Understanding Uncertainty

An Introduction to Probability and Statistics

Dylan Spicker

2024-09-03

Table of contents

Preface

What are these notes?

These notes were originally written to supplement the Winter 2024 offering of STAT 1793 at the University of New Brunswick on the Saint John campus. They have since grown into a more comprehensive resource, attempting to direct a path through introductory probability and statistics material, both for students in a Statistics program, as well those who require probability or statistics for their major. Owing to the wide range of possible audiences, not all sections of the notes will be relevant to all students. I have tried to clearly indicate requirements throughout the text.

Further, these notes remain a work-in-progress. There is additional material being added, old material being revised or clarified, and some restructuring going on. If at any point you have comments, questions, suggestions, or corrections to these notes, please do reach out!

In the writing of these notes, I am indebted to the numerous introductory texts that I have previously read, and the multitude of Statistics instructors that I have had the pleasure of studying with. As a result, I am certain that I have drawn inspiration from many sources for examples, exercises, or techniques of description. Where parallels with other work are explicit and intentional, those works are mentioned alongside the relevant passages. Where parallels are the result of internalized knowledge coming forth as inspiration, I am not able to draw direct connections. However, I will advise that the following books are all excellent resources, and each has played a part in my current approach to teaching statistics, either directly or indirectly. For introductory topics consider Engelhardt and Bain (1991); Navidi (2010); Mendenhall et al. (2013) among countless others. For problems, consider Bassett et al. (2000); Grimmett and Stirzaker (2001); Schiller, Alu Srinivasan, and Spiegel (2012).

One final note is that our coverage of concepts throughout reflects my personal biases and proclivities towards what is important in a sequence on probability and statistics. For instance, compared to others I may have a larger emphasis on probability. This is reflected both in the ordering of material (we start at Probability not at Statistics), and in the weight that each topic is given. It is my belief that you need to be able to fluently speak the language of probability in order to study statistics, and the notes begin with a strong focus on this. If these notes are being used in a two course sequence, it is my belief that the first term should consider almost exclusively probability, leaving statistical results for a second course.

Using These Notes

These notes are best viewed online. You can access them from anywhere at https://dylanspicker.github.io/Understanding-Uncertainty/index.html. Viewing online will allow you to take advantage of interactivity (such as through code blocks and solutions hiding), the search feature (visible in the left hand margin), as well as better overall formatting. There is a version of these notes in PDF format available. To download them, press the PDF icon button () next to the note title in the left hand bar. The PDF notes contain all the same content, and will be updated alongside the web notes. The formatting of the PDF document may be less visually appealing compared to the web notes. Though an effort will be made to ensure that everything remains legible, and well-formatted, there may be formatting differences. Please reach out if any components of the PDF are illegible.

Studying Statistics (and other quantitative subjects)

To successfully use these notes, I would recommend a several step procedure:

1. When reading the material, or watching lecture videos, focus on developing your understanding. Statistics is not a subject that is predominantly memory-driven. It is not about remembering the facts, formula, or question types, it is about understanding how these all come together.
    - If you get lost along the way, ask questions.
2. When you get to examples in these notes, do not look at the solutions directly. Instead, you should try to solve the examples yourself. If you get stuck, look at the solution, but only enough to see remove the current confusion, and then continue.
    - The process of learning statistics, and most quantitative subjects for that matter, relies on *doing* the work in these subjects.
    - Looking at solutions will often undermine your own understanding. It is a much different process to look at a solution and understand why it is correct, than it is to determine a solution for yourself.
3. Complete the practice questions.
    - Note: struggling with problems is a key component of learning in any mathematical discipline. It is only by struggling to understand how the pieces fit together that you will develop a proper understanding for yourself.
4. Throughout your study, make sure you are continually asking yourself how this fits together, how this confirms or contradicts what you already know, and how this may be used in your life beyond these pages.

- Drawing personal connections to the material is a very effective way of ensuring that it sticks better.

Throughout the notes there will be references to computer code in a programming language known as R. R is the most common language used for statistics by practicing statisticians, and it is a great utility for many of the types of problems we will be considering throughout these notes. It is possible to pass through the notes without writing R code, however, familiarity with it will make the process of statistics far nicer to engage with. If you are using the web notes this code can all be run directly in the browser. If you are using the PDF notes, the code and output is still provided. In either case, I would suggest getting R running on your own device, if possible. It is free to download and install. I would also suggest installing R Studio, which is a program specifically designed for writing and running R code with a lot of nice features.

Some Mathematical Background

These notes require high school mathematics to complete. Some sections of the notes require calculus (limits, derivatives, and integrals). It is possible to pass through the core of these notes skipping any sections that specifically require calculus: these are noted in text. For all sections, the assumption is that you will be comfortable manipulating mathematical expressions, solving basic equations, using a calculator, and so forth. The following are a handful of topics which will come up throughout the notes that are assumed to be known, presented here in brief as a quick reference.

Logarithms and Exponent Rules

Recall that if $a^x = b$, then we can solve for x through the use of logarithms. Specifically, $x = \log_a(b)$, where we read this as "the logarithm with base a of b." The expression means that "x is the exponent we put on a to get b." In these notes[1] we use the notation $\log(x)$ to represent the **natural logarithm**. The natural logarithm is $\log_e(x)$, where $e \approx 2.71828$ is Euler's number. The **exponential function** is given by $e^x = \exp(x)$. Which notation is used depends on the specific setting, but both are equivalent.

[1] And in most math books and courses.

Recall further the following exponent and log rules:

$$\begin{aligned} a^b \times a^c &= a^{b+c} \\ a^b \times c^b &= (ac)^b \\ a^{-b} &= \frac{1}{a^b} \\ (a^b)^c &= a^{bc} \\ a^0 &= 1 \\ \log(ab) &= \log(a) + \log(b) \\ \log(a^b) &= b\log(a) \\ \log(1) &= 0 \\ \log(e) &= 1. \end{aligned}$$

Summation and Product Notation

If we wish to add up a series of numbers, $x_1, x_2, \ldots, x_n$ then we can use summation notation to record this. Specifically,

$$x_1 + x_2 + \cdots + x_n = \sum_{i=1}^{n} x_i.$$

Here i is an indexing variable just used to keep track of which number we are currently working on. Note that the following quantities will hold

$$\begin{aligned} \sum_{i=1}^{n}(x_i + y_i) &= \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n}(x_i - y_i) &= \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} cx_i &= c\sum_{i=1}^{n} x_i. \end{aligned}$$

We can also use a similar shorthand to discuss repeated products. That is, if we want to multiply $x_1, x_2, \ldots, x_n$, then we can write

$$x_1 \times x_2 \times \cdots \times x_n = \prod_{i=1}^{n} x_i.$$

Again, i is used as an indexing variable to keep track of what value is currently being multiplied.

The following results hold

$$\prod_{i=1}^{n} x = x^n$$
$$\prod_{i=1}^{n} b^{x_i} = b^{\sum_{i=1}^{n} x_i}$$
$$\prod_{i=1}^{n} cx_i = c^n \prod_{i=1}^{n} x_i.$$

Other Important Points

The following are some additional points that are useful to know, but which do not necessarily fit together nicely.

- For any two values x and y, we get $(x+y)^2 = x^2 + 2xy + y^2$.
- The integers are the set of all whole numbers, $\{0, \pm 1, \pm 2, \dots\}$. We will often denote the set of integers using $\mathbb{Z}$.
- The natural numbers are the set of all counting numbers, either $\{0, 1, 2, 3, \dots\}$ or else $\{1, 2, 3, 4, \cdots\}$ depending on the source. We often denote the set using $\mathbb{N}$.
- The real numbers are the set of all numbers that you think of when thinking of numbers. They include the integers, and any fractions, and anything that cannot be written as a fraction (like π and e) as well. We denote the real numbers as $\mathbb{R}$.
- Intervals of real numbers are denoted as (a, b) or $[a, b]$ (or a mixture, giving $[a, b)$ or $(a, b]$). A round bracket means that the endpoint is **not** included in the set, where a square bracket means that it is. That is, a value of x is in (a, b) if $a < x < b$, where it is in $[a, b]$ if $a \leq x \leq b$.
- We write "x is in the interval (a, b)" mathematically using the $\in$ symbol. Specifically, $x \in (a, b)$.
- Infinity (∞) is not a number. Rather, it is a statement regarding the lack of a bound. When we say that something is infinity, or infinite, we simply mean that it is larger than any of the natural numbers. If you were to be given any number, then infinity will be larger than that.
- **Limits:** note that, while limits are from calculus, they are useful for formally understanding a few concepts in class. The limit of a function, $f(x)$, is simply the value that the function approaches as x approaches a specific point. So, for instance, $\lim_{x \to a} f(x)$ is the value that $f(x)$ approaches as x approaches a. If $f(a)$ exists then it is simply $f(a)$. However, $f(a)$ may not exist, at which point, we look at where the function is tending to.
- **Infinite Limits:** Of specific importance to us are infinite limits. Specifically,

$$\lim_{x \to \infty} f(x)$$

captures the beahviour of the function $f(x)$ as x gets larger and larger. In many cases, $f(x)$ will approach some specific value, which we assign to be the limiting value of the function.

Additional Resources

The following are helpful resources which may be used to supplement the material in the class. If you have any suggested resources, please feel free to send them to me, and I will include them here.

1. Open Intro Statistics.
2. Probability Course.
3. Introduction to R.
4. Learning Statistics with R.
5. Mathigon.

References

Part I

Part 1: Probability

1 Introduction to Probability

1.1 What is Probability?

At its core, statistics is the study of uncertainty. Uncertainty permeates the world around us, and in order to make sense of the world, we need to make sense of uncertainty. The language of uncertainty is **probability**. Probability is a concept which we all likely have some intuitive sense of. If there was a 90% probability of rain today, you likely considered grabbing an umbrella. You are not likely to wager your life savings on a game that has only a 1% probability of paying out. We have a sense that probability provides a measure of *likelihood*. Defining probability mathematically is a non-trivial task, and there have been many attempts to formalize it throughout history. While we will spend a good deal of time formalizing notions of probability, we first pause to emphasize the familiarity with probability that you are likely starting with.

Suppose that two friends, Charles and Sadie, meet for coffee once a week. During their meetings they have wide-ranging, deep, philosophical conversations spanning across many important topics.[1] Beyond making progress on some of the most pressing issues of our time, Charles and Sadie each adore probability. As a result, at the end of each of their meetings, they play a game to decide who will pay. The game proceeds by having them flip a coin three times. If two or more heads come up Charles pays, and otherwise Sadie pays.[2]

We can think about the strategy that they are using here and *feel* that this is going to be "fair". With two or more heads, Charles pays. With two or more tails, Sadie pays. There always has to be either two or more heads *or* two or more tails, and each is equally likely to come up[3]. The outcome of their game is uncertain before it begins, but we know that in the long run neither of the friends is going to be disadvantaged relative to the other. We can say that the probability that either of them pays is equal. It's 50-50. Everything is balanced.

[1] For instance they ask "do we all really see green as the same colour?" or "why is it that 'q' comes as early in the alphabet as it does? it deserves to be with 'UVWXYZ'?"

[2] This game, and the ensuing development relating to this game stems from a historical story at the dawn of our modern understandings of probability and chance. Specifically, questions related to this style of game were addressed by Blaise Pascal and Pierre de Fermat in letters they wrote. Their story, a fascinating read, is detailed by Devlin (2010).

[3] There has been some recent literature, see Bartoš et al. (2023), which may suggest that a coin is ever so slightly more likely to land on the same side it started on, perhaps undermining this assertion.

Now imagine that one day, in the middle of their game, Charles gets a very important phone call[4] and leaves abruptly after the first coin has been tossed. The first coin toss showed heads. Sadie, recognizing the gravity of the phone call, pays for the both of them, while realizing that Charles was well on the way to having to pay.

Example 1.1 (Basic Probability Enumeration)**.** Suppose that Sadie pays for coffee if there are two or more tails in three tosses of a coin, and Charles pays otherwise. If Charles leaves after the first coin is tossed, showing heads, what is the probability that Sadie would have had to pay?

Solution

Sadie figures that any coin toss is equally likely to show heads or tails. Because the first coin showed heads, then there are four possible sequences that could have shown up:

1. H, H, H;
2. H, H, T;
3. H, T, H;
4. H, T, T.

In three of these situations [H, H, H, H, H, T, and H, T, H] there are two heads and so Charles would have to pay. In one of them there are two tails, and so Sadie would have to pay. As a result, Charles would have to pay in 3 of the 4 (with probability 0.75) and Sadie in 1 of the 4 (with probability 0.25).

Looking at Example **??**, we can see that Sadie should likely have not paid. Only one out of every four times would Sadie have had to pay, given the first coin being heads. However, we can not be certain that, had all three tosses been observed, Sadie would *not* have paid. It is possible that we would have observed two tails, making Sadie responsible for the bill. This possibility happens one time out of four, which is more likely than the probability of rolling a four on a six-sided die[5]. Fours are rolled on six-sided dice quite frequently[6], and so it is not all together unreasonable for Sadie to have paid.

This seemingly simple concept is the core of probability. Probability serves as a method for quantifying uncertainty. It allows us to make numeric statements regarding the set of outcomes that we can observe, by quantifying the frequency with which we expect to observe those outcomes. Probability does not *remove* the uncertainty. We still need to flip the coin or roll the die to know what will happen. All probability gives us is a set of tools to quantify

[4]Someone has just pointed out the irony in the fact that there is no synonym for synonym. Technically, there are the words *metonym* and *poecilonym* and *polyonym*, but these are rarely used and Charles would wager that there is a very high probability you have never seen them.

[5]This event happens one time out of every six.

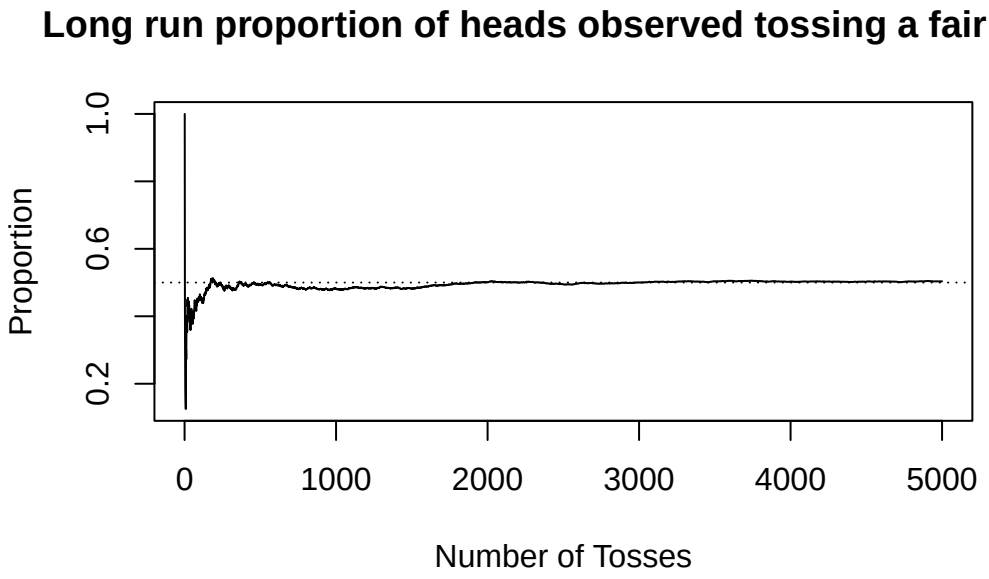
[6]Again, about one out of every six rolls

this uncertainty. These tools are critical for decision-making in the face of the ever-present uncertainty around us.

1.2 How to Interpret Probabilities (like a Frequentist)

We indicated that, intuitively, probability is a measure of the frequency with which a particular outcome occurs. This intuition can be codified exactly with the **Frequentist interpretation** of probability. According to the Frequentist interpretation (or frequentism, as it is often called), probabilities correspond to the proportion of time any event of interest[7] actually occurs in the long run. For a Frequentist, you imagine yourself performing the experiment you are running over, and over, and over, and over, and over again. Each time you answer "did the event happen?" and you count up those occurrences. As you do this more and more and more, wherever that proportion lands corresponds to the probability.

Figure 1.1: This plot simulates the repeated tossing of a coin. The x-axis represents the number of coins being tossed, and the y-axis plots the proportion of times that heads has shown up cumulatively over the tosses thus far. We can see in the long run that this proportion tends towards 0.5.



To formalize this mathematically, we first define several important terms.

Definition 1.1 (Experiment)**.** Any action to be performed whose outcome is not (or cannot) be known with certainty, before it is performed.

[7] Be that flipping heads, rolling a four, or observing rain on a given day.

Definition 1.2 (Event [Informally])**.** A specified result that may or may not occur when an experiment is performed.

Suppose that an experiment can be performed as many times as one likes, limited only by your boredom. If you take k_N to represent the number of times that the event of interest occurs when you perform the experiment N times, then a Frequentist would define the probability of the event as

$$\text{probability} = \lim_{N\to\infty} \frac{k_N}{N}.$$

If you have never taken a calculus course, and thus are unfamiliar with the concept of limits, that is not a barrier to understanding this statement. When you see a statement of the form

$$\lim_{x\to\infty} f(x),$$

simply think "what is happening to the function $f(x)$ as x grows and grows (off to ∞)?"

In practice this means that, in order to interpret probabilities, we think about repeating an experiment many, many times over. As we do that, we observe the proportion of time that any particular outcome occurs, and take that to be the defining relation for probabilities. The reason that we say the probability of flipping heads is 0.5 is because if we were to sit around and flip a coin[8] over, and over, and over again, then in the long-run we would observe a head[9] in 0.5 of cases.

Example 1.2 (Probability Interpretation)**.** How do we interpret the statement "the probability that Sadie would have had to pay, given a head on the first toss, is 0.25"? Recall that in the game, they toss three coins, and Sadie pays if two of them show tails.

Solution

This statement means that, if Sadie were to repeatedly be in the situation where one head has shown and there are two coins left to toss, then in 0.25 of these situations (in the limit, as this is repeated an infinite number of times) will end up showing two tails.

Many situations in the real world cannot be run over and over again. Think about, for instance, the probability that a particular candidate wins in a particular election. There is uncertainty there, of course, but the election can only be run once. What then? There are several ways through these types of events.

First, we can rely on the **power of imagination**. There is nothing stopping us from envisioning the hypothetical possibility of running the election over, and over, and over, and over again. If we step outside of reality for a moment, we can ask "if we could play the day of the

[8] Our experiment.
[9] Our event.

election many, many, many times, what proportion of those days would end with the candidate being elected?" If we say that the candidate has a 75% chance of being elected, then we mean that in 0.75 of those imagined worlds, the candidate wins. It is crucial to stress that in our imagination here, we need to be thinking about the **exact same day** over and over again. We cannot imagine a different path leading to the election, different speeches being given in advance, or different opposition candidates. If we start from the same place, and play it out many times over, what happens in each of those worlds?

This repeated imagining is not for everyone. As a result, alternative proposals to the interpretation of probability have been made, with the **Bayesian interpretation** (or subjectivist interpretation) being particularly prominent. To Bayesians, probability is a measure of subjective belief. To say that there is a 50% chance of a coin coming up is a statement about one's knowledge of the world. The Bayesian view, built around subjective confidence in the state of the world, can be formalized mathematically as well. A Bayesian considers the *prior evidence* that they have about the world[10] and combine this with current observations in order to update their subject beliefs, balancing these two sources of information.

Remark (Bayesian Probabilities and Belief Updating). Suppose that a Bayesian is flipping a coin. Before any flips have been made the Bayesian understandably believes that the coin will come up heads 50% of the time. However, when the coin starts to be flipped, the observations are a string of tails in a row.

After having flipped the coin five times, the individual has observed five tails. Of course, it is totally possible to flip a fair coin five times and see five tails, but there is a level of skepticism growing.

After 10 flips, the Bayesian has still not seen a head. At this point, the subjective belief is that there is likely something unfair about this coin. Even though the experiment started with a baseline assumption that the coin was fair the Bayesian no longer believes that the next flip will be a head.

As this goes on, you can imagine the Bayesian continuing to update their view of the world. To them, the probability is an evolving concept, capturing what was believed and what has been observed.

For the election example, the Bayesian interpretation is somewhat easier to think through. To say that a candidate has a 75% chance of winning the election means that "based on everything that has been observed, and any prior beliefs about the viability of the candidates, the subjective likelihood that the candidate wins the election is 0.75". If we disagree about prior beliefs, or have experienced different pieces of information, then we may disagree on the probability. That is okay.

In these notes, we focus on the Frequentist interpretation. This choice is not a strong stance on the relative merits of the two viewpoints. There is some research to suggest that Frequentist

[10]Any relevant evidence that has previously been collected.

interpretations are fairly well understood by the general public[11]. However, it is important to know and recognize that there is a world beyond Frequentist Probability and Statistics, one which is very powerful once it is unlocked.[12]

If probability measures the long term proportion of time that a particular event occurs, how can we go about computing probabilities? Do we require performing an experiment over and over again? Fortunately, the answer is no. The tools of probability we will cover allow us to make concrete statements about the probabilities of events without the need for repeated experimental runs. However, before completely dismissing the idea of repeatedly running an experiment, it is worth considering a tool we have at our disposal that renders this more possible than it has ever been: computers.

1.3 R Programming for Probability and Statistics

Throughout these notes we will make use of a programming language for statistical computing, called R. Classically, introductory statistics courses involved heavy computation of particular quantities by hand. The use of a programming language (like R) frees us from the tedium of these calculations, allowing for a deeper focus on understanding, explanation, decision-making, and complex problem-solving. This is **not** to say we will *never* do problems by hand[13], however, these notes emphasize the use of statistical computing. The sections on R programming throughout the notes are self-contained, and separate from the main material. Studying these sections will give you familiarity and comfort with reading, modifying, and writing R scripts for statistical programming.

It may be useful to have R open alongside the notes to ensure that you can get the same results that are printed throughout. In this section we will cover the very basics of using R, and reading R code. If you are interested, there are plenty of resources to becoming a more proficient R programmer.

1.3.1 Basic Introduction to R Programming

When programming, the basic idea is that we are going to write instructions in a **script** which we will tell our computer to execute. These instructions are[14] performed one-by-one, from the top to the bottom of the script. We can have instructions which operate on their own, or

[11] See, for instance Gigerenzer et al. (2005), where they study how people interpret the statement "there is a 30% chance of rain tomorrow." Interestingly, most people can convert this into a frequency statement (3 out of 10, say), even if the specific meaning is sometimes lost. They conclude that there are other issues in attempting to understanding this statement, issues which we will address later on.

[12] More on this in later books, if you so desire!

[13] To the contrary, some amount of by-hand problem-solving helps to solidify these concepts.

[14] Typically. There are some exceptions to this, but if this is your first time programming, you need not worry about that!

which interact with previous (or future) instructions to add to the complexity. The trick with programming then is to determine which actions you need the computer to perform, in which order, to accomplish the task that you are setting out to do.

To begin, we may consider an R program that uses the programming language as a basic calculator.

```
## [1] 33554420
```

Here, we ask the computer to perform arithmetic operations. We use `+` for addition, `-` for subtraction, `*` for multiplication, `/` for division, and `^` for exponentiation. With the use of parentheses, any expressions relating to these basic operations can be performed. Note that here the result is output after it is computed. Try modifying the exact expressions being computed, allowing you to feel comfortable with these types of mathematical operations.

```
## [1] 33554420
## [1] 40
```

Here, we have two lines of math running with arithmetic operations. Each is output when it is computed. These two results have no ability to interact with one another, and if we were to add more and more lines beneath, the same would continue to happen.

If we want different commands to be able to interact with one another, we need a method for storing the results. To do so, we can define **variables**. In R, to define a variable, we use the syntax `variable_name <- variable_value`. We can choose *almost* anything that we want for the variable name[15] and the variable value can also be of many types.[16] The arrow between the two is the **assignment operator**. It tells R to assign the `variable_value` to be accessible from the `variable_name`.

```
## [1] 5
## [1] 8
```

In this code we assign the variable `my_5` to contain the value 5, and the variable `my_8` to contain the value 8. We can output these as expressions themselves by typing the variable name. Directly outputting these variables is not of particular interest, however, we can use the variables in later statements by including the variable name in them.

```
## [1] 40
```

[15] The variable name must start with a letter and can be a combination of letters, numbers, periods, and underscores.

[16] more on this later

Here, instead of outputting the variables, we multiply them together. We could have used `5 * 8` in this case for the same result, however, we are afforded a lot more flexibility with this approach. Much of this flexibility comes from our capacity to *change* the values of variables over time. Consider the following script, and try to understand why the output is the way that it is.

```
## [1] 40
## [1] 80
```

At the top point in the script, before the first `my_5*my_8` call, the variable `my_5` has the value `5`. However, after this is called, the value is updated to be `10`. Then, when we call `my_5*my_8` again, this is now simplified to `10 * 8`, giving the result. Perhaps more importantly, we can take the value of an expression and assign that to a variable itself.

```
## [1] 40
```

Here, we define another variable. This time, `result` now contains the result of multiplying our previous two variables. Thus, when we output it, we get the same value. Take a moment to read through the following script, and try to understand what will happen at the end.

The result is 1600. Why? We can read through this step-by-step in order to understand this. First we set our variables to have the values of `5` and `8`. Then, `result` is made to be the product of our two variables, which in this case is `40`. After that, we set the value of `my_5` to be the same as the value of `result`, which gives `my_5` equal to `40`. At this point, `result` equals `40`, `my_5` equals `40`, and `my_8` is equal to `8`. The next line updates `my_8` to be the same as the value of `my_5`, which we just clarified was `40`. As a result, all the variables we have defined now take on the value of `40`. The final line before output, `result <- my_5 * my_8` updates the value of `result` to be the product of the two variables again, this time giving `40 * 40` which gives 1600.

1.3.2 Function Calls in R

Up until this point we have only used numerical operations and variable assignment. While this allows R to serve as a very powerful calculator, we often want computers to do much more than arithmetic. As a result, we need to explore **functions** in R. A function is a piece of code which takes in various arguments and outputs some value (or values). Most of the way that we will use R in these notes is through the use of function calls.[17] This is exactly analogous to a mathematical function: it is some rule which maps input to output. In fact, some of the most basic functions in R are functions which relate to mathematical functions.

[17] Note: when you begin to write programs for yourself, a lot of your time will be spent writing custom functions. If this is of interest to you, I suggest looking into programming more! For now, we will not need to define our own functions.

```
## [1] 22026.47
## [1] 3.162278
## [1] 2.302585
```

1. Computes the exponential function applied to `x`, that is $e^x = e^{10}$.
2. Computes the square root of `x`, that is $\sqrt{x} = \sqrt{10}$.
3. Computes the natural logarithm of `x`, that is $\log(x) = \log(10)$.

The basic format of a function call will always be `function_name(param1, param2, ...)`. Each of these functions required only a single parameter, however, there are some functions which take more than one parameter. If we have decimal numbers, for instance, we may wish to round them. To do so, we can use the `round` function in R, which takes two parameters: the number we wish to round, and how many digits we wish to keep.

```
## [1] 3.142
```

There are a few things to note about this sequence of function calls. First, notice that we assign the output of a function to a new variable. This behaves exactly as we saw above with numeric calculations. Next, consider that the output of the function[18] has a value of 3.142. That is: we rounded the value to 3 decimal points, exactly as would be expected. Finally, notice that the second parameter passed to the function call is **named**. That is, instead of calling `round(unrounded_value, 3)`, we have `round(unrounded_value, digits = 3)`. If you had made the first call this would have worked perfectly.[19] However, R also provides the ability to pass in parameter names alongside the parameter values with the syntax `function_name(param_name = param_value, ...)`.

The benefit to doing this is two-fold. First, it is easier to read what is happening, especially for function calls that you have never seen before. Second, it removes the need to have the parameters ordered correctly. It is best practice to **always** include parameter names where you can.

Now, you may be wondering "how do you know what the parameter names should be?" The names are built-in to the different functions that you are working with, and with experience you will become quite familiar with them. However, at any point you can also run the command `?function_name`, replacing `function_name` with the name of a function you are interested in, to open documentation about that function. There you will see not only the names of the different parameters, but useful information regarding what the function does, and examples of how to call it.

When we run this code, we are given *a lot* of information. We can see the function name, the details, how it's called, and so forth. In the **Arguments** section we get a list of all the arguments we can pass to the function, along with a description of them. In this case we see

[18] Which we have stored in the variable `rounded_value`
[19] Try it out to convince yourself!

that `round` can take a parameter called `x` for the number to be rounded, and `digits` (which we have previously seen).

```
## [1] 3.142
```

This code produces the exact same output, where now both parameters are named. Specifically, we could read this function call as "round the value of `x` to have `digits` decimal points." If you had instead written `round(3, unrounded_value)`, you would get the value `3`, since now we are rounding `3` to `3.141592` decimal points.

1.3.3 Moving Beyond Numeric Data

So far, everything that we have looked at has been numeric data. We have seen integers, and decimals. You can have negative results, say by taking `my_var <- -5`. And while numbers are frequently useful, we will require further types of data to write useful computer programs. In these notes, we will focus on three additional data types: textual data which (referred to as **strings**), true and false binary data (referred to as **logicals** in R), and lists of the same data type (referred to as **vectors**). They will behave in much the same way as numeric data, with different functions and techniques which can be applied to them.

To define a string of text, we encapsulate the text that we are interested in within quotation marks (either single `'` or double `"` quotation marks will work).

```
## [1] "This is a string."
```

Two commonly used functions which rely on strings are `paste` and `print`. Each will take in strings as input, and they do not need to be named. The function `paste` can take in as many strings as you would like. It will "paste" together all the strings provided, creating a longer string out of these. The function `print` will display the string that is passed as output. Until now, we have been running these programs in a way where all calls are displayed as output: this will not always be the case, and so print can come in handy there.

```
## [1] "Hello! Welcome to R programming, Dylan !"
```

Note that `paste` has several additional options which can be investigated in the documentation. This is only the simplest use case for it. In general, strings are particularly helpful when we wish to have output from the computer that will be human-readable. Where strings are largely for humans, logicals are largely for computers.

Much of what computer programming entails is checking whether certain conditions hold, and then taking different actions depending on what is found. In order to do this, the computer

needs a way to represent true and false statements. In R, these are codified with the values `TRUE` and `FALSE`. Note, the capital letters here make a difference. You cannot use `true` or `True` or any other combination thereof. More important than being able to specify the values `TRUE` and `FALSE` directly is the ability to detect whether certain statements are `TRUE` or `FALSE`. For this we require comparison operators.

If we think of mathematical comparisons we can state whether two things are equal, or not equal, and whether one thing is less than (or equal to) or greater than (or equal to) another. We can run all of these same checks in R.

- To check whether two quantities are equal you use `quantity_1 == quantity_2`. This statement will be `TRUE` if `quantity_1` and `quantity_2` are exactly the same, and will be `FALSE` otherwise.
- To check whether two quantities are not equal, you can use `quantity_1 != quantity_2`. This statement will be `TRUE` if the quantities differ from one another.
- To check whether one quantity is larger than another, you can use `quantity_1 > quantity_2`. If you want to know whether it is greater than *or* equal to, you can use `quantity_1 >= quantity_2`.
- To check whether one quantity is smaller than another, you can use `quantity_1 < quantity_2`. If you want to know whether it is less than *or* equal to, you can use `quantity_1 <= quantity_2`.

```
## [1] TRUE
## [1] FALSE
## [1] FALSE
## [1] TRUE
## [1] FALSE
## [1] FALSE
## [1] FALSE
## [1] TRUE
## [1] TRUE
## [1] FALSE
## [1] TRUE
## [1] TRUE
```

There are two key points to note beyond this. First, we will of course not normally compare two constants to one another. We already know that `5==5`, so we would not need to check it. We can, however, plug-in variables, perhaps with unknown values, and have the same types of statements being made. Second, the checks for equality and inequality also work with other data types (like strings).

```
## [1] TRUE
```

```
## [1] FALSE
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] FALSE
## [1] FALSE
## [1] TRUE
## [1] TRUE
## [1] FALSE
```

The final checks may be slightly odd. Here we are comparing across different types of data. When we do this R will automatically try to convert from one type to the other. With strings and numbers this is not too challenging. If they can be converted nicely between types, then they are and the values are compared. Otherwise, R will conclude they are not equal by default. For logicals, it is important to note that `TRUE == 1` and `FALSE == 0`. We will often use these values interchangeably.

The final data type that we will consider are vectors. Vectors store multiple values, of the same type, in a single object. Thus, we may have a vector of numeric data, or a vector of strings, or a vector of logicals. The vectors will always contain the same type throughout, but they are stored in a single object (and as such, for instance, can be stored in a single variable). To define a vector we call `c(...)`, where the `...` contains the set of objects we want to store in the vector. The `c` stands for **c**oncatenate, as we are *concatenating* together the set of items into a single container.

```
## [1] "vector"  "of"      "strings"
## [1] 3 1 4 1 5
## [1]  TRUE  TRUE FALSE  TRUE
## [1] FALSE  TRUE FALSE
```

We see that each of these vectors holds one type of object. Vectors can be of arbitrary and different lengths. It is also possible to combine multiple vectors *of the same type* into one, by using the `c` function again.

```
##  [1] 3 1 4 1 5 9 2 6 5 3
```

Here we combine different numeric vectors together. We also show, when forming `combined_v3`, how numeric vectors can have single items added onto them. That is, if you have a single number, it can be treated as a vector with one element in it. This becomes very useful when building up vectors within code.

In addition to combining multiple vectors together, we can also select elements out of a vector. To do this, we use a set of square brackets after the vector's name, with a number within those square brackets specifying the **index** of the vector we are interested in. The index is the element position starting at 1[20] and running to the length of the vector. We can include a vector of indices to select multiple elements at once.

```
## [1] "D"
## [1] "Y"
## [1] "L"
## [1] "A"
## [1] "N"
## [1] "D" "Y" "L" "A" "N"
```

Note that, each element is selectable individually, giving a single item of that type (in this case, strings). If you select multiple of the elements together, it will create a vector of that type (in this case, a string vector). In addition to selecting elements in this way by their indices, you can also update the elements in the same way.

```
## [1] 2 0 2 3
## [1] 2 0 2 4
```

In this example we are changing the last element of the vector. Sometimes we may not know how long the vector actually is, if for instance, it is being built-up as our code runs. If we ever want to check the length of a vector, we can call the function `length` which takes as input only one vector, and outputs the numeric value of its length.

```
## [1] 5
```

1.3.4 Program Control Flow

We have seen different types of data, different ways of manipulating data, functions, and variables so far. In order to bring all of these concepts together into useful programs we need some way to control the flow of our programs. We have seen that, by default, programs execute from the top until the bottom. However, it will often be the case that we want to have certain code running only if certain conditions hold, or that we want to repeat some piece of code many times over. To accomplish these tasks we require **control flow statements**. We will

[20] If you have programmed in the past there is a good chance the language you have learned is "0 indexed" rather than "1 indexed". In R, all vectors start at position 1 and count up, which is not the case in many languages. Be careful of this.

consider only two types of control flow statements now, which will serve well enough to read most of what needs to be read for these notes.

The first type of statement is the **conditional statement**. Conditional statements execute only when certain conditions hold. The simplest conditional statement is an **if** statement. The format to define an if statement is `if (condition){ ... }`, where `condition` is some logical condition to be evaluated. If the condition is `TRUE` then the code contained in { … } is evaluated. If the condition is `FALSE` it is not.

```
## [1] "My number is a positive."
```

In this case, these conditional statement check to see whether the number we entered is larger than zero and whether the number we entered is smaller than zero, respectively. When run, notice that at most one of the statements is executed.[21] In this case, we know that only one (or neither) of these statements can be true. When that is the case it may make sense to make use of `else` clauses in our conditional logic.

```
## [1] "My number is not a positive."
```

Here, if the number is greater than `0`, then we run the first block of code, otherwise we run the second block of code. Thus, whenever a positive number is entered, we see "My number is a positive", and whenever a non-positive number is entered, we see "My number is not a positive." We can extend `else` blocks to be `else if` blocks, where further conditions can be specified.

```
## [1] "My number is a large positive value."
```

In these case we can pass through each conditional statement in order. First, is the number larger than `50`? If so, print the statement, otherwise we check the next condition, is the number greater than `0`? Note that if we are checking this condition we *know* that the number is less than or equal to `50` since it failed the first check. We continue through the rest of this procedure down until the last else block. This block is run only when all of the other conditions fail: that is, our value is not larger than `50`, or larger than `0`, or smaller than `-50`, or smaller than `0`. The only value that satisfies this is `0` itself.

Sometimes, we wish to check compound conditions. That is, we want to know whether multiple conditions hold, or perhaps, whether at least one of many conditions hold. These statements can be converted into "and" and "or" statements, respectively. To denote "and" statements we use `&&` and to denote "or" statements we use `||`. Thus, the check `my_val > 0 && my_val < 100` returns true only if `my_val` is both above `0` **and** below 100. The check `my_val < -50 || my_val > 50` returns true whenever **either** `my_val` is less than `-50` **or** `my_val` is greater than `50`.

[21] In fact, if `my_number` is set to `0` then none of the statements are executed.

```
## [1] "Either 5 or a negative value."
```

Conditional statements can grow to be very complex, however, with these rules you can read through them top-to-bottom, substituting for "and" and "or" where necessary. It is also possible, where required, to place one conditional statement inside the code block for another, and to combine them with any of the other techniques that we have learned thus far.

The final piece of control flow that we will consider for now is the `for` loop. The idea with a `for` loop is that we want to repeat the same action either a certain number of times, or for every item in a set of items. To do so, we use the syntax `for(x in vector){...}`, where the code in `...` will be performed once for every single item in the vector. Within the code block specified by `...`, the value `x` will take on the current value in the loop.

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

Notice that three values are printed, in order, `1` then `2` then `3`. The loop code is run three different times, one for each element in the list. Each time the loop is running the next value from the list gets assigned to the variable `x`. The first time it runs it gets the first element, and so forth. As a result, we can use these values in our calculations in whatever way we need to.

```
## [1] "The square of 1 is 1"
## [1] "The square of 2 is 4"
## [1] "The square of 3 is 9"
```

Whenever we are trying to form a numeric vector with consecutive elements, as we are in `c(1,2,3)`, we can make this easier on ourselves by specifying the upper and lower bounds of the range, separated by a colon. That is `c(1,2,3) == 1:3`. This is often useful when specify a loop as we very often want to repeat something a set number of times. Note that we do not ever *need* to use the value of the looping variable. Sometimes, we just want things to repeat, and so the loop is a convenient way to do that.

```
## [1] "This will get printed 5 times."
## [1] "This will get printed 5 times."
## [1] "This will get printed 5 times."
## [1] "This will get printed 5 times."
## [1] "This will get printed 5 times."
```

1.3.5 Reading Through a More Complex R Program

Take a moment to read through the following R program and try to understand what is happening exactly. There are comments throughout which will assist in the parsing of the script. We have seen comments up until now, without drawing explicit attention to them. In R, anything placed after a `#` on the line is considered a comment. The programming language ignores these and so they are only there to help other individuals who may be reading through. It is good practice to comment your code to help others, and also to help yourself whenever you return to it in the future. In these notes code will typically be commented. If you are reading the PDF version of the notes often these comments will be annotations beside the code (numbering certain lines) with the comments provided below, for the sake of legibility.

Note, this combines everything that we have learned, and it is entirely understandable if it takes some time to process. Fortunately, you can always try running the script yourself, and playing with different components of it. Remember, if you do not know what a function call does, you can use the documentation[22] and try playing with it some yourself. To help with the interpretation here, note that this is an implementation of the game that Charles and Sadie have been playing.

```
## [1] "Now starting round 1"
## [1] "The flip was a H which benefits Charles"
## [1] "Now starting round 2"
## [1] "The flip was a T which benefits Sadie"
## [1] "Now starting round 3"
## [1] "The flip was a T which benefits Sadie"
## [1] "Sadie has scored enough points to win."
## [1] "After flipping the coin 3 times, Charles scored a total of 1 points"
## [1] "while Sadie scored a total of 2 points."
## [1] "As a result Sadie won the game and will not have to pay!"
```

1.3.6 R Programming for Probability Interpretations

Recall that the motivation for the discussion of R was the Frequentist interpretation of probability. Computers are very effective at repeatedly performing some action. As a result, we can use computers to mimic the idea of repeatedly performing an experiment. Consider the case of flipping a coin over and over again.

We can use `sample(x, size)` as a function to select `size` realizations from the set contained in `x`. Thus, if we take `sample(x = c("H","T"), size=1)` we can view this as flipping a coin one time. If we use the loop structure we talked before, then we can simulate the experience of repeatedly flipping a coin. Consider the following R code. Note, any time that we are doing

[22] The internet is also a wonderful resource, one which even very experienced developers make frequent use of.

something which is randomized in R (such as drawing random samples) we also will make use of the `set.seed()` function. This function takes in an integer value as an argument, and by providing the *same* integer value we can make sure to always get the same random numbers generated.[23] This helps to ensure the repeatability of any R analysis, and it is good practice to do. To see what happens without seeding, try modifying the following code without a seed, and running it several times. Then, set the seed (to any number you like) and do the same process.

```
## [1] 0.522
```

1. A seed ensures that the random numbers generated by the program are always the same. This helps to be able to reproduce our work.
2. This is how many times we want to repeat the experiment.
3. This is where we are going to store the results of our tosses. It creates an empty list for us.
4. Here we are going to loop over the experiments, one for each run.
5. This is our coin toss. We are going to sample 1 from either 'H' or 'T'
6. If the coin toss is heads, then we add a 1 to the list. Otherwise, we add a zero to the list.
7. Return the mean of all of the tosses.

It is worth adjusting some of the parameters within the simulation, and seeing what happens. What if you ran the experiment only 5 times? Ten thousand times? What if instead of counting the number of heads, we wanted to count the number of tails? What if we wanted to count the number of times that a six-sided die rolled a 4? All of these settings can be investigated with modifications to the provided script.

References

[23] Technically, we cannot use a computer to generate random numbers. We can only generate *pseudo random* numbers, which are close enough for most purposes.

2 The Mathematical Foundations of Statistical Experiments

2.1 The Sample Space and Events

In Chapter 1 we saw the mathematical formulation for the Frequentist interpretation of probability. To study probability, we require a more detailed mathematical model. We want a description, framed in terms of mathematical objects, which will allow us to work out probabilities of interest. In general, to form such a probability model we need both a list of all possible outcomes that the experiment can produce, as well as the probabilities of these outcomes.

We call the list of outcomes that can occur from an experiment the **sample space** of the experiment. The sample space is denoted $\mathcal{S}$, and is defined as the set of all possible outcomes from the experiment. For instance, if the experiment is flipping a coin we have $\mathcal{S} = \{\mathrm{H}, \mathrm{T}\}$. If the experiment is rolling a six-sided die then $\mathcal{S} = \{1, 2, 3, 4, 5, 6\}$.

Definition 2.1 (Sample Space)**.** The sample space of a statistical experiment is the set of all possible outcomes that can be realized from that experiment. The sample space is typically denoted $\mathcal{S}$, or with similar script letters.

Example 2.1 (Enumerating Sample Spaces)**.** Write down the complete sample space, $\mathcal{S}$ for the game that Sadie and Charles play, based on flipping and observing a coin three times in sequence.

Solution

For Sadie and Charles their experiment involves tossing a coin three times in sequence. As a result each outcome is a three-dimensional list of values, given for instance by $(\mathrm{H}, \mathrm{H}, \mathrm{H})$. We can write the full sample space as

$$\mathcal{S} = \{(\mathrm{H}, \mathrm{H}, \mathrm{H}), (\mathrm{H}, \mathrm{H}, \mathrm{T}), (\mathrm{H}, \mathrm{T}, \mathrm{H}), (\mathrm{H}, \mathrm{T}, \mathrm{T}), (\mathrm{T}, \mathrm{H}, \mathrm{H}), (\mathrm{T}, \mathrm{H}, \mathrm{T}), (\mathrm{T}, \mathrm{T}, \mathrm{H}), (\mathrm{T}, \mathrm{T}, \mathrm{T})\}.$$

With the sample space formally defined, we can revisit Definition **??**, and formally define the concept of an event.

Definition 2.2 (Event)**.** An event is any collection of outcomes from a sample space for a statistical experiment. Mathematically, an event, E, is a subset of $\mathcal{S}$, and we write $E \subset \mathcal{S}$.

Take for instance the experiment of a single coin. In this case, we may define $E_1 = \{\mathrm{H}\}$, $E_2 = \{\mathrm{T}\}$, and $E_3 = \{\mathrm{H}, \mathrm{T}\}$ as examples of possible events. Here, E_1 corresponds to the event that a head is observed, E_2 corresponds to the event that a tail is observed, and E_3 corresponds to the event that either a tails or a heads was observed. Note that for each event we have $E_1 \subseteq \mathcal{S}$, $E_2 \subseteq \mathcal{S}$, and $E_3 \subseteq \mathcal{S}$.[1]

Example 2.2 (Basic Event Listing)**.** List several events from the game that Charles and Sadie are playing. Indicate why these are events.

Solution

Recall that an event is any subset of the sample space. In Example **??** we define $\mathcal{S}$ for this game. As a result we can take sets which contain any combinations of these elements. For instance $E_1 = \{(\mathrm{H}, \mathrm{H}, \mathrm{H})\}$, or $E_2 = \{(\mathrm{H}, \mathrm{H}, \mathrm{T}), (\mathrm{H}, \mathrm{T}, \mathrm{H})\}$, or

$$E_3 = \{(\mathrm{H}, \mathrm{H}, \mathrm{H}), (\mathrm{H}, \mathrm{H}, \mathrm{T}), (\mathrm{H}, \mathrm{T}, \mathrm{H}), (\mathrm{H}, \mathrm{T}, \mathrm{T}), (\mathrm{T}, \mathrm{H}, \mathrm{H}), (\mathrm{T}, \mathrm{H}, \mathrm{T}), (\mathrm{T}, \mathrm{T}, \mathrm{H}), (\mathrm{T}, \mathrm{T}, \mathrm{T})\}.$$

These are all events since $E_1 \subseteq \mathcal{S}$, $E_2 \subseteq \mathcal{S}$, and $E_3 \subseteq \mathcal{S}$.

Example 2.3 (Event Identification)**.** Is "Charles has to pay" an event from the game that Charles and Sadie are playing? Why? Explain.

Solution

"Charles has to pay" is not directly an event, as it is not a subset of the sample space. This could plausibly be seen as a real-world description of a possible event, but it is not *itself* an event. The strictness of this language will be relaxed as time goes on, however, it is important to familiarize yourself with how descriptions are converted to events. In this case, we could take an event to be

$$E = \{(\mathrm{H}, \mathrm{H}, \mathrm{H}), (\mathrm{H}, \mathrm{H}, \mathrm{T}), (\mathrm{H}, \mathrm{T}, \mathrm{H}), (\mathrm{T}, \mathrm{H}, \mathrm{H})\},$$

and state that if E occurs then Charles has to pay.[2]

Example 2.4 (Defining Events from Real-World Descriptions)**.** What event corresponds to the description "Sadie has to pay" in the game that Charles and Sadie are playing? Recall

[1] Note that the symbol $\subseteq$ is the *subset or equal* symbol. If we write $A \subseteq B$, then A is either a subset of B or else A is equal to B. This is in contrast to using $A \subseteq B$ which suggests that A is not equal to B. The distinction is analogous to the difference between writing $x < y$ versus $x \leq y$. Throughout these notes we will predominantly use $\subseteq$.

[2] When speaking to a statistician, they would understand "Charles has to pay" as an event that *can* occur based on the defined sample space, by simply transforming it into the language of the sample space. However, the distinction is important to make: events are always subsets of the sample space. Once this is second nature, it is a rule that can be loosened, as the knowledge can always be fallen back on when needed. Simply put: you need to know the rules in order to break them!

that they flip a coin three times, and Charles will pay if at least two heads come up, while Sadie will pay if at least two tails come up.

Solution

Sadie will have to pay whenever there are two or more tails. As a result we can enumerate the possible outcomes that leads to Sadie paying. We have

1. (T, T, T);
2. (H, T, T);
3. (T, H, T);
4. (T, T, H).

Any other outcome will have fewer than two tails, and as a result, Sadie will not have to pay. Thus, to form an event, we consider the set with each of these outcomes in it. This gives

$$E = \{(\mathrm{T}, \mathrm{T}, \mathrm{T}), (\mathrm{H}, \mathrm{T}, \mathrm{T}), (\mathrm{T}, \mathrm{H}, \mathrm{T}), (\mathrm{T}, \mathrm{T}, \mathrm{H})\}.$$

While the events $E_1 = \{\mathrm{H}\}$ and $E_2 = \{\mathrm{T}\}$ each correspond to a simple outcome from the sample space, $E_3 = \{\mathrm{H}, \mathrm{T}\}$ corresponds to a combined event. We call direct outcomes **simple events** and more complex outcomes like E_3 **compound events**.

Definition 2.3 (Simple Event)**.** A simple event is any event which corresponds to exactly one outcome from the sample space. A simple event only has one way of occurring. The size of the set for a simple event will be 1. The sample space, in turn, is made up of a collection of simple events.

Definition 2.4 (Compound Event)**.** A compound event is any event which corresponds to more than one outcome from the sample space. A compound event can occur in multiple different ways. The size of the set for the compound event will be greater than 1.

If we consider rolling a six-sided die, then an example of a simple event is that a four shows up, denoted $\{4\}$. A compound event could be that an even number is rolled, $\{2, 4, 6\}$, or that a number greater than or equal to four is rolled, $\{4, 5, 6\}$.

Example 2.5 (Identifying Simple and Compound Events)**.** List an example of (at least) one simple and one compound event from the game that Charles and Sadie are playing.

Solution

An example of a simple event would be $E_1 = \{(\mathrm{H}, \mathrm{H}, \mathrm{H})\}$ since it is comprised of exactly one outcome. If three heads are rolled, this event occurs, there is no other way for it to

occur. An example of a compound event would be

$$E_2 = \{(\mathrm{H}, \mathrm{H}, \mathrm{H}), (\mathrm{T}, \mathrm{H}, \mathrm{H}), (\mathrm{H}, \mathrm{T}, \mathrm{H}), (\mathrm{H}, \mathrm{H}, \mathrm{T})\}.$$

Here there are four outcomes that correspond to this event, and if any of those outcomes are observed the event occurs.

We say that an event "occurs" if any of the outcomes comprising the event occur. As a result we can have more than one event occurring as the result of a run of a statistical experiment. Suppose that we are rolling a fair, six-sided die. Consider the events "an even number was rolled" and "a number greater than or equal to four was rolled." If a four or a six are rolled, both of these events happen simultaneously. Our goal when working with probability will be to assign probability values to different events. We will talk about how likely, or unlikely, events of interest are, given the underlying statistical experiment.

Above, we defined E_3 to be equal to $\mathcal{S}$. As a result, we can say that $\mathcal{S}$ *is* an event since $\mathcal{S} \subseteq \mathcal{S}$. This is the event that any outcome is observed, which is certain to happen. Since it is certain to happen, we know it happens with probability 1. There is another "special" event which is important to consider. We call this the *null event*. Denoted $\emptyset$, the null event is an event that corresponds to "nothing in the sample space". We know that every time an experiment is run something in the sample space occurs, and so the null event is assigned probability zero.

Definition 2.5 (Null Event)**.** The null event, denoted $\emptyset$ or $\{\}$, is an event from a statistical experiment which corresponds to nothing within the sample space. The null event has probability zero, and it is impossible to observe. Note that, no matter the sample space, $\emptyset \subset \mathcal{S}$.

2.2 Set Operations for Event Manipulation

Ultimately, all events are sets. These sets are subsets of the sample space, and can contain single or multiple outcomes. Every quantity that we are interested in can be expressed as some set of outcomes of interest. In building up these sets it is common to construct them through the use of "and", "or", and "not" statements. That is, we may say that our event occurs if some outcome **OR** another outcome occurs, or perhaps our outcomes occurs if some outcome does **NOT** occur.[3]

Consider the example of drawing cards from a standard 52-card deck. In such a deck there are 13 card ranks, and four card suits, with one of each combination present. If we draw a single card we can think of the outcomes of the experiment as being any of the 52 possible

[3]While both *or* and *not* language is likely clear from the examples we have seen so far, *and* language may be slightly less obvious. While we will explore this in more depth shortly, note that you could not have two simple events occurring simultaneously. If E_1 and E_2 are both simple events, then you can have E_1 **or** E_2, and you can have **not** E_1, but you cannot have E_1 **and** E_2. This is *not* true for compound events.

combinations of rank and suit. We are often interested in an event such as "the card is red", which is the same as saying "the card is a heart **or** the card is a diamond." Perhaps we want to know whether the card was an ace through ten, this is the same as saying "the card is **not** a Jack **or** a Queen **or** a King." If we are interested in the event that the ace of spades was drawn, this can be expressed by saying that "the card was a spade **and** the card was an ace."

As you begin to pay attention to the linguistic representation of events that we use, you will notice more and more the use of these words to form compound events in particular. As a result, we give each of them a mathematical operation which allow us to quickly and compactly express these quantities in notation.

Example 2.6 (Description of Events)**.** Describe "Charles has to pay", based on the game Charles and Sadie are playing, using language revolving around "or", "and", and "not" each. That is, describe observing at least two heads, on three flips of a coin, one time using "or", one time using "and", and one time using "not".[4]

Solution

There are many possible ways of doing so. Consider the following:

1. **OR**: Two heads are observed OR three heads are observed.
2. **AND**: Not two tails are observed AND not three tails are observed.
3. **NOT**: Not more than one tails are observed.

We define mathematical operations to encapsulate the use of **or**, **and**, and **not**. These operations apply to any mathematical sets, whether they refer to events or not.

Definition 2.6 (Union)**.** The union encodes the use of "or" in reference to two or more sets. Formally, with two sets A and B, the union of A and B is the set of all elements that are contained in A, or B, or both A and B. We write $A \cup B$ and read that as A union B. When we wish to take the union of many sets, $A_1, A_2, \ldots, A_n$, we write this as

$$A_1 \cup A_2 \cup \cdots \cup A_n = \bigcup_{i=1}^{n} A_i.$$

Definition 2.7 (Intersection)**.** The intersection captures the use of "and" in reference to two or more sets. Formally, the intersection of two sets, A, and B, is the set that contains all elements that belong to both A and B. We write $A \cap B$, and say "A intersect B." When we wish to take the intersection of many sets, $A_1, A_2, \ldots, A_n$, we write

$$A_1 \cap A_2 \cap \cdots \cap A_n = \bigcap_{i=1}^{n} A_i.$$

[4]It may be helpful to notice that you can mix and match these terms to your hearts content!

Definition 2.8 (Complement)**.** The complement makes formal the concept of "not." The complement of a set is the set of all elements which occur in the sample space but are not in the given set. We write this as A^C and say "A complement." When dealing with a sample space, $\mathcal{S}$, the complement of A is the set of all elements in $\mathcal{S}$ that are not in A.

Example 2.7 (Basic Set Operations)**.** For the game being played by Charles and Sadie, take $E_1 = \{(\mathrm{H},\mathrm{H},\mathrm{H})\}$, $E_2 = \{(\mathrm{H},\mathrm{H},\mathrm{H}),(\mathrm{T},\mathrm{H},\mathrm{H}),(\mathrm{H},\mathrm{T},\mathrm{H})\}$, and $E_3 = \{(\mathrm{H},\mathrm{H},\mathrm{T})\}$. Express the following events.

a. $E_1 \cup E_2$;
b. $E_1 \cap E_2$;
c. E_2^C;
d. $E_2 \cap E_3$;
e. $E_1 \cup E_2 \cup E_3$.

Solution

Directly from definitions we can write down each of the following sets:

a.
$$E_1 \cup E_2 = \{(\mathrm{H},\mathrm{H},\mathrm{H}),(\mathrm{T},\mathrm{H},\mathrm{H}),(\mathrm{H},\mathrm{T},\mathrm{H})\} = E_2.$$
As a result, the union of E_1 and E_2 is simply E_2.[5]

b.
$$E_1 \cap E_2 = \{(\mathrm{H},\mathrm{H},\mathrm{H})\} = E_1.$$
As a result, the intersection of E_1 and E_2 is simply E_1.[6]

c.
$$E_2^C = \{(\mathrm{H},\mathrm{H},\mathrm{T}),(\mathrm{H},\mathrm{T},\mathrm{T}),(\mathrm{T},\mathrm{H},\mathrm{T}),(\mathrm{T},\mathrm{T},\mathrm{H}),(\mathrm{T},\mathrm{T},\mathrm{T})\}.$$

d. For $E_2 \cap E_3$ note that they share no elements. As a result, the intersection will be empty since there are no elements common to both of them. This gives $E_2 \cap E_3 = \emptyset$.

e.
$$E_1 \cup E_2 \cup E_3 = \{(\mathrm{H},\mathrm{H},\mathrm{H}),(\mathrm{T},\mathrm{H},\mathrm{H}),(\mathrm{H},\mathrm{T},\mathrm{H}),(\mathrm{H},\mathrm{H},\mathrm{T})\}.$$

Definition 2.9 (Disjoint Events)**.** Two events, E_1 and E_2 are said to be disjoint whenever their intersection is the null event. That is, if $E_1 \cap E_2 = \emptyset$ then E_1 and E_2 are disjoint events.

These concepts allow us to more compactly express sets of interest, and in particular, will be quite useful when it comes to assigning probability. The more times you work with the set operations, the more familiar they will become, and as a result, practice is always useful.

[6]Note that whenever we have two events, A and B, with $A \subseteq B$, then $A \cup B = B$.
[6]Note that whenever we have two events, A and B, with $A \subseteq B$ then $A \cap B = A$.

Consider rolling a 6-sided die, and take A to be the event that a 6 is rolled, B to be the event that the roll was at least 5, C to be the event that the roll was less than 4, and D to be the event that the roll was odd.

- If we consider D^C this is the event that the roll was even;
- $A \cup C$ is the event that a 6 was rolled or that a number less than 4 was rolled, which is to say anything other than a 4 or a 5, which we may also express as $\{4, 5\}^C$.
- If we take $A \cup B$ then this will be the same as B, and $A \cap B$ will be A.
- If we take the event $A \cap C$, notice that no outcomes satisfy both conditions, and so $A \cap C = \emptyset$.
- We can also join together multiple operations. $D^C \cap C$ gives us even numbers than less than 4, which is to say the outcome 2.
- Similarly, $(A \cap B)^C$ would represent the event that a number less than 6 is rolled.

Example 2.8 (Set Operations with Decks of Cards)**.** Charles and Sadie are tiring of flipping their coin, and so they wish to start using decks of cards sometimes instead. Before they formalize a game based on decks of cards, they want to make sure that they are both very comfortable working with these. Suppose that the sample space is defined to be the set of 52 standard cards that may be drawn on a single draw from the deck. Describe how set operations can be used to form events corresponding to:

a. A red card is observed.
b. Any card between an ace and a ten is observed.
c. The ace of spades is observed.

Solution

First, we define several events. Note, these can be defined in shorthand to prevent needing to write out many different cards. we take D to be the event that a diamond is observed, we take H to be the event that a heart is observed, take S to be the event that a spade is observed.[7] Take A to be the event that an ace is observed, J to be the event that a Jack is observed, Q to be the event that a Queen is observed, and K to be the event that a King is observed.[8] We can use unions, intersections, and complements to express the previously mentioned scenarios.

a. To represent outcomes corresponding to "the card is red", we can use $D \cup H$.
b. To represent outcomes corresponding to "an ace through ten", we can use $(J \cup Q \cup K)^C$.
c. To represent the outcome "the ace of spades", we may use $A \cap S$.

Working with these basic set operations should eventually become second nature. There are often very many ways of expressing the same event using these different operations, and finding

[8]Note, these three are compound events with 13 different outcomes contained within them.
[8]These are all compound events with four different options.

the most useful method of representing a particular event can often be the key to solving challenging probability questions. The first step in making sure that these tools are available to you is in ensuring that the basic operations are fully understood, and this comes via practice. Remember, unions represent "ors", intersections represent "ands", and complements represent "nots".

2.2.1 Using R To Represent Sample Spaces and Events and Performing Set Operations

We have seen how R can encode sets of elements using vectors. For instance, we may take `sample_space <- 1:6` to represent the sample space of rolling a six-sided die. We can form events by taking subsets of the relevant quantities, selecting via indices. Fortunately, there are also all of the basic set operations implemented in R.

We can use `union(x, y)` to perform the union of `x` and `y`, `intersect(x, y)` to perform the intersection of `x` and `y`, and `setdiff(x = sample_space, y)` to perform the complement of `y` (assuming that `sample_space` contains the full sample space).[9]

```
# Define the Sample Space of Rolling a 20 Sided Die
sample_space <- 1:20

# Define some Events
E1 <- sample_space[2]
E2 <- sample_space[c(1, 3, 5, 7)]
E3 <- sample_space[c(1, 2, 4, 8)]

# Consider Set Operations
union(x = E1, y = E2) # E1 union E2 = {1, 2, 3, 5, 7}
union(x = E1, y = E3) # E1 union E3 = {1, 2, 4, 8}

intersect(x = E1, y = E3) # E1 intersect E3 = {2}
intersect(x = E1, y = E2) # E1 intersect E2 = {}

setdiff(x = sample_space, y = E1) # E1 complement

# (E2 union E3) complement
setdiff(x = sample_space, y = union(E2, E3))
## [1] 2 1 3 5 7
## [1] 2 1 4 8
```

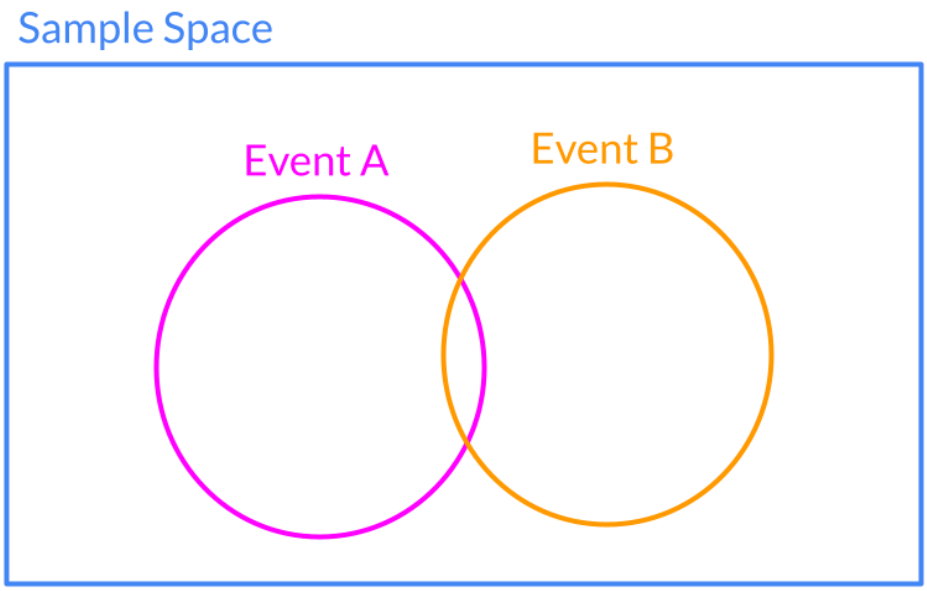
[9] R does not implement complements directly, and instead implements the set difference operation. The set difference function, `setdiff(x, y)` returns the set of all elements in `x` which are not in `y`, a sort of subtracting of sets. The complement of a set is defined to be $A^C = \text{setdiff}(\mathcal{S}, A)$, indicating why this works!

```
## [1] 2
## integer(0)
##  [1]  1  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  [1]  6  9 10 11 12 13 14 15 16 17 18 19 20
```

2.3 Venn Diagrams

The sample space is partitioned into outcomes, and the outcomes can be grouped together into events. These events are sets and can be manipulated via basic set operations. Sometimes it is convenient to represent this process graphically through the use of Venn diagrams. In a Venn diagram, the sample space is represented by a rectangle with the possible outcomes placed inside, and events are drawn inside of this as circles containing the relevant outcomes.

Figure 2.1: A basic Venn diagram, representing the sample space and two different events. In practice, the sample space would have the possible outcomes written into the rectangle, and the circled events would end up containing the relevant outcomes for those events.

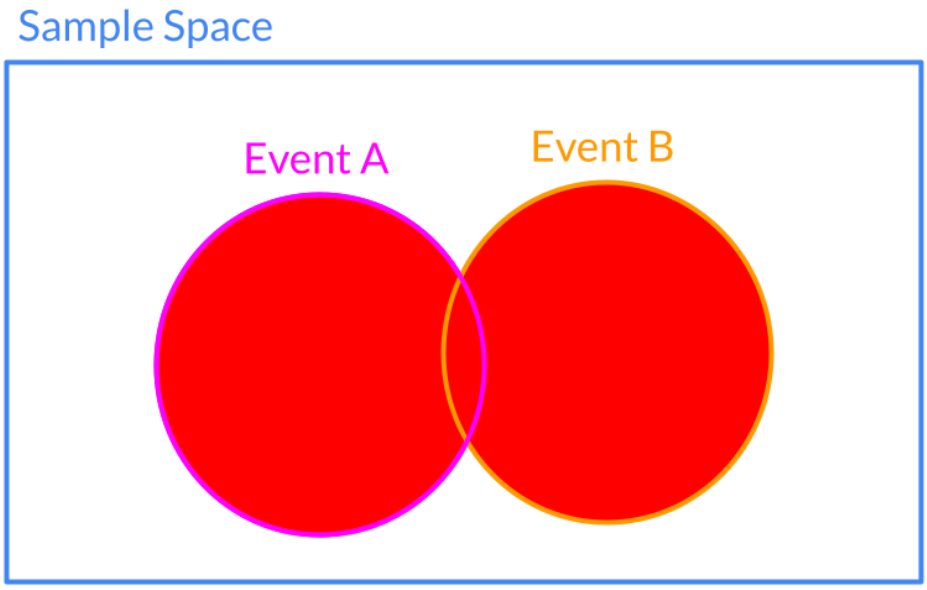


On the Venn diagram then, the overlap between circles represents their intersection, the combined area of two (or more) circles represents their union, and everything outside of a given circle represents the complement. This can be a fairly useful method for representing sample

spaces, and for visualizing the basic set operations that we use to manipulate events inside the sample spaces.

A word of caution: Venn diagrams are useful tools, but they are not suitable as mathematical proofs directly. It is possible to convince yourself of false truths if the wrong diagrams are used, and as a result, Venn diagrams should be thought of as aids to understanding, rather than as a rigorous tool in and of themselves.[10]

Figure 2.2: **Union:** The union of events A and B is shaded here in red. The union of two sets is all of the contents of both sets, including the overlap between the two.



[10]This is a general principle in mathematics. Coming up with one example that makes something *seem* true does not form an argument demonstrating that it *is* true. Venn Diagrams should largely be thought of as specific examples of the underlying phenomena, which are great if you're a visual learner!

Figure 2.3: **Intersection:** The intersection of events A and B is shaded here in red. The intersection of two sets is all of the content shared by both sets, given by the overlapping area of the two circles.

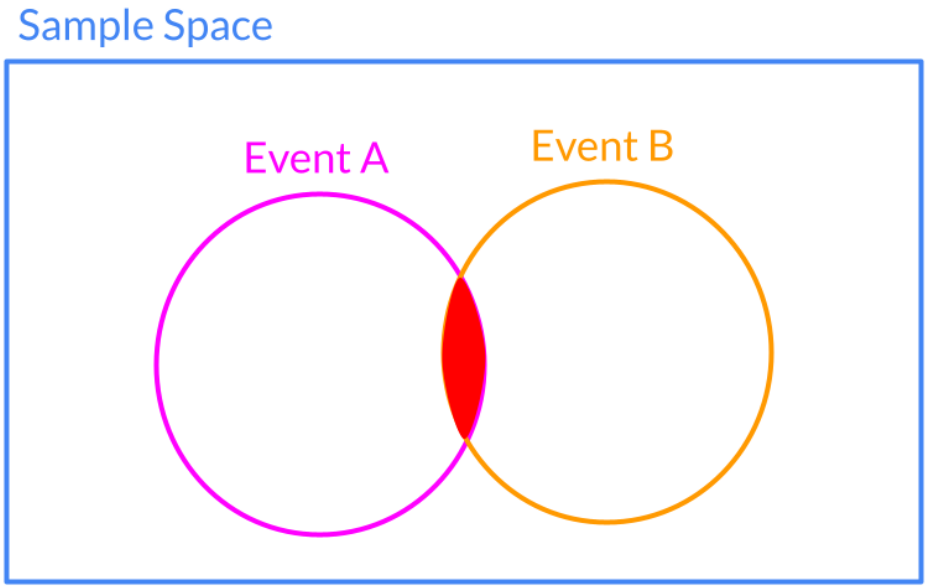
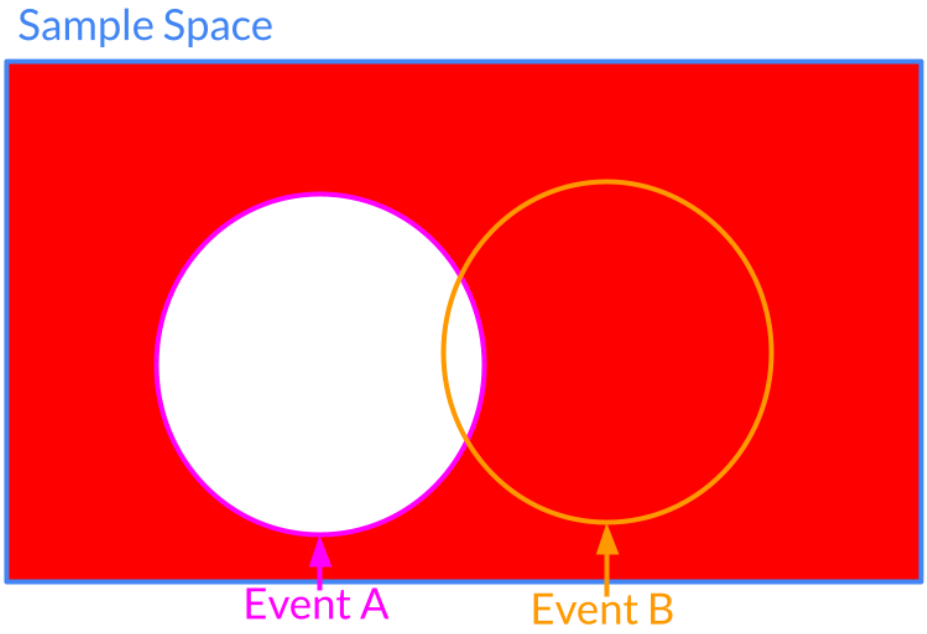


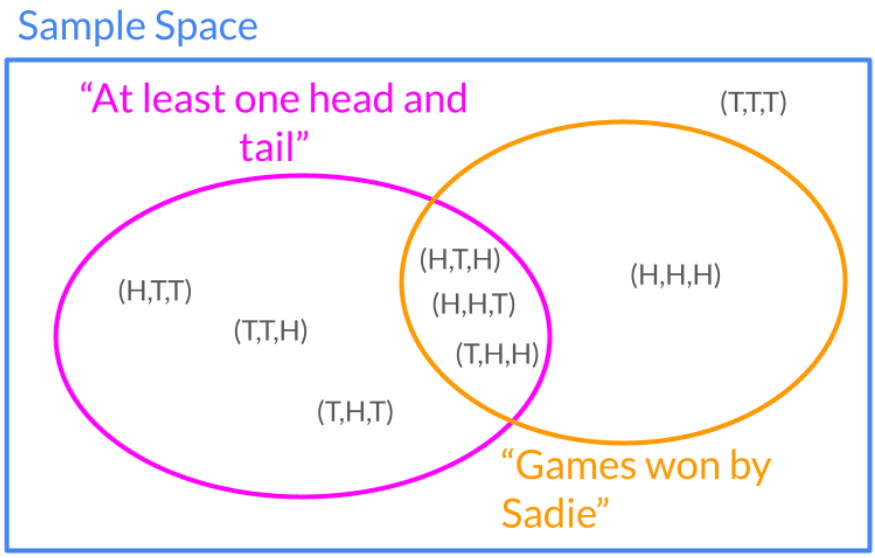
Figure 2.4: **Complement:** The complement of event A is shaded here in red. The complement of a sets is all of area inside of the sample space, not inside of the set. Here we show the complement of Event A, though Event B would be similar.



Example 2.9 (Venn Diagram with Defined Events)**.** Draw a Venn diagram representing the original game that Charles and Sadie played. On the diagram draw the events corresponding to "At least one head **and** one tail are observed", and "Sadie won the game". Recall that three coins are tossed, and Sadie wins if at least two of them show heads.

Solution

The sample space contains the eight possible options. Only $(\mathrm{T}, \mathrm{T}, \mathrm{T})$ does not belong to at least one of the events. Both events share $(\mathrm{H}, \mathrm{H}, \mathrm{T})$, $(\mathrm{H}, \mathrm{T}, \mathrm{H})$, and $(\mathrm{T}, \mathrm{H}, \mathrm{H})$.



Sample spaces, events, and the manipulation of these quantities forms a critical component of understanding probability models. In particular, they describe the complete set of occurrences in a statistical experiment that we could be interested in assigning probability values to. To formalize a probability model, however, we also need some rule for assigning probability values.

Exercises

Exercise 2.1. For each of the following experiments, describe the relevant sample space and identify one possible event of interest.

a. The quality control inspection of smartphone screens from a manufacturing process.
b. Monitoring the ongoing structural integrity of a newly built bridge.
c. A clinical trial studying the effectiveness of a new drug.

d. Epidemiological monitoring of a disease outbreak.
e. Dealing a hand of black jack.
f. Observing the launch conditions for a rocket launch.
g. Debugging in software development.
h. Playing the lottery.

Exercise 2.2. A card is drawn at random from an ordinary deck of 52 playing cards. Let A be the event that a king is drawn, and B the event that a club is drawn. In words, describe the following events.

a. $A \cup B$;
b. $A \cap B$;
c. $A \cup B^C$;
d. $A^C \cup B^C$;
e. $(A \cap B) \cup (A \cap B^C)$.

Exercise 2.3. Suppose that $\mathcal{S} = \{\phi, \lambda, \Delta, \mu\}$. List all possible events from the corresponding experiment.

Exercise 2.4. Suppose an experiment is run which generates realizations from positive integers. Take A to be the event $\{1, 5, 31, 56, 101\}$, $B = \{22, 56, 5, 103, 87\}$, $C = \{41, 13, 7, 101, 48\}$, and D to be the event that the number is odd. Identify (write down or describe) each of the following events.

a. D^C
b. $A \cap B$
c. $C \cup A$
d. $C \cap D$
e. $(A \cup B) \cup (C \cup D)$
f. $A \cap D^C$

Exercise 2.5. Suppose that a 20 sided die is rolled. The events of interest are: A the outcome is a multiple of 4, and B the outcome is a multiple of 5.

a. Draw a Venn Diagram representing the sample space and events.
b. Identify the event $A \cup B$. What does this correspond to in words?
c. Identify the event $A \cap B^C$. What does this correspond to in words?
d. How would you denote the event "neither a multiple of 4 nor 5.'' in terms of A and B?

Exercise 2.6. Suppose that two indistinguishable coins are flipped. The events of interest are: A exactly two heads are seen, and B at least one head is seen.

a. Draw a Venn Diagram representing the sample space and events.

b. Describe every possible outcome with respect to the identified events.
c. Give an event, in terms of the number of heads observed, which is equivalent to the sample space.

Exercise 2.7. A cinema has 12 screens, numbered 1 through 12. Before opening, an employee checks to ensure that the projectors are correctly calibrated.

Let A be the event that all the screens are correctly calibrated, B be the event that the third screen is not correctly calibrated, C be the event that exactly one screen is not correctly calibrated, and D be the event that 5 and 8 are correctly calibrated.

Which of the following pairs of events are disjoint?

a. A and B.
b. B and D.
c. C and D.
d. B and C.

3 The Core Concepts of Probability

3.1 Assigning Probabilities (and The Equally Likely Outcome Model)

There are a plethora of ways to assign probabilities to different events. At the most basic level any rule that maps from the space of possible events to real numbers between 0 and 1 can be used as rules for probability assignment. That is, probability assignment is a set of rules which says "for this event assign this probability."

Example 3.1 (Coin Toss Probabilities)**.** Suppose that the fair coin used by Charles and Sadie is tossed one time. Write down the probability assignments relating to this experiment.

Solution

In this case we have $\mathcal{S} = \{\text{H}, \text{T}\}$. Thus, the possible events for which we need to assign probabilities are $\emptyset$, $\{\text{H}\}$, $\{\text{T}\}$, and $\{\text{H}, \text{T}\} = \mathcal{S}$. For any probability model we have $P(\emptyset) = 0$ and $P(\mathcal{S}) = 1$. When we say that a coin is "fair" we are saying that $P(\text{T}) = P(\text{H})$, and since these are the only two possible outcomes in the sample space, we must have that they each have probability 0.5.

Not every assignment of probability values is going to be valid. Suppose, for instance, that we have a six-sided die, each side labelled with a number from one to six. If I told you that there was a probability of 0.5 that it comes up 1, 0.5 that it comes up 2, 0.5 that it comes up 3, 0.5 that it comes up 4, 0.5 that it comes up 5, and 0.5 that it comes up 6, you would probably call me a liar.[1] If, as we have previously seen, probabilities represent the long run proportion of time that a particular event is observed, we cannot have 6 different outcomes each occurring in half of all cases.

Beyond the restrictions that we impose to form "valid" probability rules, we have another concern: scalability. It is perfectly acceptable to indicate that in an experiment with 3 outcomes, the first has a probability of 0.25, the second of 0.3, and the third of 0.45. What if the experiment has 100 possible outcomes? Or 1000? It quickly becomes apparent that enumerating the probabilities of each event in the sample space is not an efficient way of assigning probabilities in practice. A core focus of our study of probability will be finding techniques that allow us

[1] Or else conclude that I was mistaken and maybe should not be teaching probability.

to efficiently encode probability information into manageable objects. Once we have done this we will be in a position where we can manipulate these (comparatively) simple mathematical quantities in order to make statements and conclusions about any of the events of interest, even if they have never been explicitly outlined as having an assigned probability.

While we will consider myriad methods for accomplishing these goals throughout our study of probability, we begin with a very useful model which simplifies probability assignment, without any added complexity, and creates a solid foundation for us to explore the properties of probability models. We start by considering **equally likely outcomes.** As the name suggests, the probability model considering equally likely outcomes assigns an equal probability to every possible outcome of the experiment. This is a probability model that we are already distinctly familiar with: flipping a coin, rolling a die, or drawing a card are all examples of experiments which rely on the equally likely outcomes framework.

Remark (Statisticians and Urn Models). In statistics and probability courses and books you will often have instructors or authors using fairly simple models to illustrate probability concepts. There will often be questions relating to coin tosses, and dice, and decks of cards, and everyone's favourite: urns. It will very frequently be the case that a statistics question will state that there is an urn with some combination of coloured balls within it, from which you will be selecting some number either with or without replacement. The frequency of these types of examples and questions often feels disconnected from the refrain that "uncertainty is all around us" and that "statistics is relevant to every aspect of our world!"[2] Why is it that we seldom see questions or examples that are directly tied to these wide spread applications of the lessons and techniques being taught?

In part these simple experiments are cleaner to handle than "real world" situations. We can easily assume that a die is fair and that takes care of any unsuspecting wrinkles that will necessarily come along with the "real world". This is not dissimilar to working under the assumption of frictionless surfaces in introductory physics, or assuming that human beings are rational in economics. Another key point is that most of us have deep familiarity with dice, and coins, and cards.[3] The same is not going to be true of stories that are derived from different use cases in the real world. A final important point, and this will be something we see in depth in the coming chapters, is that from a statistical point of view: there is no difference. Once we have the tools to work with these quantities, we have the tools to work with any of the quantities. This actually distinguishes the use of these types of examples in statistics and probability from those for other subjects: at no point is anything that we are learning incorrect, or overly simple - we are just focusing on the raw probabilistic nature of the phenomenon. As a result, we will continue to see these simple models in these notes. I would encourage you,

[2] One of the most famous quotes from a statistician was a thought shared by John Tukey, stating "The best thing about being a statistician is that you get to play in everyone's backyard." This is a common refrain, and one rooted in truth. Statistics is everywhere, across every field of human inquiry, and can help us make sense of everything from the trivial to the deeply important.

[3] This does not help to explain why we use urns so much, of course. When was the last time any of us drew a ball from an urn?

whenever possible, to hold a topic in mind that matters more to you and start trying to draw the parallels between rolling dice, and whatever it is that you may care about.

Why urns, specifically? Well, whether it be coin flipping or dice rolling or card selection, we can model this equivalently using an urn (with 2, 6, and 52 items, respectively). The urn becomes more flexible to *exactly* dictate what the probability of any selection will be, which is a useful way of moving from equally likely models (each ball is equally likely to be selected) to arbitrary models (we can have however many identical balls in the urn as we would like).

If we have an experiment with a sample space $\mathcal{S}$ which has $|\mathcal{S}| = k$ total elements[4], then each element of the sample space occurs with probability $\frac{1}{k}$. In the case of the coin toss example, $\mathcal{S} = \{\text{H}, \text{T}\}$, and so $k = 2$ and each outcome occurs with probability $\frac{1}{2}$. In the case of drawing a card at random, there are 52 different outcomes, and so $k = 52$, and the probability of drawing any particular card is $\frac{1}{52}$.

It is critically important to recognize that the equal probability model assigns equal likelihood to the possible outcomes of an experiment, not the possible events of interest. It will not be the case that all events have the same probability. To make this concrete, consider the events A "the ace of spades is drawn" and B "any spade is drawn". It is clear that B happens more frequently than A, even though we have said that this is an experiment with equally likely outcomes. Remember: an outcome is an observation from a single experimental run, an event is any collection of these possible outcomes.

A core goal is then bridging the gap between the probability of an outcome[5] and the probability of an event. In order to do so, we will next consider the rules of probability, introducing properties that are required for valid probability assignments, and the techniques for manipulating probabilities to calculate the probabilities of quantities of interest.

3.1.1 Using R for the Equally Likely Probability Model

In the previous chapter we saw how we can codify sample spaces and events using vectors in R. In the introduction we actually saw how we can sample from a sample space using the equally likely outcome framework. Specifically, an application of the `sample` function will draw a set number of values from a sample space, giving each value an equal probability to be drawn.[6]

[4]Note that, when we have a set, using the absolute value symbols $|\cdot|$ stands for the **cardinality** of the set. Cardinality is just a fancy way of saying the size or the number of elements that the set has in it.

[5]A quantity which in the equally likely outcome framework, we know exactly.

[6]The `sample` function can also be used without equally likely events by specifying a vector of probabilities, however, this is a less common use case.

```r
# Define the Sample Space of Rolling a 20 Sided Die
sample_space <- 1:20

# Recall that whenever we wish to perform an experiment in R with
# randomness, we should call set.seed
set.seed(31415)

# The sample function takes three main parameters:
#   x: the sample space
#   size: the number of items to draw
#   replace: a logical (TRUE/FALSE) representing whether the
#            draws should be with replacement or not.
one_roll <- sample(x = sample_space, size = 1)
ten_rolls_with_replacement <- sample(x = sample_space,
                                     size = 10,
                                     replace = TRUE)
ten_rolls_without_replacement <- sample(x = sample_space,
                                        size = 10,
                                        replace = FALSE)

one_roll
ten_rolls_with_replacement
ten_rolls_without_replacement
## [1] 2
##  [1] 19 17 14  3  5 12  2 15  9  3
##  [1] 18  8 16  9  7 20  2 19 10 13
```

3.2 The Axioms of Probability

We have previously seen that not every probability assignment can be valid. For instance, assigning 0.5 probability to each outcome on a die leads to a nonsensical scenario. With just a little imagination, we can conjure equally nonsensical scenarios in other ways. For instance, it would make very little sense to discuss the probability of an event being a negative value. What would it mean for an event to occur in a negative proportion of experimental runs? Alternatively, we can consider two events that are nested in one another: say event A is that we draw the ace of spades, and event B is that we draw any spade. Every single time that A happens, we know that B also happens. But there are ways that B can occur where A does not.[7] If I told you the probability of A was 0.5 and the probability of B was 0.2, this would

[7]For instance, the Queen of spades being drawn.

violate our base instincts. How can it be more likely to draw the ace of spades than it would be to draw any spade at all?[8]

Often in mathematics when we have an intuitive set of rules[9] that particular quantities must obey, we work to add formality through defining properties of these concepts. To this end, we can define the key properties that probabilities must obey in order to be well-defined, valid probabilities. With three fairly basic properties, we can completely specify what must be true in order for a set of probabilities to be "valid", and to in turn align with our intuitions.

The Axioms of Probability

1. **Unitary:** Every valid set of probabilities must assign a probability of 1 to the full sample space. That is, $P(\mathcal{S}) = 1$. This is an intuitive requirement as every time the experiment is run we observe an outcome in the sample space. As a result, in every experimental run the event $\mathcal{S}$ occurs.
2. **Non-negative:** We require that every probability is non-negative. We can have probabilities of 0, but we can never have a probability less than zero. Again, this is sensible[10] but is important to include in our formalization. Specifically, for every event E, we must have $P(E) \geq 0$.
3. **Additivity:** the final property requires slightly more parsing on first pass. Suppose that we define a sequence of events, $E_1, E_2, E_3, ...$ such that no two events have any overlap. That is, $E_j \cap E_\ell = \emptyset$ for all $\ell \neq j$. Then, the final property we require for probabilities is that

$$P(E_1 \cup E_2 \cup E_3 \cup \cdots) = P\left(\bigcup_i E_i\right) = \sum_i P(E_i) = P(E_1) + P(E_2) + P(E_3) + \cdots.$$

   That is, the probability of the union of disjoint events is the summation of the probability of these events.

It is worth dwelling slightly on axiom 3. Consider the case of drawing a card at random from a deck of 52 cards. Using the equally likely outcome model for probability we know that the probability that any card is drawn is given by $\frac{1}{52}$. If I were to ask "what is the probability you draw that ace of spades?" under this model you can respond, immediately, with $\frac{1}{52}$. Now, if

[8] This is actually a scenario where our instincts may lead us awry in some situations. Consider the following from Kahneman and Tversky (1972): Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice, and also participated in anti-nuclear demonstrations. Which is more probable? (a) Linda is a bank teller, or (b) Linda is a bank teller and is active in the feminist movement. A majority of respondents rate (b) as being more probable, even though (a) is contained in (b).

[9] These rules, which we call "properties" are formally known as "axioms".

[10] What would it mean to have a negative probability? It is perhaps a more interesting question than it seems at first glance. It is a topic that has come up in some pretty strange places and, while it is not presently sensible to call them "probabilities" in a traditional sense, there are interesting results which follow.

I were to ask "what is the probability that you draw the ace of spades or the two of spades?" then intuitively you likely figure that this will be $\frac{2}{52}$. Note that the event E_1, "draw the ace of spades" and the event E_2 "draw the two of spades", are disjoint events. Moreover, recall that the union is the "or" and so $E_1 \cup E_2$ is the same as E_1 or E_2. Taken together then,

$$P(E_1 \cup E_2) = P(E_1) + P(E_2).$$

The axiom of additivity extends this intuition to an arbitrary number of events.

Example 3.2 (Basic Additivity)**.** Still unsure of how best to go about using cards to replace their coin game, Charles and Sadie are considering various different events and trying to understand their probabilistic behaviour. They take S, C, H, and D to be the events that a spade, club, heart, or diamond are drawn from a standard deck of cards, respectively. Further, they take C_j to be the event that a card with denomination j is drawn (j ranging from ace with 1 through King with 13). If they consider the union of any two (or more) of these events when can they leverage properties of additivity? When can't they?

Solution

In order to use the properties of additivity it is required that the two events are disjoint. Note that taking any two (or more) of S, C, H, and D will lead to disjoint events. There is no way to draw a card which has two suits on it at once. Similarly, taking any two (or more) of C_j will lead to disjoint events. However, mixing any of the suited events (S, C, H, and D) with any C_j will not be disjoint.
Consider $S \cap C_1$. The ace of spades is in S since it is a spade and it is in C_1 since it is an ace. As a result, $S \cap C_1 = \{\text{Ace of Spades}\}$. Because of this we are not able to say that $P(A \cup C_1) = P(A) + P(C_1)$. However, we can say that

$$P(S \cup C \cup H \cup D) = P(S) + P(C) + P(H) + P(D),$$

and could do the same with any subset of these sets. Similarly, we can take

$$P\left(\bigcup_{j=1}^{13} C_j\right) = \sum_{j=1}^{13} P(C_j),$$

or any of the subsets there.

These three axioms fully define valid probabilities. Any mechanism that assigns probability values to events which conform to these rules will assign valid probabilities. While it may seem counterintuitive that such basic rules fully define our notion of a probability, these rules readily give rise to many other properties that are very useful when working with probabilities.