$$y_i|z_i, \theta \sim Bernoulli(\theta + \alpha z_i) = \begin{cases} y_i|z_i = 1, \theta \sim Bernoulli(\theta + \alpha) \\ y_i|z_i = 0, \theta \sim Bernoulli(\theta) \end{cases}$$

Let's create the 2x2 table for this problem. We know $y_1...y_n$ is definitely observed because it is given. Now is it variable or constant? Well, it follows a distribution. And even though we don't know explicitly what $\mathbb{V}(y_i)$ is, we can calculate it as $\mathbb{V}(y_i) = \mathbb{E}(\mathbb{V}(y_i|z_i)) + \mathbb{V}(\mathbb{E}(y_i|z_i))$. Now, we know that z_i are given, *but* there is nothing about their distribution, so z_i is not variable.

	Observed	**Unobserved**
Variable	$y_1...y_n$	NA
Constant	$z_1...z_n$	θ, α

Now, let's discuss the theoretical versus empirical distributions for z_i a little more deeply:

- $\mathbb{V}(z_i) = 0$ since we made no assumptions about the distribution of v_i theoretically
- Now, if we computed the *empirical* variance:

  $$\frac{1}{n}\sum_{i=1}^{n}\left(z_i - \frac{\sum_{i=1}^{n} z_i}{n}\right)^2 > 0 \text{ (most likely).}$$

  However, the empirical variance has *no* bearing in our classification of z_i as constant or variable. What we believe z_i to be comes only from our problem and model statements. In the absence of something that explicitly states z_i has a distribution, we consider z_i a constant.

Example 4.5 Consider the second problem/model statement from Example 4.3. We are given $y_1...y_n, z_1...z_n$. But now our model is as follows:

$$z_i \sim Bernoulli(p), p = 0.4$$

$$y_i|z_i \sim \mathcal{N}(\theta + \alpha z_i, \sigma^2) = \begin{cases} y_i|z_i = 0 \sim \mathcal{N}(0, \sigma^2) \\ y_i|z_i = 0 \sim \mathcal{N}(0 + \alpha, \sigma^2) \end{cases}$$

Let's first classify all of our variables. Just as before in Example 4.4, y_i has a distribution and is given to us, so it is a known variable. Also, θ–a parameter describing the distribution for y_i–is not given and has no theoretical probability distribution, so it is an unknown constant. Likewise, α is not given and has no theoretical variability, so it too is an unknown constant. Now, notice that z_i has a distribution and is given, so we now classify it as a known variable. The 2x2 table for this problem would look as follows:

	Observed	**Unobserved**
Variable	$y_1...y_n, z_1...z_n$	NA
Constant	NA	θ, α

Example 4.6 Consider the third problem/model statement from Example 4.3. We are given $y_1...y_n$, and

our model assumptions are as follows:

$$z_i \sim Bernoulli(p), p = 0.4$$

$$y_i|z_i \sim \mathcal{N}(\theta + \alpha z_i, \sigma^2) = \begin{cases} y_i|z_i = 0 \sim \mathcal{N}(0, \sigma^2) \\ y_i|z_i = 0 \sim \mathcal{N}(0 + \alpha, \sigma^2) \end{cases}$$

Notice, the model statement is the exact same as in Example 4.5, but our model statement is now different: we do not know what $z_1...z_n$ are, so it is unknown. So, since we can calculate a *theoretical* variance for z_i, which will be greater than zero, it will be variable. Thus, z_i is an unknown variable. Other than that, it is the exact same 2x2 table as Example 4.5 with the exact same reasoning. The fact that z_i is not given has no bearing on the classification of other variables.

	Observed	**Unobserved**
Variable	$y_1...y_n$	$z_1...z_n$
Constant	NA	θ, α

Example 4.7 Consider the first problem/model statement from Example 4.3. We are given $y_1...y_n$, and our model assumptions are as follows:

$$y_i|z_i, \theta \sim Bernoulli(\theta + 2z_i) = \begin{cases} y_i|z_i = 1, \theta \sim Bernoulli(\theta + \alpha) \\ y_i|z_i = 0, \theta \sim Bernoulli(\theta) \end{cases}$$

Notice that this model statement is the exact same as in Example 4.4, but our model statement is now different: we do not know what $z_1...z_n$ are, so it is unknown. Additionally, if we were to calculate a *theoretical* variance for z_i, which we could by treating it as a random variable with probability 1, we would find that it has a variance of zero. So, it is a constant. Thus, it is an unknown constant. Besides that, the 2x2 table for this example and 4.4 are the exact same. The fact that $z_1...z_n$ is now unknown should not have any impact on our treatment of the other quantities.

	Observed	**Unobserved**
Variable	$y_1...y_n$	NA
Constant	NA	$\theta, \alpha, z_1...z_n$

Now that we know what known/unknown constants and variables are, we can start understanding the basics of the language of modeling between disciplines. These are the common terms that will represent each cell of our 2x2 table:

	Observed	**Unobserved**
Variable	Observed Random Variables	Latent Random Variables
Constant	Known Constants	Unknown Constants

4.2 Likelihood

A likelihood function is a function of the observed random variables–whatever they may be–given the constants from the problem and model statements.

Definition 4.2. Proper likelihood

*The **proper likelihood** is a representation of how well the empirical and theoretical distributions align for our observed random variables. Mathematically–with observed random variables $Y_1, \ldots, Y_n$, realizations $y_1, \ldots y_n$, and unknown constants θ–the proper likelihood is $\mathbb{P}$(observed random variables| unknown constants) $= \mathbb{P}(y_1 \ldots y_n|\theta)$.* ♣

Definition 4.3. Complete likelihood

*The **complete likelihood** is a representation of how well the empirical and theoretical distributions align for all our random variables, regardless of whether we observed them or not. Mathematically–with realizations of our observed random variable $y_1, \ldots y_n$, realizations of our unobserved random variable $x_1, \ldots, x_n$, and unknown constants θ–the complete likelihood is $\mathbb{P}$(observed random variables, latent random variables| unknown constants) $= \mathbb{P}(y_1, \ldots, y_n, x_1, \ldots, x_n|\theta)$.* ♣

So, if given a problem/model statement (or equivalently a 2x2 table), we should be able to provide the likelihood proper and the complete likelihood. If given latent random variables, then the complete likelihood is the simpler of the two. However, if we have latent random variables, we may need to integrate out the latent random variables to get the proper likelihood (i.e., the probability of our observed random variables given our constants).

Example 4.8 Reference Example 4.2 for the explicit problem and model statements. Here is the 2x2 table:

	Observed	**Unobserved**
Variable	$x_1 \ldots x_n, y_1 \ldots y_n$	NA
Constant	σ^2	θ

We have observed random variables, $x_1 \ldots x_n, y_1 \ldots y_n$, and an unknown constant, θ. So,

$$\text{Proper likelihood} = \mathbb{P}(y_1 \ldots y_n, x_1 \ldots x_n|\theta).$$

Notice, there are no latent random variables, so the complete and proper likelihoods are equivalent.

Example 4.9 Refer to Example 4.4 for the explicit problem and model statements. Here is the 2x2 table:

	Observed	**Unobserved**
Variable	$y_1 \ldots y_n$	NA
Constant	$z_1 \ldots z_n$	θ, α

Notice, we have observed random variables $y_1 \ldots y_n$ and unobserved constants $z_1 \ldots z_n$ but no latent random variables. Very similar to Example 4.8, the proper and complete likelihoods are equivalent:

$$\text{Proper likelihood = complete likelihood} = \mathbb{P}(y_1 \ldots y_n|z_1 \ldots z_n).$$

Example 4.10 Refer to Example 4.6 for the explicit problem and model statements. Here is the 2x2 table:

	Observed	**Unobserved**
Variable	$y_1 \ldots y_n$	$z_1 \ldots z_n$
Constant	NA	θ, α

We now have observed random variables $y_1 \ldots y_n$, latent random variables $z_1 \ldots z_n$, and no observed constants. The proper likelihood will still be $\mathbb{P}(y_1 \ldots y_n|\theta, \alpha)$, but its calculation is somewhat difficult because we must

account for . So, what we can do is find the *complete likelihood* and integrate out the latent variables:

$$Complete\ likelihood = \mathbb{P}(y_1 ... y_n, z_1 ... z_n | \alpha, \theta).$$

Now, can get the likelihood proper from the complete likelihood:

$$\begin{aligned} likelihood &= \mathbb{P}(y_1 ... y_n | \alpha, \theta) \\ &= \int \mathbb{P}(y_1 ... y_n, x_1 ... x_n | \alpha, \theta) dx_i \\ &= \int \text{complete likelihood } dx_i. \end{aligned}$$

Example 4.11 Let's find the likelihood for the following, very simple example:

$$y_i \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2).$$

Then,

$$\begin{aligned} likelihood &= \prod_{i=1}^{n} \mathbb{P}(y_i | \mu, \sigma^2) \\ &= \prod_{i=1}^{n} \mathcal{N}(y_i | \mu, \sigma^2). \end{aligned}$$

We assume y_i to be observed random variables, but μ and σ are unknown constants.

We can just use the assumed distributions to find the likelihood, but rarely is real-world modeling ever so simple. We can identify a series of equations that are relevant to our solution, and we can rebuild the likelihood.