Chapter 7

Sampling

One of the foundational ideas in statistics is that we can make inferences about an entire population based on a relatively small sample of individuals from that population. In this chapter we will introduce the concept of statistical sampling and discuss why it works.

Anyone living in the United States will be familiar with the concept of sampling from the political polls that have become a central part of our electoral process. In some cases, these polls can be incredibly accurate at predicting the outcomes of elections. The best known example comes from the 2008 and 2012 US Presidential elections, when the pollster Nate Silver correctly predicted electoral outcomes for 49/50 states in 2008 and for all 50 states in 2012. Silver did this by combining data from 21 different polls, which vary in the degree to which they tend to lean towards either the Republican or Democratic side. Each of these polls included data from about 1000 likely voters – meaning that Silver was able to almost perfectly predict the pattern of votes of more than 125 million voters using data from only 21,000 people, along with other knowledge (such as how those states have voted in the past).

7.1 How do we sample?

Our goal in sampling is to determine some feature of the full population of interest, using just a small subset of the population. We do this primarily to save time and effort – why go to the trouble of measuring every individual in the population when just a small sample is sufficient to accurately estimate the variable of interest?

In the election example, the population is all voters, and the sample is the set of 1000 individuals selected by the polling organization. The way in which we select the sample is critical to ensuring that the sample is *representative* of the entire population, which is a main goal of statistical sampling. It's easy to imagine a non-representative sample; if a pollster only called individuals whose names they had received from the local Democratic party, then it would be unlikely that the results of the poll would be representative of the population as a whole. In general, we would define a representative poll as being one in which every member of the population has an equal chance of being selected. When this fails, then we have to worry about whether the statistic that we compute on the sample is *biased* - that is, whether its value is systematically different from the population value (which we refer to as a *parameter*). Keep in mind that we generally don't know this population parameter, because if we did then we wouldn't need to sample! But we will use examples where we have access to the entire population, in order to explain some of the key ideas.

It's important to also distinguish between two different ways of sampling: with replacement versus without replacement. In sampling *with replacement*, after a member of the population has been sampled, they are put back into the pool so that they can potentially be sampled again. In *sampling without replacement*, once a member has been sampled they are not eligible to be sampled again. It's most common to use sampling

without replacement, but there will be some contexts in which we will use sampling with replacement, as when we discuss a technique called *bootstrapping* in Chapter 8.

7.2 Sampling error

Regardless of how representative our sample is, it's likely that the statistic that we compute from the sample is going to differ at least slightly from the population parameter. We refer to this as *sampling error*. The value of our statistical estimate will also vary from sample to sample; we refer to this distribution of our statistic across samples as the *sampling distribution*.

Sampling error is directly related to the quality of our measurement of the population. Clearly we want the estimates obtained from our sample to be as close as possible to the true value of the population parameter. However, even if our statistic is unbiased (that is, in the long run we expect it to have the same value as the population parameter), the value for any particular estimate will differ from the population estimate, and those differences will be greater when the sampling error is greater. Thus, reducing sampling error is an important step towards better measurement.

We will use the NHANES dataset as an example; we are going to assume that NHANES is the entire population, and then we will draw random samples from the population. We will have more to say in the next chapter about exactly how the generation of "random" samples works in a computer.

```
# load the NHANES data library
library(NHANES)

# create a NHANES dataset without duplicated IDs
NHANES <-
  NHANES %>%
  distinct(ID, .keep_all = TRUE)

#create a dataset of only adults
NHANES_adult <-
  NHANES %>%
  filter(
    !is.na(Height),
    Age >= 18
  )

#print the NHANES population mean and standard deviation of adult height
sprintf(
  "Population height: mean = %.2f",
  mean(NHANES_adult$Height)
)
```

```
## [1] "Population height: mean = 168.35"
```

```
sprintf(
  "Population height: std deviation = %.2f",
  sd(NHANES_adult$Height)
)
```

```
## [1] "Population height: std deviation = 10.16"
```

In this example, we know the adult population mean and standard deviation for height because we are assuming that the NHANES dataset contains the entire population of adults. Now let's take a single sample of 50 individuals from the NHANES population, and compare the resulting statistics to the population parameters.

```
# sample 50 individuals from NHANES dataset
exampleSample <-
  NHANES_adult %>%
  sample_n(50)

#print the sample mean and standard deviation of adult height
sprintf(
  'Sample height: mean = %.2f',
  mean(exampleSample$Height)
  )
```

```
## [1] "Sample height: mean = 169.46"
```

```
sprintf(
  'Sample height: std deviation = %.2f',
  sd(exampleSample$Height)
)
```

```
## [1] "Sample height: std deviation = 10.07"
```

The sample mean and standard deviation are similar but not exactly equal to the population values. Now let's take a large number of samples of 50 individuals, compute the mean for each sample, and look at the resulting sampling distribution of means. We have to decide how many samples to take in order to do a good job of estimating the sampling distribution – in this case, let's take 5000 samples so that we are really confident in the answer. Note that simulations like this one can sometimes take a few minutes to run, and might make your computer huff and puff. The histogram in Figure 7.1 shows that the means estimated for each of the samples of 50 individuals vary somewhat, but that overall they are centered around the population mean.

```
# compute sample means across 5000 samples from NHANES data
sampSize <- 50 # size of sample
nsamps <- 5000 # number of samples we will take

# set up variable to store all of the results
sampMeans <- array(NA, nsamps)

# Loop through and repeatedly sample and compute the mean
for (i in 1:nsamps) {
  NHANES_sample <- sample_n(NHANES_adult, sampSize)
  sampMeans[i] <- mean(NHANES_sample$Height)
}

sprintf(
  "Average sample mean = %.2f",
  mean(sampMeans)
)
```

```
## [1] "Average sample mean = 168.33"
```

```
sampMeans_df <- tibble(sampMeans = sampMeans)
```

7.3 Standard error of the mean

Later in the course it will become essential to be able to characterize how variable our samples are, in order to make inferences about the sample statistics. For the mean, we do this using a quantity called the *standard*

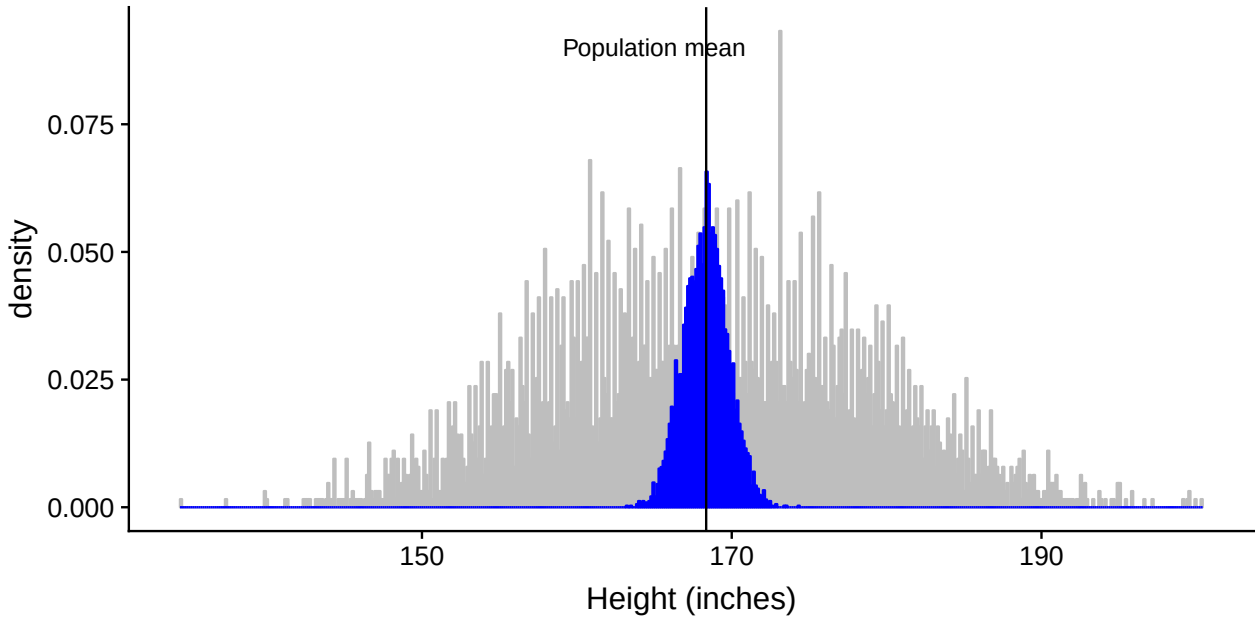

Figure 7.1: The blue histogram shows the sampling distribution of the mean over 5000 random samples from the NHANES dataset. The histogram for the full dataset is shown in gray for reference.

error of the mean (SEM), which one can think of as the standard deviation of the sampling distribution. If we know the population standard deviation, then we can compute the standard error using:

$$SEM = \frac{\sigma}{\sqrt{n}}$$

where n is the size of the sample. We don't usually know σ (the population standard deviation), so instead we would usually plug in our estimate of σ, which is the standard deviation computed on the sample ($\hat{\sigma}$):

$$SEM = \frac{\hat{\sigma}}{\sqrt{n}}$$

However, we have to be careful about computing SEM using the estimated standard deviation if our sample is small (less than about 30).

Because we have many samples from the NHANES population and we actually know the population parameter, we can confirm that the SEM estimated using the population parameter is very close to the observed standard deviation of the samples that we took from the NHANES dataset.

```
# compare standard error based on population to standard deviation
# of sample means

sprintf(
  'Estimated standard error based on population SD: %.2f',
  sd(NHANES_adult$Height)/sqrt(sampSize)
)
```

```
## [1] "Estimated standard error based on population SD: 1.44"
```

```
sprintf(
  'Standard deviation of sample means = %.2f',
```

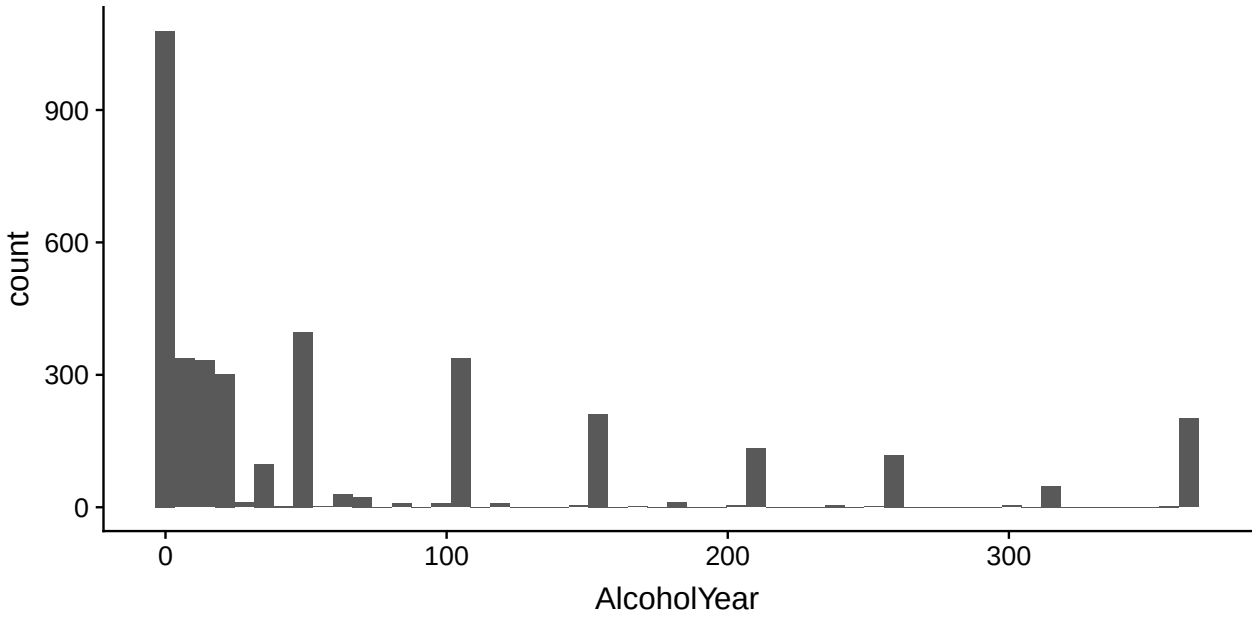

Figure 7.2: Distribution of the variable AlcoholYear in the NHANES dataset, which reflects the number of days that the individual drank in a year.

```
  sd(sampMeans)
)
```

```
## [1] "Standard deviation of sample means = 1.43"
```

The formula for the standard error of the mean says that the quality of our measurement involves two quantities: the population variability, and the size of our sample. Of course, because the sample size is the denominator in the formula for SEM, a larger sample size will yield a smaller SEM when holding the population variability constant. We have no control over the population variability, but we *do* have control over the sample size. Thus, if we wish to improve our sample statistics (by reducing their sampling variability) then we should use larger samples. However, the formula also tells us something very fundamental about statistical sampling – namely, that the utility of larger samples diminishes with the square root of the sample size. This means that doubling the sample size will *not* double the quality of the statistics; rather, it will improve it by a factor of $\sqrt{2}$. In Section 10.3 we will discuss statistical power, which is intimately tied to this idea.

7.4 The Central Limit Theorem

The Central Limit Theorem tells us that as sample sizes get larger, the sampling distribution of the mean will become normally distributed, *even if the data within each sample are not normally distributed.*

We can also see this in real data. Let's work with the variable AlcoholYear in the NHANES distribution, which is highly skewed, as shown in Figure 7.2.

This distribution is, for lack of a better word, funky – and definitely not normally distributed. Now let's look at the sampling distribution of the mean for this variable. Figure 7.3 shows the sampling distribution for this variable, which is obtained by repeatedly drawing samples of size 50 from the NHANES dataset and taking the mean. Despite the clear non-normality of the original data, the sampling distribution is remarkably close to the normal.

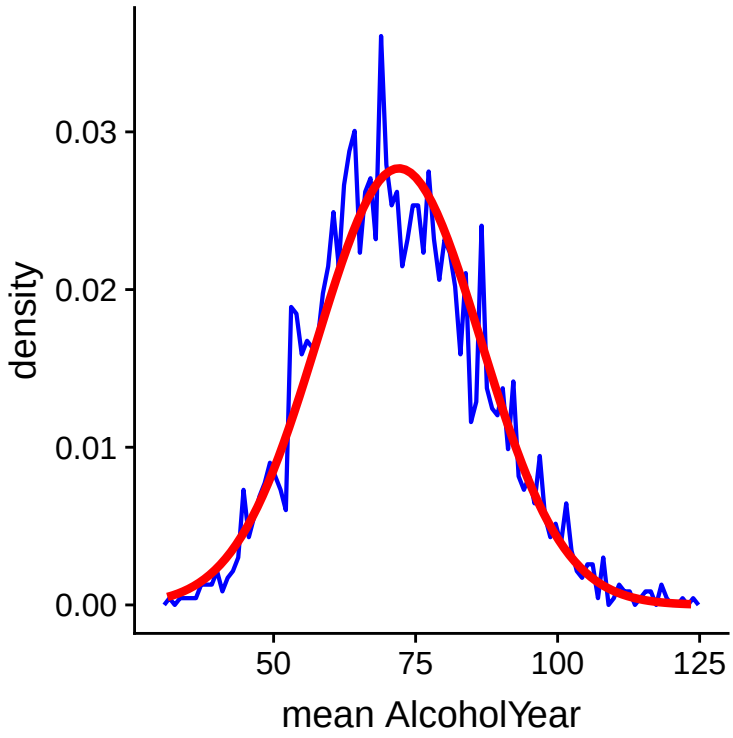

Figure 7.3: The sampling distribution of the mean for AlcoholYear in the NHANES dataset, obtained by drawing repeated samples of size 50, in blue. The normal distribution with the same mean and standard deviation is shown in red.

7.5 Confidence intervals

Most people are familiar with the idea of a "margin of error" for political polls. These polls usually try to provide an answer that is accurate within +/- 3 percent. For example, when a candidate is estimated to win an election by 9 percentage points with a margin of error of 3, the percentage by which they will win is estimated to fall within 6-12 percentage points. In statistics we refer to this range of values as the *confidence interval*, which provides a measure of our degree of uncertainty about how close our estimate is to the population parameter. The larger the condidence interval, the greater our uncertainty.

We saw in the previous section that with sufficient sample size, the sampling distribution of the mean is normally distributed, and that the standard error describes the standard deviation of this sampling distribution. Using this knowledge, we can ask: What is the range of values within which we would expect to capture 95% of all estimates of the mean? To answer this, we can use the normal distribution, for which we know the values between which we expect 95% of all sample means to fall. Specifically, we use the *quantile* function for the normal distribution (`qnorm()` in R) to determine the values of the normal distribution that fall at the 2.5% and 97.5% points in the distribution. We choose these points because we want to find the 95% of values in the center of the distribution, so we need to cut off 2.5% on each end in order to end up with 95% in the middle. Figure 7.4 shows that this occurs for $Z \pm 1.96$.

Using these cutoffs, we can create a confidence interval for the estimate of the mean:

$$CI_{95\%} = \bar{X} \pm 1.96 * SEM$$

Let's compute the confidence interval for the NHANES height data,

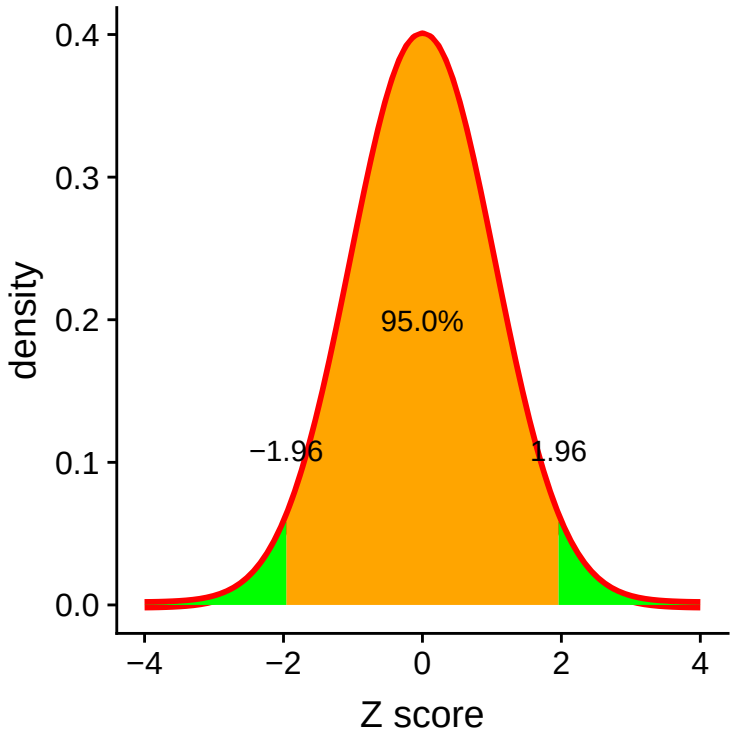

Figure 7.4: Normal distribution, with the orange section in the center denoting the range in which we expect 95 percent of all values to fall. The green sections show the portions of the distribution that are more extreme, which we would expect to occur less than 5 percent of the time.

```
# compute confidence intervals

NHANES_sample <- sample_n(NHANES_adult,250)

sample_summary <- NHANES_sample %>%
    summarize(mean=mean(Height),
              sem=sd(Height)/sqrt(sampSize)) %>%
    mutate(CI_lower=mean-1.96*sem,
           CI_upper=mean+1.96*sem)
pander(sample_summary)
```

mean	sem	CI_lower	CI_upper
166.869	1.446	164.036	169.702

Confidence intervals are notoriously confusing, primarily because they don't mean what we would hope they mean. It seems natural to think that the 95% confidence interval tells us that there is a 95% chance that the population mean falls within the interval. However, as we will see throughout the course, concepts in statistics often don't mean what we think they should mean. In the case of confidence intervals, we can't interpret them in this way because the population parameter has a fixed value – it either is or isn't in the interval. The proper interpretation of the 95% confidence interval is that it is the interval that will capture the true population mean 95% of the time. We can confirm this by resampling the NHANES data repeatedly and counting how often the interval contains the true population mean.

```
# compute how often the confidence interval contains the true population mean
nsamples <- 2500
sampSize <- 100

ci_contains_mean <- array(NA,nsamples)

for (i in 1:nsamples) {
  NHANES_sample <- sample_n(NHANES_adult, sampSize)
```

7.6 Suggested readings

- *The Signal and the Noise: Why So Many Predictions Fail - But Some Don't*, by Nate Silver

Chapter 8

Resampling and simulation

The use of computer simulations has become an essential aspect of modern statistics. For example, one of the most important books in practical computer science, called *Numerical Recipes*, says the following:

> "Offered the choice between mastery of a five-foot shelf of analytical statistics books and middling ability at performing statistical Monte Carlo simulations, we would surely choose to have the latter skill."

In this chapter we will introduce the concept of a Monte Carlo simulation and discuss how it can be used to perform statistical analyses.

8.1 Monte Carlo simulation

The concept of Monte Carlo simulation was devised by the mathematicians Stan Ulam and Nicholas Metropolis, who were working to develop an atomic weapon as part of the Manhattan Project (https://en.wikipedia.org/wiki/Manhattan_Project). They needed to compute the average distance that a neutron would travel in a substance before it collided with an atomic nucleus, but they could not compute this using standard mathematics. Ulam realized that these computations could be simulated using random numbers, just like a casino game. In a casino game, numbers are drawn at random; to estimate the probability of a specific outcome, you could play the game hundreds of times. Ulam's uncle had gambled at the Monte Carlo casino in Monaco, which is apparently where the name came from for this new technique.

There are four steps to performing a Monte Carlo simulation:

1. Define a domain of possible values
2. Generate random numbers within that domain from a probability distribution
3. Perform a computation using the random numbers
4. Combine the results across many repetitions

As an example, let's say that I want to figure out how much time to allow for an in-class quiz. Say that we know that the distribution of quiz completion times is normal, with mean of 5 mins and standard deviation of 1 min. Given this, how long does the test period need to be so that we expect everyone to finish 99% of the time? There are two ways to solve this problem. The first is to calculate the answer using a mathematical theory known as the statistics of extreme values. However, this is quite complicated mathematically. Alternatively, we could use Monte Carlo simulation. To do this, we need to generate random samples from a normal distribution.

8.2 Randomness in statistics

The term "random" is often used colloquially to refer to things that are bizarre or unexpected, but in statistics the term has a very specific meaning: A process is *random* if it is unpredictable. For example, if I flip a fair coin 10 times, the value of the outcome on one flip does not provide me with any information that lets me predict the outcome on the next flip. It's important to note that the fact that something is unpredictable doesn't necessarily mean that it is not deterministic. For example, when we flip a coin, the outcome of the flip is determined by the laws of physics; if we knew all of the conditions in enough detail, we should be able to predict the outcome of the flip. However, many factors combine to make the outcome of the coin flip unpredictable in practice.

Psychologists have shown that humans actually have a fairly bad sense of randomness. First, we tend to see patterns when they don't exist. In the extreme, this leads to the phenomenon of *pareidolia*, in which people will perceive familiar objects within random patterns (such as perceiving a cloud as a human face or seeing the Virgin Mary in a piece of toast). Second, humans tend to think of random processes as self-correcting, which leads us to expect that we are "due for a win" after losing many rounds in a game, a phenonenon known as the "gambler's fallacy".

8.3 Generating random numbers

Running a Monte Carlo simulation requires that we generate random numbers. Generating truly random numbers (i.e. numbers that are completely unpredictable) is only possible through physical processes, such as the decay of atoms or the rolling of dice, which are difficult to obtain and/or too slow to be useful for computer simulation (though they can be obtained from the NIST Randomness Beacon).

In general, instead of truly random numbers we use *pseudo-random* numbers generated using a computer algorithm; these numbers will seem random in the sense that they are difficult to predict, but the series of numbers will actually repeat at some point. For example, the random number generator used in R will repeate after $2^{19937} - 1$ numbers. That's far more than the number of seconds in the history of the universe, and we generally think that this is fine for most purposes in statistical analysis.

In R, there is a function to generate random numbers for each of the major probability distributions, such as:

- `runif()` - uniform distribution (all values between 0 and 1 equally)
- `rnorm()` - normal distribution
- `rbinom()` - binomial distribution (e.g. rolling the dice, coin flips)

Figure 8.1 shows examples of numbers generated using the `runif()` and `rnorm()` functions, generated using the following code:

```
p1 <-
  tibble(
    x = runif(10000)
  ) %>%
  ggplot((aes(x))) +
  geom_histogram(bins = 100) +
  labs(title = "Uniform")

p2 <-
  tibble(
    x = rnorm(10000)
  ) %>%
  ggplot(aes(x)) +
  geom_histogram(bins = 100) +
  labs(title = "Normal")
```

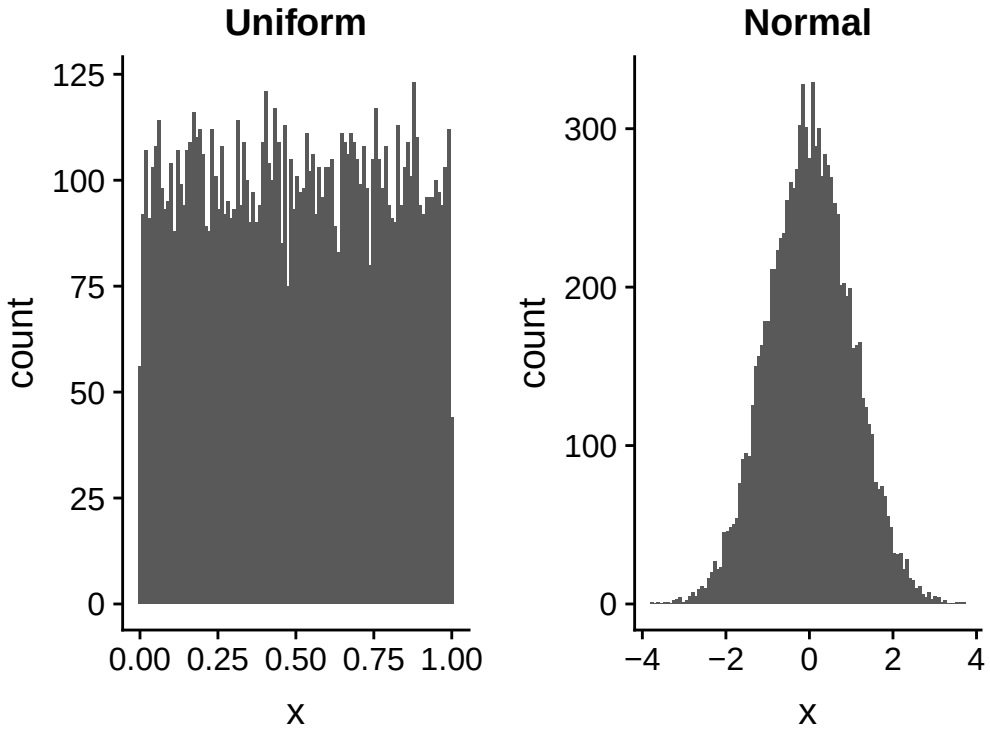

Figure 8.1: Examples of random numbers generated from a uniform (left) or normal (right) distribution.

```
plot_grid(p1, p2, ncol = 3)
```

You can also generate random numbers for any distribution if you have a *quantile* function for the distribution. This is the inverse of the cumulative distribution function; instead of identifying the cumulative probabilities for a set of values, the quantile function identifies the values for a set of cumulative probabilities. Using the quantile function, we can generate random numbers from a uniform distribution, and then map those into the distribution of interest via its quantile function.

By default, R will generate a different set of random numbers every time you run one of the random number generator functions described above. However, it is also possible to generate exactly the same set of random numbers, by setting what is called the *random seed* to a specific value. We will do this in many of the examples in this book, in order to make sure that the examples are reproducible.

```
# if we run the rnorm() command twice, it will give us different sets of pseudorandom numbers each time
print(rnorm(n = 5))
```

```
## [1]  1.48  0.18  0.21 -0.15 -1.72
```

```
print(rnorm(n = 5))
```

```
## [1] -0.691 -2.231  0.391  0.029 -0.647
```

```
# if we set the random seed to the same value each time, then it will give us the same series of pseudo
set.seed(12345)
print(rnorm(n = 5))
```

```
## [1]  0.59  0.71 -0.11 -0.45  0.61
```

```
set.seed(12345)
print(rnorm(n = 5))
```

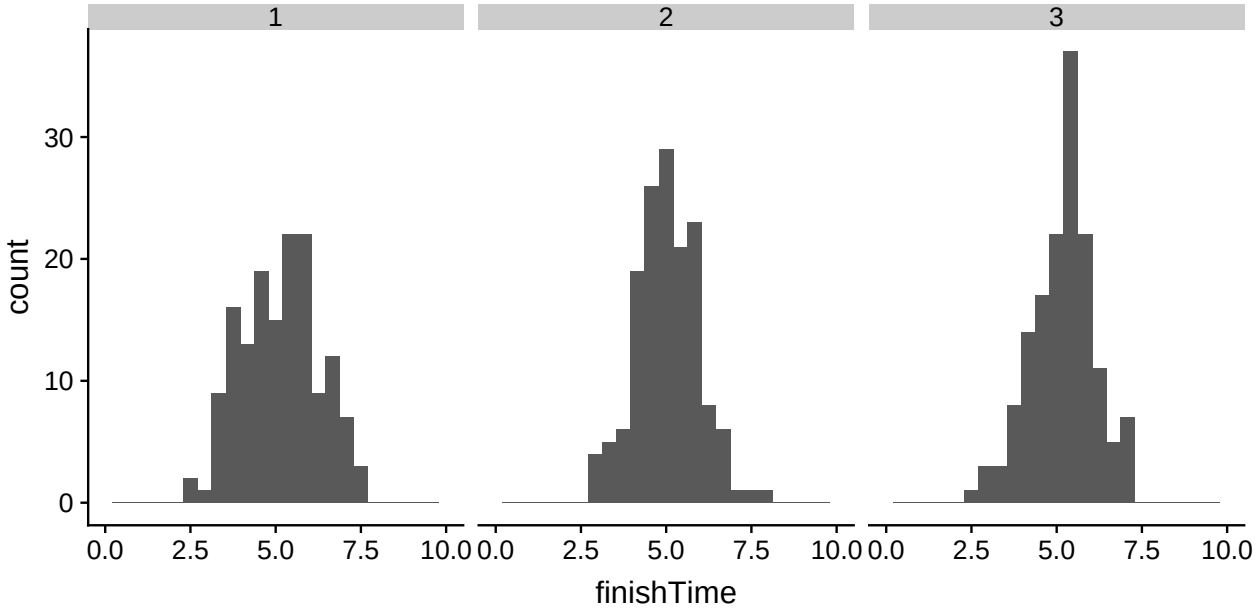

Figure 8.2: Simulated finishing time distributions.

```
## [1]  0.59  0.71 -0.11 -0.45  0.61
```

8.4 Using Monte Carlo simulation

Let's go back to our example of exam finishing times. Let's say that I administer three quizzes and record the finishing times for each student for each exam, which might look like the distributions presented in Figure 8.2.

However, what we really want to know is not what the distribution of finishing times looks like, but rather what the distribution of the *longest* finishing time for each quiz looks like. To do this, we can simulate a large number of quizzes (using the assumption that the finishing times are distributed normally, as stated above); for each of these simulated quizzes, we can record the longest finishing time. To do this, we create a new function in R called `sampleMax()` which simulates a sample of the appropriate size (i.e. the number of students in the class) from the appropriate distribution (i.e., normal), and returns the maximum value in the sample. We then repeat this simulation a large number of times (5000 should be enough) using the `replicate()` function, which will store all of the outputs into a single variable. The distribution of finishing times is shown in Figure 8.3.

```
# sample maximum value 5000 times and compute 99th percentile
nRuns <- 5000
sampSize <- 150

sampleMax <- function(sampSize = 150) {
  samp <- rnorm(sampSize, mean = 5, sd = 1)
  return(max(samp))
}

maxTime <- replicate(nRuns, sampleMax())

cutoff <- quantile(maxTime, 0.99)
sprintf("99th percentile of maxTime distribution: %.2f", cutoff)
```

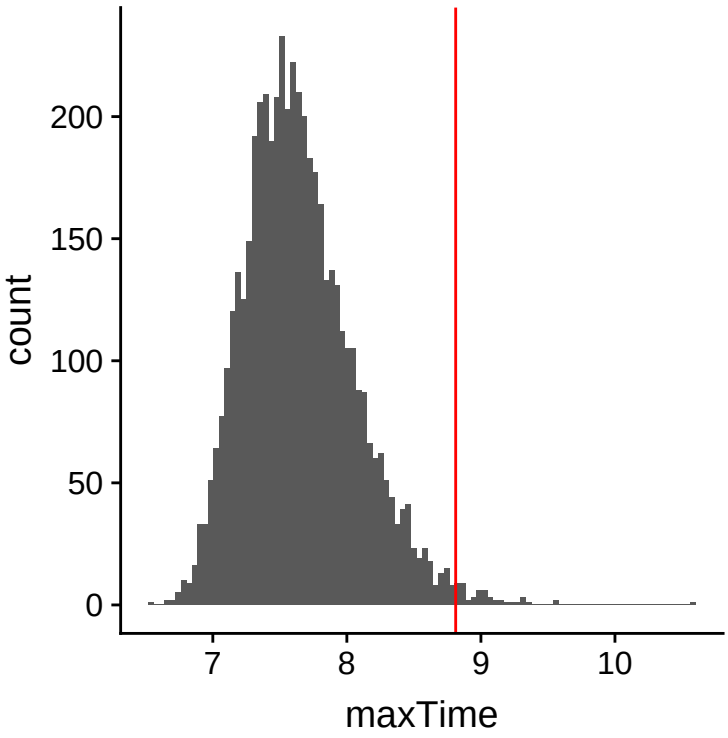

Figure 8.3: Distribution of maximum finishing times across simulations.

```
## [1] "99th percentile of maxTime distribution: 8.81"
```

This shows that the 99th percentile of the finishing time distribution falls at 8.81, meaning that if we were to give that much time for the quiz, then everyone should finish 99% of the time. It's always important to remember that our assumptions matter – if they are wrong, then the results of the simulation are useless. In this case, we assumed that the finishing time distribution was normally distributed with a particular mean and standard deviation; if these assumptions are incorrect (and they almost certainly are), then the true answer could be very different.

8.5 Using simulation for statistics: The bootstrap

So far we have used simulation to demonstrate statistical principles, but we can also use simulation to answer real statistical questions. In this section we will introduce a concept known as the *bootstrap* that lets us use simulation to quantify our uncertainty about statistical estimates. Later in the course, we will see other examples of how simulation can often be used to answer statistical questions, especially when theoretical statistical methods are not available or when their assumptions are too stifling.

8.5.1 Computing the bootstrap

In the section above, we used our knowledge of the sampling distribution of the mean to compute the standard error of the mean and confidence intervals. But what if we can't assume that the estimates are normally distributed, or we don't know their distribution? The idea of the bootstrap is to use the data themselves to estimate an answer. The name comes from the idea of pulling one's self up by one's own bootstraps, expressing the idea that we don't have any external source of leverage so we have to rely upon the data themselves. The bootstrap method was conceived by Bradley Efron of the Stanford Department of Statistics, who is one of the world's most influential statisticians.

The idea behind the bootstrap is that we repeatedly sample from the actual dataset; importantly, we sample *with replacement*, such that the same data point will often end up being represented multiple times within one of the samples. We then compute our statistic of interest on each of the bootstrap samples, and use the distribution of those estimates.

Let's start by using the bootstrap to estimate the sampling distribution of the mean, so that we can compare the result to the standard error of the mean (SEM) that we discussed earlier.

```
# perform the bootstrap to compute SEM and compare to parametric method

nRuns <- 2500
sampleSize <- 32

heightSample <-
  NHANES_adult %>%
  sample_n(sampleSize)

bootMeanHeight <- function(df) {
  bootSample <- sample_n(df, dim(df)[1], replace = TRUE)
  return(mean(bootSample$Height))
}

bootMeans <- replicate(nRuns, bootMeanHeight(heightSample))

SEM_standard <- sd(heightSample$Height) / sqrt(sampleSize)
sprintf("SEM computed using sample SD: %f", SEM_standard)
```

```
## [1] "SEM computed using sample SD: 1.595789"
```

```
SEM_bootstrap <- sd(bootMeans)
sprintf("SEM computed using SD of bootstrap estimates: %f", SEM_bootstrap)
```

```
## [1] "SEM computed using SD of bootstrap estimates: 1.586913"
```

Figure 8.4 shows that the distribution of means across bootstrap samples is fairly close to the theoretical estimate based on the assumption of normality. We can also use the bootstrap samples to compute a confidence interval for the mean, simply by computing the quantiles of interest from the distribution of bootstrap samples.

```
# compute bootstrap confidence interval

bootCI <- quantile(bootMeans, c(0.025, 0.975))
pander("bootstrap confidence limits:")
```

bootstrap confidence limits:

```
pander(bootCI)
```

2.5%	98%
164.634	170.883

```
# now let's compute the confidence intervals using the sample mean and SD
sampleMean <- mean(heightSample$Height)

normalCI <-
  tibble(
```

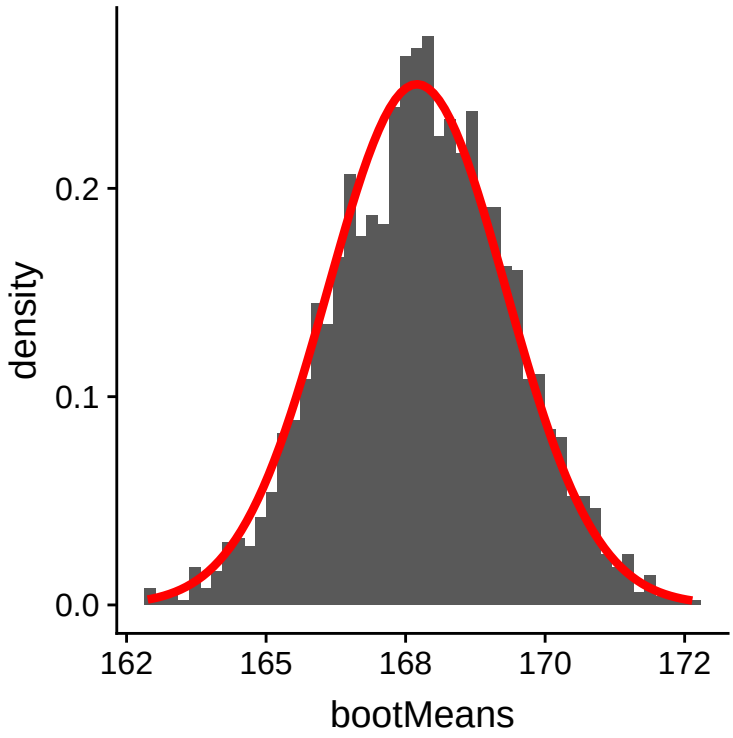

Figure 8.4: An example of bootstrapping to compute the standard error of the mean. The histogram shows the distribution of means across bootstrap samples, while the red line shows the normal distribution based on the sample mean and standard deviation.

```
  "2.5%" = sampleMean - 1.96 * SEM_standard,
  "97.5%" = sampleMean + 1.96 * SEM_standard
)

print("confidence limits based on sample SD and normal distribution:")
```

```
## [1] "confidence limits based on sample SD and normal distribution:"
```

```
pander(normalCI)
```

2.5%	97.5%
164.575	170.831

We would not usually employ the bootstrap to compute confidence intervals for the mean (since we can generally assume that the normal distribution is appropriate for the sampling distribution of the mean, as long as our sample is large enough), but this example shows how the method gives us roughly the same result as the standard method based on the normal distribution. The bootstrap would more often be used to generate standard errors for estimates of other statistics where we know or suspect that the normal distribution is not appropriate.

8.6 Suggested readings

- *Computer Age Statistical Inference: Algorithms, Evidence and Data Science*, by Bradley Efron and Trevor Hastie

Chapter 9

Hypothesis testing

In the first chapter we discussed the three major goals of statistics:

- Describe
- Decide
- Predict

In this chapter we will introduce the ideas behind the use of statistics to make decisions – in particular, decisions about whether a particular hypothesis is supported by the data.

9.1 Null Hypothesis Statistical Testing (NHST)

The specific type of hypothesis testing that we will discuss is known (for reasons that will become clear) as *null hypothesis statistical testing* (NHST). If you pick up almost any scientific or biomedical research publication, you will see NHST being used to test hypotheses, and in their introductory psychology textbook, Gerrig & Zimbardo (2002) referred to NHST as the "backbone of psychological research". Thus, learning how to use and interpret the results from hypothesis testing is essential to understand the results from this research.

It is also important for you to know, however, that NHST is deeply flawed, and that many statisticians and researchers (including myself) think that it has been the cause of serious problems in science, which we will discuss in Chapter 17. For more than 50 years, there have been calls to abandon NHST in favor of other approaches (like those that we will discuss in the following chapters):

- "The test of statistical significance in psychological research may be taken as an instance of a kind of essential mindlessness in the conduct of research" (Bakan, 1966)
- Hypothesis testing is "a wrongheaded view about what constitutes scientific progress" (Luce, 1988)

NHST is also widely misunderstood, largely because it violates our intuitions about how statistical hypothesis testing should work. Let's look at an example to see.

9.2 Null hypothesis statistical testing: An example

There is great interest in the use of body-worn cameras by police officers, which are thought to reduce the use of force and improve officer behavior. However, in order to establish this we need experimental evidence, and it has become increasingly common for governments to use randomized controlled trials to test such ideas. A randomized controlled trial of the effectiveness of body-worn cameras was performed by the Washington, DC government and DC Metropolitan Police Department in 2015/2016 in order to test the hypothesis that body-worn cameras are effective. Officers were randomly assigned to wear a body-worn camera or not, and

their behavior was then tracked over time to determine whether the cameras resulted in less use of force and fewer civilian complaints about officer behavior.

Before we get to the results, let's ask how you would think the statistical analysis might work. Let's say we want to specifically test the hypothesis of whether the use of force is decreased by the wearing of cameras. The randomized controlled trial provides us with the data to test the hypothesis – namely, the rates of use of force by officers assigned to either the camera or control groups. The next obvious step is to look at the data and determine whether they provide convincing evidence for or against this hypothesis. That is: What is the likelihood that body-worn cameras reduce the use of force, given the data and everything else we know?

It turns out that this is *not* how null hypothesis testing works. Instead, we first take our hypothesis of interest (i.e. whether body-worn cameras reduce use of force), and flip it on its head, creating a *null hypothesis* – in this case, the null hypothesis would be that cameras do not reduce use of force. Importantly, we then assume that the null hypothesis is true. We then look at the data, and determine whether the data are sufficiently unlikely under the null hypothesis that we can reject the null in favor of the *alternative hypothesis* which is our hypothesis of interest. If there is not sufficient evidence to reject the null, then we say that we "failed to reject" the null.

Understanding some of the concepts of NHST, particularly the notorious "p-value", is invariably challenging the first time one encounters them, because they are so counter-intuitive. As we will see later, there are other approaches that provide a much more intuitive way to address hypothesis testing (but have their own complexities). However, before we get to those, it's important for you to have a deep understanding of how hypothesis testing works, because it's clearly not going to go away any time soon.

9.3 The process of null hypothesis testing

We can break the process of null hypothesis testing down into a number of steps:

1. Formulate a hypothesis that embodies our prediction (*before seeing the data*)
2. Collect some data relevant to the hypothesis
3. Specify null and alternative hypotheses
4. Fit a model to the data that represents the alternative hypothesis and compute a test statistic
5. Compute the probability of the observed value of that statistic assuming that the null hypothesis is true
6. Assess the "statistical significance" of the result

For a hands-on example, let's use the NHANES data to ask the following question: Is physical activity related to body mass index? In the NHANES dataset, participants were asked whether they engage regularly in moderate or vigorous-intensity sports, fitness or recreational activities (stored in the variable *PhysActive*). They also measured height and weight and computed Body Mass Index:

$$BMI = \frac{weight(kg)}{height(m)^2}$$

9.3.1 Step 1: Formulate a hypothesis

For step 1, we hypothesize that BMI should be greater for people who do not engage in physical activity, compared to those who do.

9.3.2 Step 2: Collect some data

For step 2, we collect some data. In this case, we will sample 250 individuals from the NHANES dataset. Figure 9.1 shows an example of such a sample, with BMI shown separately for active and inactive individuals.

```
# sample 250 adults from NHANES and compute mean BMI separately for active
# and inactive individuals

sampSize <- 250

NHANES_sample <-
  NHANES_adult %>%
  sample_n(sampSize)

sampleSummary <-
  NHANES_sample %>%
  group_by(PhysActive) %>%
  summarize(
    N = length(BMI),
    mean = mean(BMI),
    sd = sd(BMI)
  )

# calculate the mean difference in BMI between active
# and inactive individuals; we'll use this later to calculate the t-statistic
meanDiff <-
  sampleSummary %>%
  select(
    PhysActive,
    mean
  ) %>%
  spread(PhysActive, mean) %>%
  mutate(
    meanDiff = No - Yes
  ) %>%
  pull(meanDiff)

# calculate the summed variances in BMI for active
# and inactive individuals; we'll use this later to calculate the t-statistic
sumVariance <-
  sampleSummary %>%
  select(
    PhysActive,
    N,
    sd
  ) %>%
  gather(column, stat, N:sd) %>%
  unite(temp, PhysActive, column) %>%
  spread(temp, stat) %>%
  mutate(
    sumVariance = No_sd**2 / No_N + Yes_sd**2 / Yes_N
  ) %>%
  pull(sumVariance)


# print sampleSummary table
pander(sampleSummary)
```

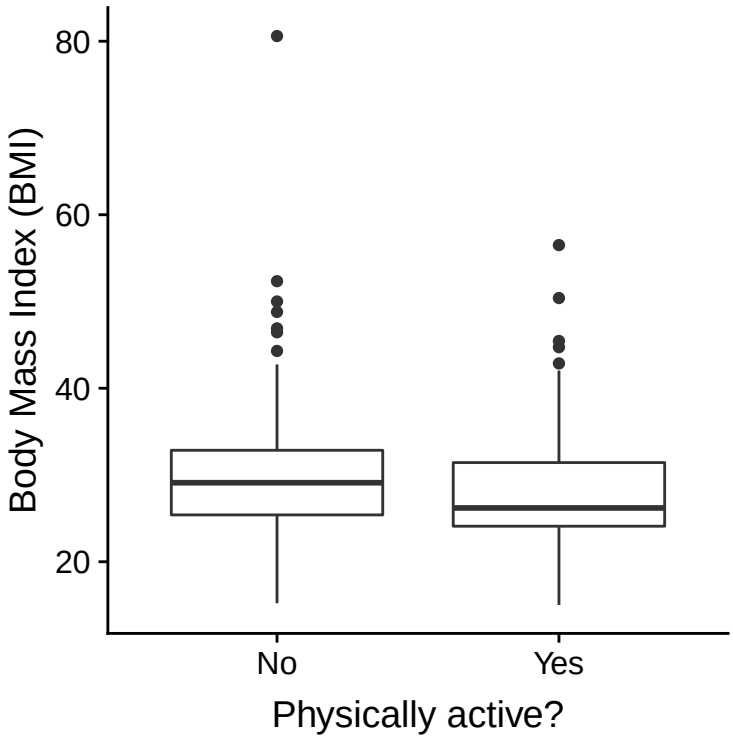

Figure 9.1: Box plot of BMI data from a sample of adults from the NHANES dataset, split by whether they reported engaging in regular physical activity.

PhysActive	N	mean	sd
No	135	30.25	8.2
Yes	115	28.6	6.88

9.3.3 Step 3: Specify the null and alternative hypotheses

For step 3, we need to specify our null hypothesis (which we call H_0) and our alternative hypothesis (which we call H_A). H_0 is the baseline against which we test our hypothesis of interest: that is, what would we expect the data to look like if there was no effect? The null hypothesis always involves some kind of equality ($=$, $\leq$, or $\geq$). H_A describes what we expect if there actually is an effect. The alternative hypothesis always involves some kind of inequality ($\neq$, $>$, or $<$). Importantly, null hypothesis testing operates under the assumption that the null hypothesis is true unless the evidence shows otherwise.

We also have to decide whether to use *directional* or *non-directional* hypotheses. A non-directional hypothesis simply predicts that there will be a difference, without predicting which direction it will go. For the BMI/activity example, a non-directional null hypothesis would be:

$H0 : BMI_{active} = BMI_{inactive}$

and the corresponding non-directional alternative hypothesis would be:

$HA : BMI_{active} \neq BMI_{inactive}$

A directional hypothesis, on the other hand, predicts which direction the difference would go. For example, we have strong prior knowledge to predict that people who engage in physical activity should weigh less than those who do not, so we would propose the following directional null hypothesis:

$H0 : BMI_{active} \geq BMI_{inactive}$

and directional alternative:

$HA : BMI_{active} < BMI_{inactive}$

9.3.4 Step 4: Fit a model to the data and compute a test statistic

For step 4, we want to use the data to compute a statistic that will ultimately let us decide whether the null hypothesis is rejected or not. To do this, the model needs to quantify the amount of evidence in favor of the alternative hypothesis, relative to the variability in the data. Thus we can think of the test statistic as providing a measure of the size of the effect compared to the variability in the data. In general, this test statistic will have a probability distribution associated with it, because that allows us to determine how likely our observed value of the statistic is under the null hypothesis.

For the BMI example, we need a test statistic that allows us to test for a difference between two means, since the hypotheses are stated in terms of mean BMI for each group. One statistic that is often used to compare two means is the *t-statistic*, first developed by the statistician William Sealy Gossett, who worked for the Guiness Brewery in Dublin and wrote under the pen name "Student" - hence, it is often called "Student's t-statistic". The t-statistic is appropriate for comparing the means of two groups when the sample sizes are relatively small and the population standard deviation is unknown. The t-statistic for comparison of two independent groups is computed as:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

where $\bar{X}_1$ and $\bar{X}_2$ are the means of the two groups, S_1^2 and S_2^2 are the estimated variances of the groups, and n_1 and n_2 are the sizes of the two groups. The t-statistic is distributed according to a probability distribution known as a t distribution. The t distribution looks quite similar to a normal distribution, but it differs depending on the number of degrees of freedom, which for this example is the number of observations minus 2, since we have computed two means and thus given up two degrees of freedom. When the degrees of freedom are large (say 1000), then the t distribution looks essentialy like the normal distribution, but when they are small then the t distribution has longer tails than the normal (see Figure 9.2).

9.3.5 Step 5: Determine the probability of the data under the null hypothesis

This is the step where NHST starts to violate our intuition – rather than determining the likelihood that the null hypothesis is true given the data, we instead determine the likelihood of the data under the null hypothesis - because we started out by assuming that the null hypothesis is true! To do this, we need to know the probability distribution for the statistic under the null hypothesis, so that we can ask how likely the data are under that distribution. Before we move to our BMI data, let's start with some simpler examples.

9.3.5.0.1 Randomization: A very simple example

Let's say that we wish to determine whether a coin is fair. To collect data, we flip the coin 100 times, and we count 70 heads. In this example, $H_0 : P(heads) = 0.5$ and $H_A : P(heads) \neq 0.5$, and our test statistic is simply the number of heads that we counted. The question that we then want to ask is: How likely is it that we would observe 70 heads if the true probability of heads is 0.5. We can imagine that this might happen very occasionally just by chance, but doesn't seem very likely. To quantify this probability, we can use the *binomial distribution*:

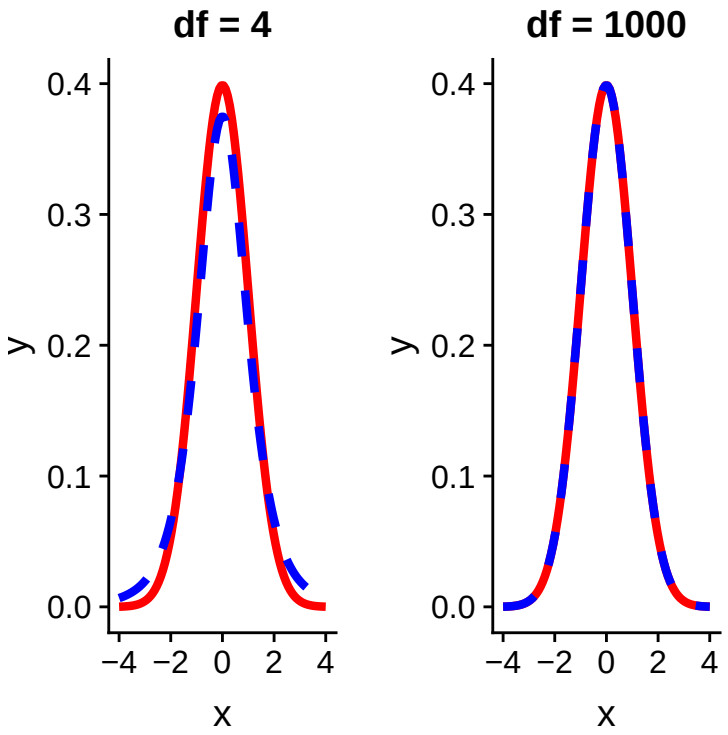

Figure 9.2: Each panel shows the t distribution (in blue dashed line) overlaid on the normal distribution (in solid red line). The left panel shows a t distribution with 4 degrees of freedom, in which case the distribution is similar but has slightly wider tails. The right panel shows a t distribution with 1000 degrees of freedom, in which case it is virtually identical to the normal.

$$P(X < k) = \sum_{i=0}^{k} \binom{N}{k} p^i (1-p)^{(n-i)}$$

This equation will tell us the likelihood of a certain number of heads or fewer, given a particular probability of heads. However, what we really want to know is the probability of a certain number or more, which we can obtain by subtracting from one:

$$P(X \geq k) = 1 - P(X < k)$$

We can compute the probability for our example using the `pbinom()` function in R as follows:

```
# compute the probability of 69 or fewer heads, when P(heads)=0.5
p_lt_70 <- pbinom(69, 100, 0.5)
sprintf("probability of 69 or fewer heads given P(heads)=0.5: %0.6f", p_lt_70)
```

```
## [1] "probability of 69 or fewer heads given P(heads)=0.5: 0.999961"
```

```
# the probability of 70 or more heads is simply the complement of p_lt_70
p_ge_70 <- 1 - p_lt_70
sprintf("probability of 70 or more heads given P(heads)=0.5: %0.6f", p_ge_70)
```

```
## [1] "probability of 70 or more heads given P(heads)=0.5: 0.000039"
```

This computation shows us that the likelihood of getting 70 heads if the coin is indeed fair is very small. Now, what if we didn't have the `pbinom()` function to tell us the probability of that number of heads? We could instead determine it by simulation – we repeatedly flip a coin 100 times using a true probability of 0.5, and then compute the distribution of the number of heads across those simulation runs. Figure 9.3 shows the result from this simulation.

```
# simulate tossing of 100,000 flips of 100 coins to identify empirical
# probability of 70 or more heads out of 100 flips

# create function to toss coins
tossCoins <- function() {
  flips <- runif(100) > 0.5
  return(sum(flips))
}

# use a large number of replications since this is fast
coinFlips <- replicate(100000, tossCoins())

p_ge_70_sim <- mean(coinFlips >= 70)
sprintf(
  "empirical probability of 70 or more heads given P(heads)=0.5: %0.6f",
  p_ge_70_sim
)
```

```
## [1] "empirical probability of 70 or more heads given P(heads)=0.5: 0.000020"
```

Here we can see that the probability computed via simulation (0.000020) is very close to the theoretical probability (.00004).

Let's do the analogous computation for our BMI example. First we compute the t statistic using the values from our sample that we calculated above:

```
tStat <-
  meanDiff / sqrt(sumVariance)
```

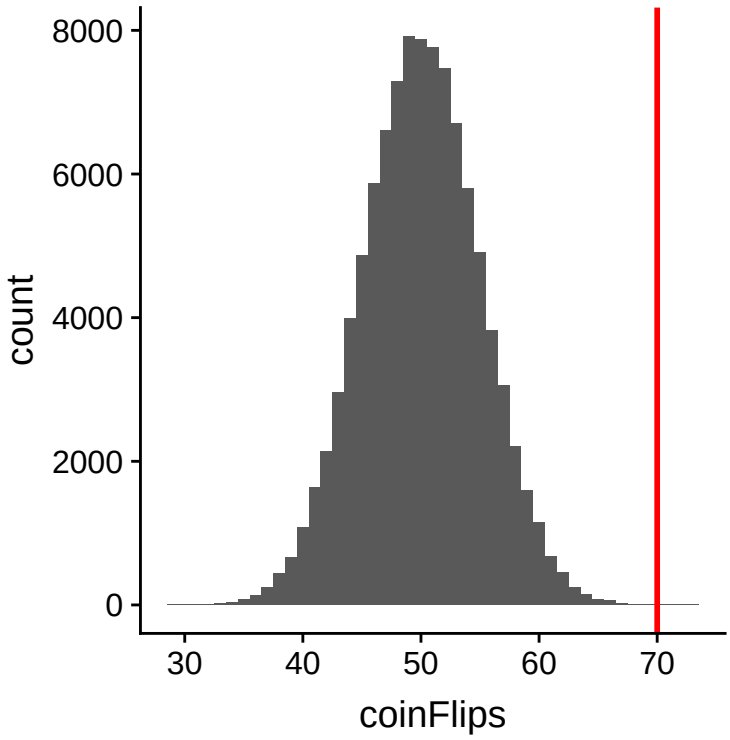

Figure 9.3: Distribution of numbers of heads (out of 100 flips) across 100,000 simulated runs.

```
sprintf("t statistic = %0.3f", tStat)
```

```
## [1] "t statistic = 1.735"
```

The question that we then want to ask is: What is the likelihood that we would find a t statistic of this size, if the true difference between groups is zero or less (i.e. the directional null hypothesis)?
We can use the t distribution to determine this probability. Our sample size is 250, so the appropriate t distribution has 248 degrees of freedom. We can use the `pt()` function in R to determine the probability of finding a value of the t-statistic greater than or equal to our observed value. Note that we want to know the probability of a value greater than our observed value, but by default `pt()` gives us the probability of a value less than the one that we provide it, so we have to tell it explicitly to provide us with the "upper tail" probability (by setting `lower.tail = FALSE`).

```
pvalue_tdist <-
  pt(tStat, df = 248, lower.tail = FALSE)

sprintf("p(t > %0.2f, df = 248) = %0.3f", tStat, pvalue_tdist)
```

```
## [1] "p(t > 1.74, df = 248) = 0.042"
```

This tells us that our observed t-statistic value of 1.74 is relatively unlikely if the null hypothesis really is true.

In this case, we used a directional hypothesis, so we only had to look at one end of the null distribution. If we wanted to test a non-directional hypothesis, then we would need to be able to identify how unexpected the size of the effect is, regardless of its direction. In the context of the t-test, this means that we need to know how likely it is that the statistic would be as extreme in either the positive or negative direction. To do this, we multiply the observed t value by -1, since the t distribution is centered around zero, and then add together the two tail probabilities to get a *two-tailed* p-value:

```
pvalue_tdist_twotailed <-
  pt(tStat, df = 248, lower.tail = FALSE) +
  pt(-1 * tStat, df = 248, lower.tail = TRUE)

sprintf(
  "p(t > %0.2f or t< %0.2f, df = 248) = %0.3f",
  tStat,
  -1 * tStat, pvalue_tdist_twotailed
)
```

```
## [1] "p(t > 1.74 or t< -1.74, df = 248) = 0.084"
```

Here we see that the p value for the two-tailed test is twice as large as that for the one-tailed test, which reflects the fact that an extreme value is less surprising since it could have occurred in either direction.

How do you choose whether to use a one-tailed versus a two-tailed test? The two-tailed test is always going to be more conservative, so it's always a good bet to use that one, unless you had a very strong prior reason for using a one-tailed test. In that case, you should have written down the hypothesis before you ever looked at the data. In Chapter 17 we will discuss the idea of pre-registration of hypotheses, which formalizes the idea of writing down your hypotheses before you ever see the actual data. You should *never* make a decision about how to perform a hypothesis test once you have looked at the data, as this can introduce serious bias into the results.

9.3.5.1 Computing p-values using randomization

So far we have seen how we can use the t-distribution to compute the probability of the data under the null hypothesis, but we can also do this using simulation. The basic idea is that we generate simulated data like those that we would expect under the null hypothesis, and then ask how extreme the observed data are in comparison to those simulated data. The key question is: How can we generate data for which the null hypothesis is true? The general answer is that we can randomly rearrange the data in a specific way that makes the data look like they would if the null was really true. This is similar to the idea of bootstrapping, in the sense that it uses our own data to come up with an answer, but it does it in a different way.

9.3.5.1.1 Randomization: a simple example

Let's start with a simple example. Let's say that we want to compare the mean squatting ability of football players with cross-country runners, with $H_0 : \mu_{FB} \leq \mu_{XC}$ and $H_A : \mu_{FB} > \mu_{XC}$. We measure the maximum squatting ability of 5 football players and 5 cross-country runners (which we will generate randomly, assuming that $\mu_{FB} = 300$, $\mu_{XC} = 140$, and $\sigma = 30$.

```
# generate simulated data for squatting ability across football players
# and cross country runners

# reset random seed for this example
set.seed(12345678)

# create a function to round values to nearest product of 5,
# to keep example simple
roundToNearest5 <- function(x, base = 5) {
  return(base * round(x / base))
}

# create and show data frame containing simulated data
squatDf <- tibble(
```

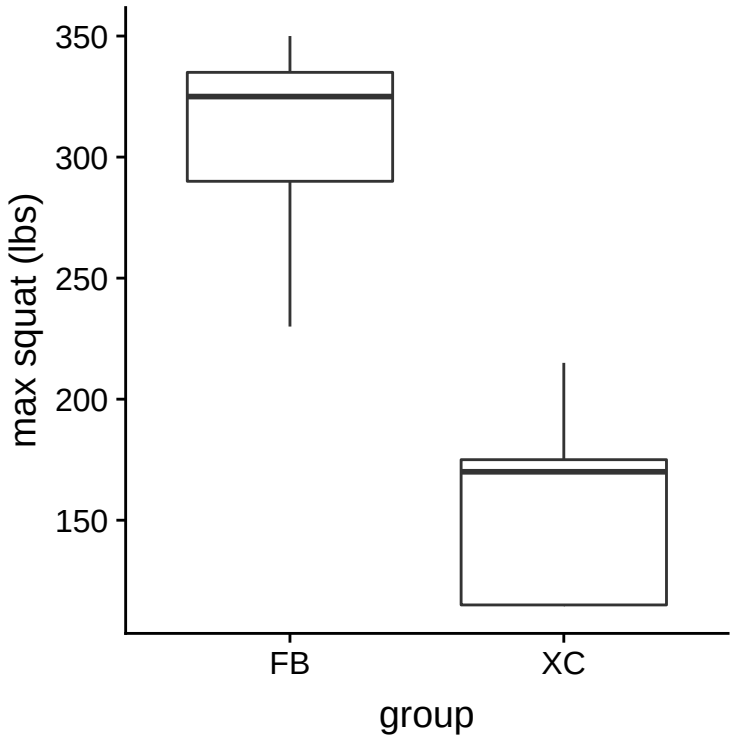

Figure 9.4: Box plots of simulated squatting ability for football players and cross-country runners.

```
  group = as.factor(c(rep("FB", 5), rep("XC", 5))),
  squat = roundToNearest5(c(rnorm(5) * 30 + 300, rnorm(5) * 30 + 140))
)

pander(squatDf)
```

group	squat
FB	335
FB	350
FB	230
FB	290
FB	325
XC	115
XC	115
XC	170
XC	175
XC	215

From the plot in Figure 9.4 it's clear that there is a large difference between the two groups. We can do a standard t-test to test our hypothesis, using the `t.test()` command in R:

```
# compute and print t statistic comparing two groups

tt <-
  t.test(
    squat ~ group,
```

```
    data = squatDf,
    alternative = "greater",
    var.equal = TRUE
  )

sprintf("p(t > %0.2f, df = 8) = %0.5f", tt$statistic, tt$p.value)
```

```
## [1] "p(t > 5.14, df = 8) = 0.00044"
```

This shows that the likelihood of such a difference under the null hypothesis is very small, using the t distribution to define the null. Now let's see how we could answer the same question using randomization. The basic idea is that if the null hypothesis of no difference between groups is true, then it shouldn't matter which group one comes from (football players versus cross-country runners) – thus, to create data that are like our actual data but also conform to the null hypothesis, we can randomly reorder the group labels for the individuals in the dataset, and then recompute the difference between the groups. The results of such a shuffle are shown in Figure 9.5.

```
# create a scrambled version of the group membership variable

dfScram <-
  squatDf %>%
  mutate(
    scrambledGroup = sample(group)
  ) %>%
  select(-group)

pander(dfScram)
```

squat	scrambledGroup
335	FB
350	XC
230	FB
290	XC
325	XC
115	FB
115	FB
170	XC
175	FB
215	XC

After scrambling the labels, we see that the two groups are now much more similar, and in fact the cross-country group now has a slightly higher mean. Now let's do that 10000 times and store the t statistic for each iteration; this may take a moment to complete.

```
# shuffle data 10,000 times and compute distribution of t values

nRuns <- 10000

shuffleAndMeasure <- function(df) {
  dfScram <-
    df %>%
    mutate(
      scrambledGroup = sample(group)
    )
```

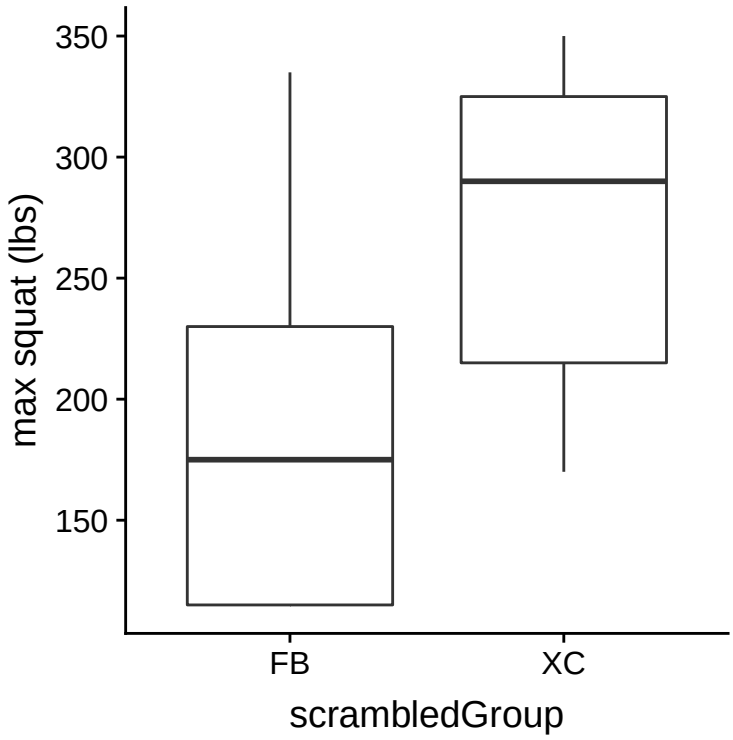

Figure 9.5: Box plots for subjects assigned to each group after scrambling group labels.

```
  tt <- t.test(
    squat ~ scrambledGroup,
    data = dfScram,
    alternative = "greater",
    var.equal = TRUE
  )
  return(tt$statistic)
}

shuffleDiff <- replicate(nRuns, shuffleAndMeasure(squatDf))

sprintf("mean t value across shuffles = %0.3f", mean(shuffleDiff))
```

```
## [1] "mean t value across shuffles = -0.004"
```

We can now look at the distribution of mean differences across the shuffled datasets. Figure 9.6 shows the histogram of the group differences across all of the random shuffles. As expected under the null hypothesis, this distribution is centered at zero.

We can see that the distribution of t values after shuffling roughly follows the theoretical t distribution under the null hypothesis (with mean=0), showing that randomization worked to generate null data. We also see something interesting if we compare the shuffled t values to the actual t value:

```
# compute number of runs on which t statistic for shuffle data was
# equal to observed t statistic

equalSum <- sum(shuffleDiff == tt$statistic)
sprintf("Number of runs on which shuffled t == observed t: %d", equalSum)
```

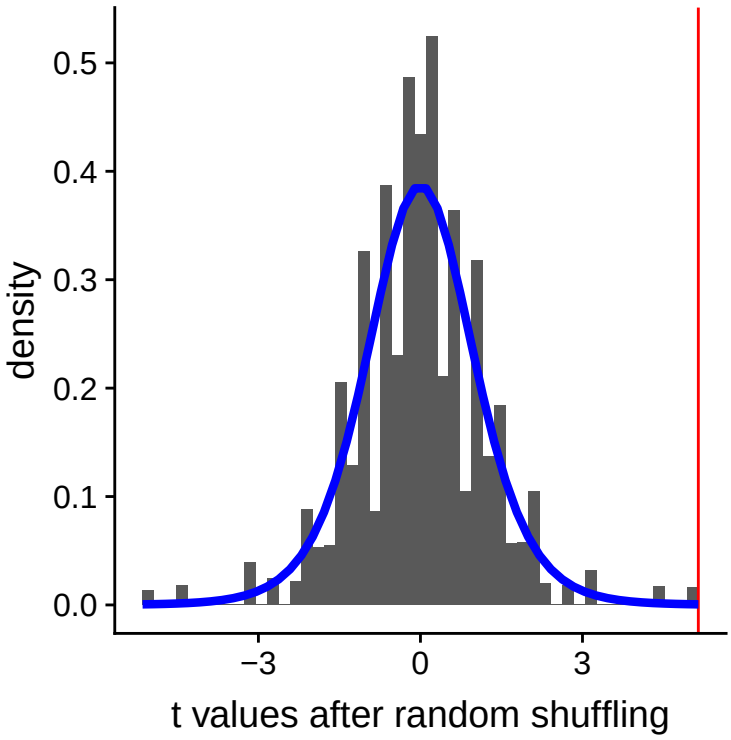

Figure 9.6: Histogram of differences between the football and cross-country groups after randomly shuffling group membership. The red line denotes the actual difference observed between the two groups, and the blue line shows the theoretical t distribution for this analysis.

```
## [1] "Number of runs on which shuffled t == observed t: 33"
```

```
# compute number of runs on which t statistic for shuffle data was
# equal to observed t statistic times -1

equalSumMinus <- sum(shuffleDiff == tt$statistic * -1)
sprintf("Number of runs on which shuffled t == observed t*-1: %d", equalSumMinus)
```

```
## [1] "Number of runs on which shuffled t == observed t*-1: 28"
```

There are 33 shuffle runs on which the t statistic for the shuffled data was exactly the same as the observed data – which means that shuffling resulted in the same labeling as the actual data! This is unlikely, but not *that* unlikely, and we can actually compute its likelihood using a bit of probability theory. The number of possible permutations of 10 items is 10! which comes out to 3628800. The number of possible rearrangements of each set of 5 is 5! which comes out to 120, so the number of possible rearrangements of two sets of five is $5! * 5!$ or 14400. Thus, we expect that 0.0039 of the random labelings should come out exactly the same as the original, which is fairly close to the 0.0033 that we see in our simulation. We have a similar expectation for the number of times that the labeling will be exactly opposite of the true labeling, giving us the negative of the observed t value.

We can compute the p-value from the randomized data by measuring how many of the shuffled values are at least as extreme as the observed value:

```
# compute p value using randomization
pvalRandomization <- mean(shuffleDiff >= tt$statistic)

sprintf(
  'p(t > %0.2f, df = 8) using randomization = %0.5f',
```

```
  tt$statistic,
  pvalRandomization
)
```

```
## [1] "p(t > 5.14, df = 8) using randomization = 0.00330"
```

This p-value is very similar to the p-value that we obtained using the t distribution, and both are quite extreme, suggesting that the observed data are very unlikely to have arisen if the null hypothesis is true - and in this case we *know* that it's not true, because we generated the data.

9.3.5.1.2 Randomization: BMI/activity example

Now let's use randomization to compute the p-value for the BMI/activity example. In this case, we will randomly shuffle the `PhysActive` variable and compute the difference between groups after each shuffle, and then compare our observed t statistic to the distribution of t statistics from the shuffled datasets.

```
# create function to shuffle BMI data

shuffleBMIstat <- function() {
  bmiDataShuffled <-
    NHANES_sample %>%
    select(BMI, PhysActive) %>%
    mutate(
      PhysActive = sample(PhysActive)
    )
  # compute the difference
  simResult <- t.test(
    BMI ~ PhysActive,
    data = bmiDataShuffled,
    var.equal = TRUE
  )
  return(simResult$statistic)
}

# run function 5000 times and save output

nRuns <- 5000
meanDiffSimDf <-
  data.frame(
    meanDiffSim = replicate(nRuns, shuffleBMIstat())
  )
```

Let's look at the results. Figure 9.7 shows the distribution of t values from the shuffled samples, and we can also compute the probability of finding a value as large or larger than the observed value:

```
# compute the empirical probability of t values larger than observed
# value under the randomization null
bmtTTest <-
  t.test(
  BMI ~ PhysActive,
  data = NHANES_sample,
  var.equal = TRUE,
  alternative = "greater"
)

bmiPvalRand <-
```

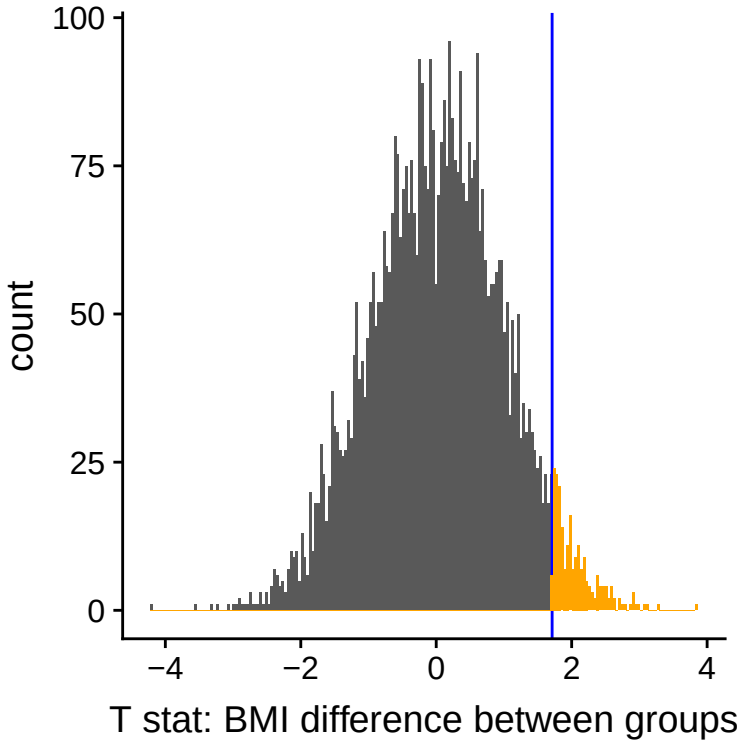

Figure 9.7: Histogram of t statistics after shuffling of group labels, with the observed value of the t statistic shown in the blue line, and values more extreme than the observed value shown in orange.

```
  mean(meanDiffSimDf$meanDiffSim >= bmtTTest$statistic)

sprintf(
  "p(mean > %0.2f, df = 248) using randomization = %0.5f",
  bmtTTest$statistic,
  bmiPvalRand
)
```

```
## [1] "p(mean > 1.71, df = 248) using randomization = 0.04380"
```

```
sprintf(
  "p(mean > %0.2f, df = 248) using parametric t-test = %0.5f",
  bmtTTest$statistic,
  bmtTTest$p.value
  )
```

```
## [1] "p(mean > 1.71, df = 248) using parametric t-test = 0.04413"
```

Again, the p-value obtained from randomization (0.044) is very similar to the one obtained using the t distribution (0.044). The advantage of the randomization test is that it doesn't require that we assume that the data from each of the groups are normally distributed, though the t-test is generally quite robust to violations of that assumption. In addition, the randomization test can allow us to compute p-values for statistics when we don't have a theoretical distribution like we do for the t-test.

We do have to make one main assumption when we use the randomization test, which we refer to as *exchangeability*. This means that all of the observations are distributed in the same way, such that we can interchange them without changing the overall distribution. The main place where this can break down is when there are related observations in the data; for example, if we had data from individuals in 4 different

families, then we couldn't assume that individuals were exchangeable, because siblings would be closer to each other than they are to individuals from other families. In general, if the data were obtained by random sampling, then the assumption of exchangeability should hold.

9.3.6 Step 6: Assess the "statistical significance" of the result

The next step is to determine whether the p-value that results from the previous step is small enough that we are willing to reject the null hypothesis and conclude instead that the alternative is true. How much evidence do we require? This is one of the most controversial questions in statistics, in part because it requires a subjective judgment – there is no "correct" answer.

Historically, the most common answer to this question has been that we should reject the null hypothesis if the p-value is less than 0.05. This comes from the writings of Ronald Fisher, who has been referred to as "the single most important figure in 20th century statistics"(Efron 1998):

> "If P is between .1 and .9 there is certainly no reason to suspect the hypothesis tested. If it is below .02 it is strongly indicated that the hypothesis fails to account for the whole of the facts. We shall not often be astray if we draw a conventional line at .05 ... it is convenient to draw the line at about the level at which we can say: Either there is something in the treatment, or a coincidence has occurred such as does not occur more than once in twenty trials" (Fisher 1925)

However, Fisher never intended $p < 0.05$ to be a fixed rule:

> "no scientific worker has a fixed level of significance at which from year to year, and in all circumstances, he rejects hypotheses; he rather gives his mind to each particular case in the light of his evidence and his ideas" [fish:1956]

Instead, it is likely that it became a ritual due to the reliance upon tables of p-values that were used before computing made it easy to compute p values for arbitrary values of a statistic. All of the tables had an entry for 0.05, making it easy to determine whether one's statistic exceeded the value needed to reach that level of significance.

The choice of statistical thresholds remains deeply controversial, and recently (Benjamin et al., 2018) it has been proposed that the standard threshold be changed from .05 to .005, making it substantially more stringent and thus more difficult to reject the null hypothesis. In large part this move is due to growing concerns that the evidence obtained from a significant result at $p < .05$ is relatively weak; we will discuss this in much more detail in our later discussion of reproducibility in Chapter 17.

9.3.6.1 Hypothesis testing as decision-making: The Neyman-Pearson approach

Whereas Fisher thought that the p-value could provide evidence regarding a specific hypothesis, the statisticians Jerzy Neyman and Egon Pearson disagreed vehemently. Instead, they proposed that we think of hypothesis testing in terms of its error rate in the long run:

> "no test based upon a theory of probability can by itself provide any valuable evidence of the truth or falsehood of a hypothesis. But we may look at the purpose of tests from another viewpoint. Without hoping to know whether each separate hypothesis is true or false, we may search for rules to govern our behaviour with regard to them, in following which we insure that, in the long run of experience, we shall not often be wrong" (J. Neyman and Pearson 1933)

That is: We can't know which specific decisions are right or wrong, but if we follow the rules, we can at least know how often our decisions will be wrong.

To understand the decision making framework that Neyman and Pearson developed, we first need to discuss statistical decision making in terms of the kinds of outcomes that can occur. There are two possible states of reality (H_0 is true, or H_0 is false), and two possible decisions (reject H_0, or fail to reject H_0). There are two ways in which we can make a correct decision:

- We can decide to reject H_0 when it is false (in the language of decision theory, we call this a *hit*)
- We can fail to reject H_0 when it is true (we call this a *correct rejection*)

There are also two kinds of errors we can make:

- We can decide to reject H_0 when it is actually true (we call this a *false alarm*, or *Type I error*)
- We can fail to reject H_0 when it is actually false (we call this a *miss*, or *Type II error*)

Neyman and Pearson coined two terms to describe the probability of these two types of errors in the long run:

- P(Type I error) = α
- P(Type II error) = β

That is, if we set α to .05, then in the long run we should make a Type I error 5% of the time. Whereas it's common to set α as .05, the standard value for β is .2 - that is, we are willing to accept that 20% of the time we will fail to detect a true effect. We will return to this below when we discuss statistical power in Section 10.3, which is the complement of Type II error.

9.3.7 What does a significant result mean?

There is a great deal of confusion about what p-values actually mean (Gigerenzer, 2004). Let's say that we do an experiment comparing the means between conditions, and we find a difference with a p-value of .01. There are a number of possible interpretations.

9.3.7.1 Does it mean that the probability of the null hypothesis being true is .01?

No. Remember that in null hypothesis testing, the p-value is the probability of the data given the null hypothesis ($P(data|H_0)$). It does not warrant conclusions about the probability of the null hypothesis given the data ($P(H_0|data)$). We will return to this question when we discuss Bayesian inference in a later chapter, as Bayes theorem lets us invert the conditional probability in a way that allows us to determine the latter probability.

9.3.7.2 Does it mean that the probability that you are making the wrong decision is .01?

No. This would be $P(H_0|data)$, but remember as above that p-values are probabilities of data under H_0, not probabilities of hypotheses.

9.3.7.3 Does it mean that if you ran the study again, you would obtain the same result 99% of the time?

No. The p-value is a statement about the likelihood of a particular dataset under the null; it does not allow us to make inferences about the likelihood of future events such as replication.

9.3.7.4 Does it mean that you have found a meaningful effect?

No. There is an important distinction between *statistical significance* and *practical significance*. As an example, let's say that we performed a randomized controlled trial to examine the effect of a particular diet on body weight, and we find a statistically significant effect at p<.05. What this doesn't tell us is how much weight was actually lost, which we refer to as the *effect size* (to be discussed in more detail in Chapter 10). If we think about a study of weight loss, then we probably don't think that the loss of ten ounces (i.e. the weight of a bag of potato chips) is practically significant. Let's look at our ability to detect a significant difference of 1 ounce as the sample size increases.

```
# create simulated data for weight loss trial

weightLossTrial <- function(nPerGroup, weightLossOz = 1) {
  # mean and SD in Kg based on NHANES adult dataset
  kgToOz <- 35.27396195 # conversion constant for Kg to Oz
  meanOz <- 81.78 * kgToOz
  sdOz <- 21.29 * kgToOz
  # create data
  controlGroup <- rnorm(nPerGroup) * sdOz + meanOz
  expGroup <- rnorm(nPerGroup) * sdOz + meanOz - weightLossOz
  ttResult <- t.test(expGroup, controlGroup)
  return(c(
    nPerGroup, weightLossOz, ttResult$p.value,
    diff(ttResult$estimate)
  ))
}

nRuns <- 1000
sampSizes <- 2**seq(5,17) # powers of 2

simResults <- c() ## create an empty list to add results onto
for (i in 1:length(sampSizes)) {
  tmpResults <- replicate(
    nRuns,
    weightLossTrial(sampSizes[i], weightLossOz = 10)
  )
  summaryResults <- c(
    tmpResults[1, 1], tmpResults[2, 1],
    sum(tmpResults[3, ] < 0.05),
    mean(tmpResults[4, ])
  )
  simResults <- rbind(simResults, summaryResults)
}


simResultsDf <-
  as.tibble(simResults) %>%
  rename(
    sampleSize = V1,
    effectSizeLbs = V2,
    nSigResults = V3,
    meanEffect = V4
  ) %>%
  mutate(pSigResult = nSigResults / nRuns)
```

Figure 9.8 shows how the proportion of significant results increases as the sample size increases, such that with a very large sample size (about 262,000 total subjects), we will find a significant result in more than 90% of studies when there is a 1 ounce weight loss. While these are statistically significant, most physicians would not consider a weight loss of one ounce to be practically or clinically significant. We will explore this relationship in more detail when we return to the concept of *statistical power* in Section 10.3, but it should already be clear from this example that statistical significance is not necessarily indicative of practical significance.

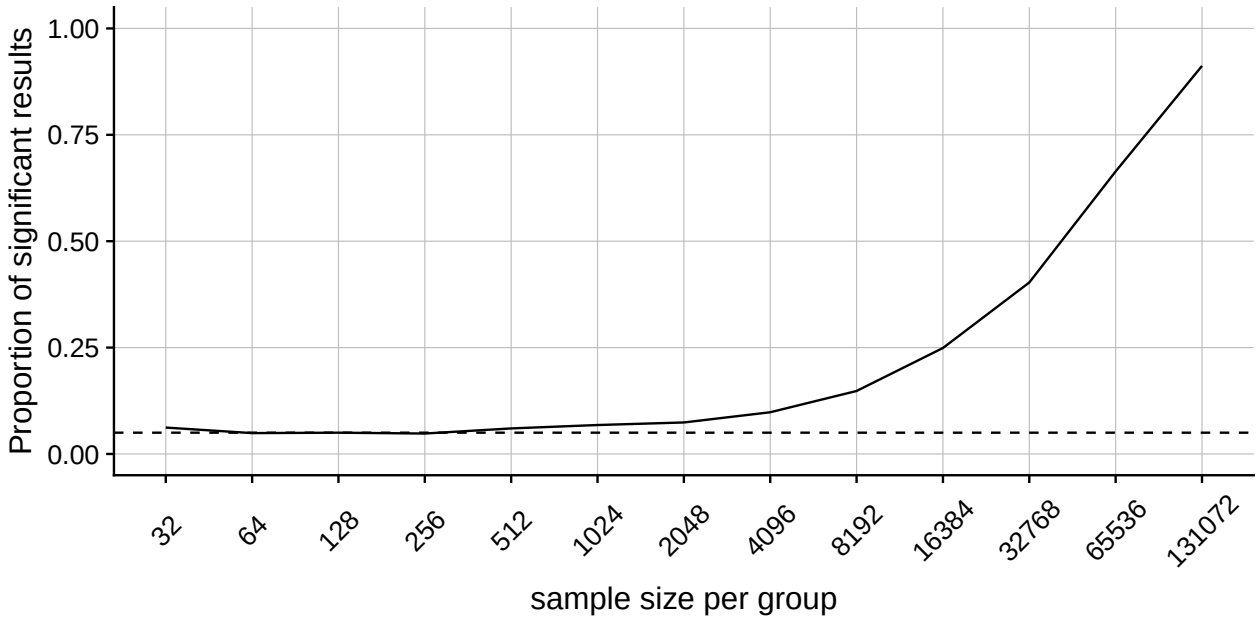

Figure 9.8: The proportion of signifcant results for a very small change (1 ounce, which is about .001 standard deviations) as a function of sample size.

9.4 NHST in a modern context: Multiple testing

So far we have discussed examples where we are interested in testing a single statistical hypothesis, and this is consistent with traditional science which often measured only a few variables at a time. However, in modern science we can often measure millions of variables per individual. For example, in genetic studies that quantify the entire genome, there may be many millions of measures per individual, and in brain imaging we often collect data from more than 100,000 locations in the brain at once. When standard hypothesis testing is applied in these contexts, bad things can happen unless we take appropriate care.

Let's look at an example to see how this might work. There is great interest in understanding the genetic factors that can predispose individuals to major mental illnesses such as schizophrenia, because we know that about 80% of the variation between individuals in the presence of schizophrenia is due to genetic differences. The Human Genome Project and the ensuing revolution in genome science has provided tools to examine the many ways in which humans differ from one another in their genomes. One approach that has been used in recent years is known as a *genome-wide association study* (GWAS), in which the genome of each individual is characterized at one million or more places in their genome to determine which letters of the genetic code (which we call "variants") they have at that location. After these have been determined, the researchers perform a statistical test at each location in the genome to determine whether people diagnosed with schizoprenia are more or less likely to have one specific variant at that location.

Let's imagine what would happen if the researchers simply asked whether the test was significant at $p<.05$ at each location, when in fact there is no true effect at any of the locations. To do this, we generate a large number of simulated t values from a null distribution, and ask how many of them are significant at $p<.05$. Let's do this many times, and each time count up how many of the tests come out as significant (see Figure 9.9).

```
# simulate 1500 studies with 10,000 tests each, thresholded at p < .05

nRuns <- 1500 # number of simulated studies to run
nTests <- 10000 # number of simulated genes to test in each run
```

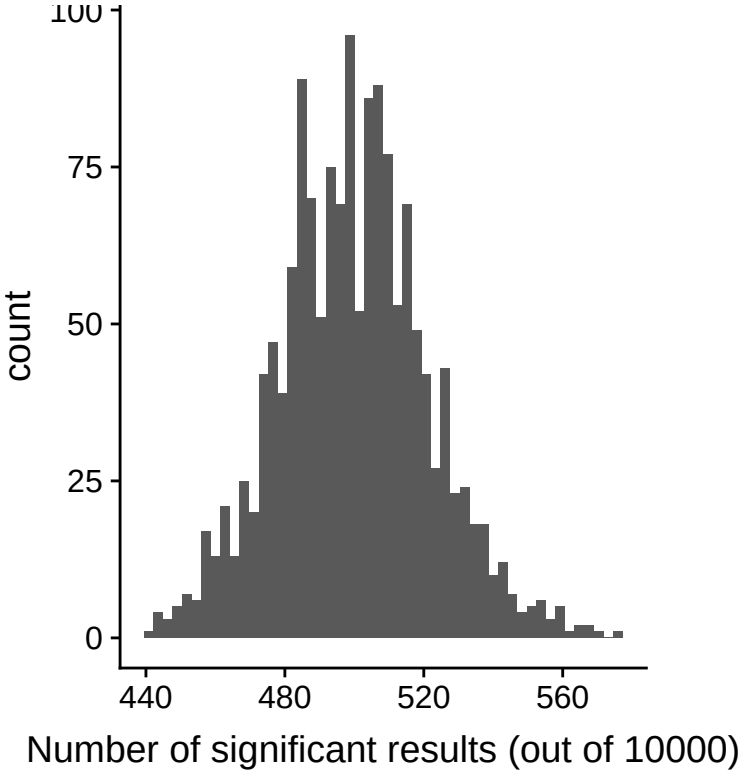

Figure 9.9: A histogram of the number of significant results in each set of 1 million statistical tests, when there is in fact no true effect.

```
uncAlpha <- 0.05 # alpha level

uncOutcome <- replicate(nRuns, sum(rnorm(nTests) < qnorm(uncAlpha)))

sprintf("mean proportion of significant tests per run: %0.2f", mean(uncOutcome) / nTests)
```

```
## [1] "mean proportion of significant tests per run: 0.05"
```

```
# compute proportion of studies with at least one false positive result,
# known as the familywise error rate
sprintf("familywise error rate: %0.3f", mean(uncOutcome > 0))
```

```
## [1] "familywise error rate: 1.000"
```

This shows that about 5% of all of the tests were significant in each run, meaning that if we were to use $p < .05$ as our threshold for statistical significance, then even if there were no truly significant relationships present, we would still "find" about 500 genes that were seemingly significant (the expected number of significant results is simply $n * \alpha$). That is because while we controlled for the error per test, we didn't control the *familywise error*, or the error across all of the tests, which is what we really want to control if we are going to be looking at the results from a large number of tests. Using $p<.05$, our familywise error rate in the above example is one – that is, we are pretty much guaranteed to make at least one error in any particular study.

A simple way to control for the familywise error is to divide the alpha level by the number of tests; this is known as the *Bonferroni* correction, named after the Italian statistician Carlo Bonferroni. Using the data from our example above, we see in Figure 9.10 that only about 5 percent of studies show any significant results using the corrected alpha level of 0.000005 instead of the nominal level of .05. We have effectively

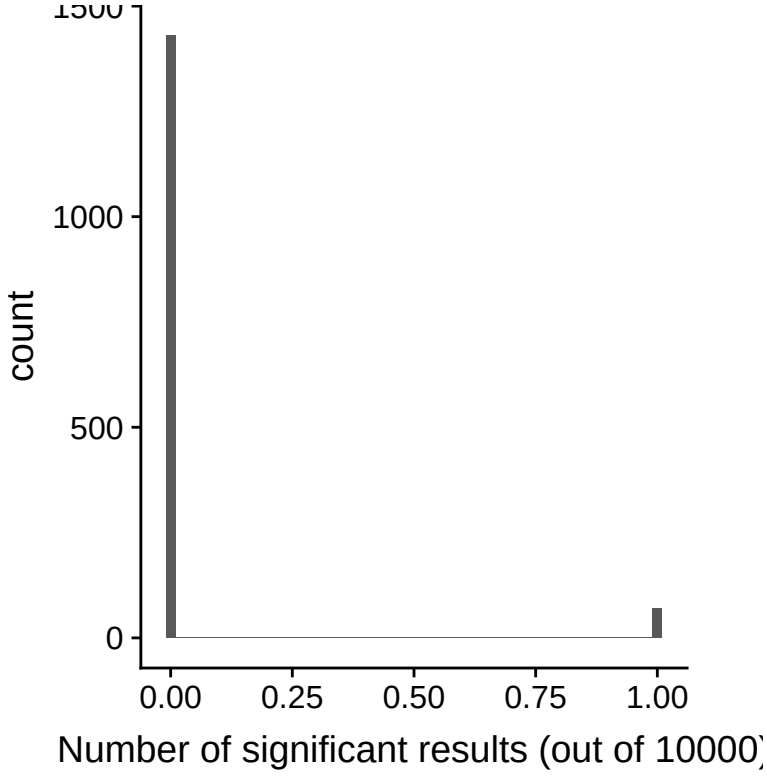

Figure 9.10: A histogram of the number of significant results across all simulation runs after applying the Bonferroni correction for multiple tests.

controlled the familywise error, such that the probability of making *any* errors in our study is controlled at right around .05.

```
# compute Bonferroni-corrected alpha
corAlpha <- 0.05 / nTests

corOutcome <- replicate(nRuns, sum(rnorm(nTests) < (qnorm(corAlpha))))

sprintf("corrected familywise error rate: %0.3f", mean(corOutcome > 0))
```

```
## [1] "corrected familywise error rate: 0.046"
```

9.5 Suggested readings

- Mindless Statistics, by Gerd Gigerenzer

Chapter 10

Confidence intervals, effect sizes, and statistical power

In the previous chapter we discussed how we can use data to test hypotheses. Those methods provided a binary answer: we either reject or fail to reject the null hypothesis. However, this kind of decision overlooks a couple of important questions. First, we would like to know how much uncertainty we have about the answer (regardless of which way it goes). In addition, sometimes we don't have a clear null hypothesis, so we would like to see what range of estimates are consistent with the data. Second, we would like to know how large the effect actually is, since as we saw in the weight loss example in the previous chapter, a statistically significant effect is not necessarily a practically important effect.

In this chapter we will discuss methods to address these two questions: confidence intervals to provide a measure of our uncertainty about our estimates, and effect sizes to provide a standardized way to understand how large the effects are. We will also discuss the concept of *statistical power* which tells us how well we can expect to find any true effects that might exist.

10.1 Confidence intervals

So far in the book we have focused on estimating the specific value of a statistic. For example, let's say we want to estimate the mean weight of adults in the NHANES dataset. We could take a sample from the dataset and estimate the mean:

```
# take a sample from adults in NHANES and summarize their weight

sampSize <- 250
NHANES_sample <- sample_n(NHANES_adult, sampSize)

sample_summary <-
  NHANES_sample %>%
  summarize(
    meanWeight = mean(Weight),
    sdWeight = sd(Weight)
  )
pander(sample_summary)
```

meanWeight	sdWeight
82.77	22.27

In this sample, the mean weight was 82.77 kilograms. We refer to this as a *point estimate* since it provides us with a single number to describe the difference. However, we know from our earlier discussion of sampling error that there is some uncertainty about this estimate, which is described by the standard error. You should also remember that the standard error is determined by two components: the population standard deviation (which is the numerator), and the square root of the sample size (which is in the denominator). The population standard deviation is an unknown but fixed parameter that is not under our control, whereas the sample size *is* under our control. Thus, we can decrease our uncertainty about the estimate by increasing our sample size – up to the limit of the entire population size, at which point there is no uncertainty at all because we can just calculate the population parameter directly from the data of the entire population.

You may also remember that earlier we introduced the concept of a *confidence interval*, which is a way of describing our uncertainty about a statistical estimate. Remember that a confidence interval describes an interval that will on average contain the true population parameter with a given probability; for example, the 95% confidence interval is an interval that will capture the true population parameter 95% of the time. Note again that this is not a statement about the population parameter; any particular confidence interval either does or does not contain the true parameter. As Jerzy Neyman, the inventor of the confidence interval, said:

> "The parameter is an unknown constant and no probability statement concerning its value may be made."(J Neyman 1937)

The confidence interval for the mean is computed as:

$$CI = point\ estimate\ \pm critical\ value$$

where the critical value is determined by the sampling distribution of the estimate. The important question, then, is what that sampling distribution is.

10.1.1 Confidence intervals using the normal distribution

If we know the population standard deviation, then we can use the normal distribution to compute a confidence interval. We usually don't, but for our example of the NHANES dataset we do (it's 21.3 for weight).

Let's say that we want to compute a 95% confidence interval for the mean. The critical value would then be the values of the standard normal distribution that capture 95% of the distribution; these are simply the 2.5th percentile and the 97.5th percentile of the distribution, which we can compute using the `qnorm()` function in R, and come out to ±1.96. Thus, the confidence interval for the mean ($\bar{X}$) is:

$$CI = \bar{X} \pm 1.96 * SE$$

Using the estimated mean from our sample (82.77) and the known population standard deviation, we can compute the confidence interval of [80.13,85.41].

10.1.2 Confidence intervals using the t distribution

As stated above, if we knew the population standard deviation, then we could use the normal distribution to compute our confidence intervals. However, in general we don't – in which case the t distribution is more appropriate as a sampling distribution. Remember that the t distribution is slightly broader than the normal distribution, especially for smaller samples, which means that the confidence intervals will be slightly wider than they would if we were using the normal distribution. This incorporates the extra uncertainty that arises when we make conclusions based on small samples.

We can compute the 95% confidence interval in a way similar to the normal distribution example above, but the critical value is determined by the 2.5th percentile and the 97.5th percentile of the t distribution, which we can compute using the `qt()` function in R. Thus, the confidence interval for the mean ($\bar{X}$) is:

$$CI = \bar{X} \pm t_{crit} * SE$$

where t_{crit} is the critical t value. For the NHANES weight example (with sample size of 250), the confidence interval would be:

```
# compute confidence intervals for weight in NHANES data

sample_summary <-
  sample_summary %>%
  mutate(
    cutoff_lower = qt(0.025, sampSize),
    cutoff_upper = qt(0.975, sampSize),
    CI_lower = meanWeight + cutoff_lower * sdWeight / sqrt(sampSize),
    CI_upper = meanWeight + cutoff_upper * sdWeight / sqrt(sampSize)
  )
pander(sample_summary)
```

meanWeight	sdWeight	cutoff_lower	cutoff_upper	CI_lower	CI_upper
82.77	22.27	-1.97	1.97	80	85.54

Remember that this doesn't tell us anything about the probability of the true population value falling within this interval, since it is a fixed parameter (which we know is 81.77 because we have the entire population in this case) and it either does or does not fall within this specific interval (in this case, it does). Instead, it tells us that in the long run, if we compute the confidence interval using this procedure, 95% of the time that confidence interval will capture the true population parameter.

10.1.3 Confidence intervals and sample size

Because the standard error decreases with sample size, the means confidence interval should get narrower as the sample size increases, providing progressively tighter bounds on our estimate. Figure 10.1 shows an example of how the confidence interval would change as a function of sample size for the weight example. From the figure it's evident that the confidence interval becomes increasingly tighter as the sample size increases, but increasing samples provide diminishing returns, consistent with the fact that the denominator of the confidence interval term is proportional to the square root of the sample size.

10.1.4 Computing confidence intervals using the bootstrap

In some cases we can't assume normality, or we don't know the sampling distribution of the statistic. In these cases, we can use the bootstrap (which we introduced in Chapter 8). As a reminder, the bootstrap involves repeatedly resampling the data *with replacement*, and then using the distribution of the statistic computed on those samples as a surrogate for the sampling distribution of the statistic.

Earlier we ran the bootstrap using hand-crafted code, but R includes a package called `boot` that we can use to run the bootstrap and compute confidence intervals. Let's use it to compute the confidence interval for weight in our NHANES sample.

```
# compute bootstrap confidence intervals on NHANES weight data

meanWeight <- function(df, foo) {
  return(mean(df[foo, ]$Weight))
}
```

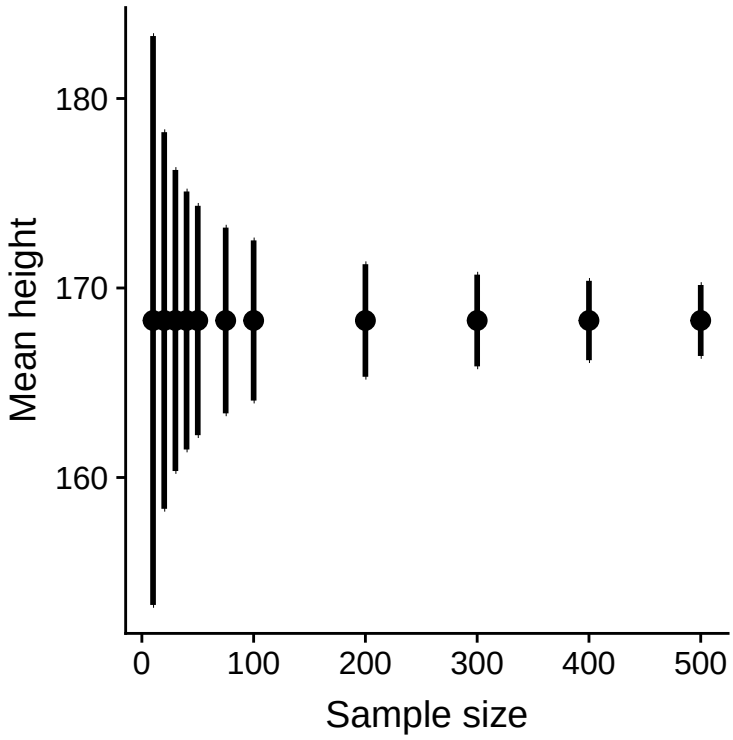

Figure 10.1: An example of the effect of sample size on the width of the confidence interval for the mean.

```
bs <- boot(NHANES_sample, meanWeight, 1000)

# use the percentile bootstrap
bootci <- boot.ci(bs, type = "perc")
print("Bootstrap confidence intervals:")
```

```
## [1] "Bootstrap confidence intervals:"
```

```
tibble(
  lower = bootci$perc[4],
  upper = bootci$perc[5]
) %>%
  pander()
```

lower	upper
80.12	85.71

These values are fairly close to the values obtained using the t distribution above, though not exactly the same.

10.1.5 Relation of confidence intervals to hypothesis tests

There is a close relationship between confidence intervals and hypothesis tests. In particular, if the confidence interval does not include the null hypothesis, then the associated statistical test would be statistically significant. For example, if you are testing whether the mean of a sample is greater than zero with $\alpha = 0.05$, you could simply check to see whether zero is contained within the 95% confidence interval for the mean.

Things get trickier if we want to compare the means of two conditions (Schenker and Gentleman 2001). There are a couple of situations that are clear. First, if each mean is contained within the confidence interval for the other mean, then there is certainly no significant difference at the chosen confidence level. Second, if there is no overlap between the confidence intervals, then there is certainly a significant difference at the chosen level; in fact, this test is substantially *conservative*, such that the actual error rate will be lower than the chosen

10.2.1 Cohen's D

One of the most common measures of effect size is known as *Cohen's d*, named after the statistician Jacob Cohen (who is most famous for his 1994 paper titled "The Earth Is Round (p < .05)"). It is used to quantify the difference between two means, in terms of their standard deviation:

$$d = \frac{\bar{X}_1 - \bar{X}_2}{s}$$

where $\bar{X}_1$ and $\bar{X}_2$ are the means of the two groups, and s is the pooled standard deviation (which is a combination of the standard deviations for the two samples, weighted by their sample sizes):

$$s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

where n_1 and n_2 are the sample sizes and s_1^2 and s_2^2 are the standard deviations for the two groups respectively.

There is a commonly used scale for interpreting the size of an effect in terms of Cohen's d:

D	Interpretation
0.2	small
0.5	medium
0.8	large

It can be useful to look at some commonly understood effects to help understand these interpretations.

```
# compute effect size for gender difference in NHANES

NHANES_sample <-
  NHANES_adult %>%
  drop_na(Height) %>%
  sample_n(250)

hsum <-
  NHANES_sample %>%
  group_by(Gender) %>%
  summarize(
    meanHeight = mean(Height),
    varHeight = var(Height),
    n = n()
  )


#pooled SD
s_height_gender <- sqrt(
  ((hsum$n[1] - 1) * hsum$varHeight[1] + (hsum$n[2] - 1) * hsum$varHeight[2]) /
    (hsum$n[1] + hsum$n[2] - 2)
)

#cohen's d
d_height_gender <- (hsum$meanHeight[2] - hsum$meanHeight[1]) / s_height_gender

sprintf("Cohens d for male vs. female height = %0.2f", d_height_gender)
```

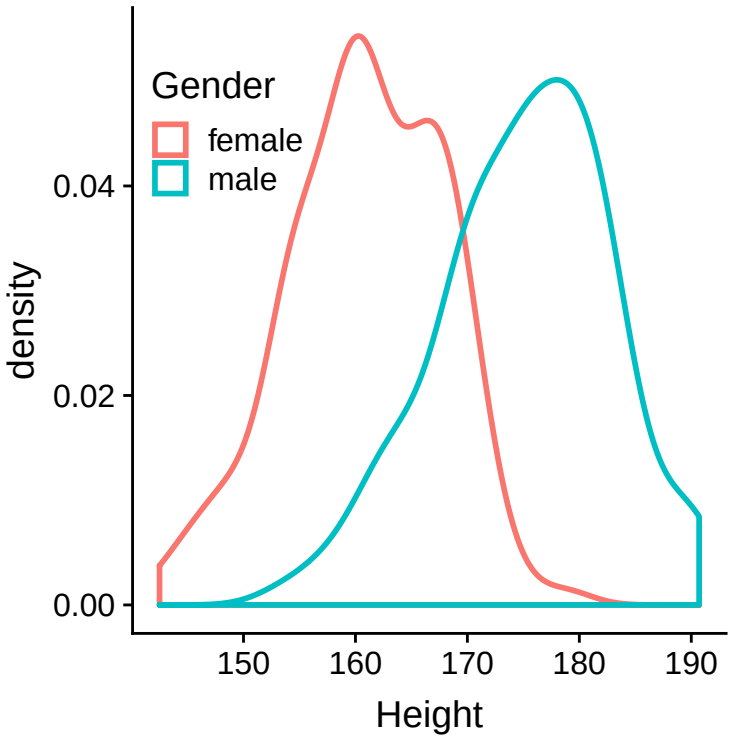

Figure 10.2: Smoothed histogram plots for male and female heights in the NHANES dataset, showing clearly distinct but also clearly overlapping distributions.

```
## [1] "Cohens d for male vs. female height = 1.95"
```

The effect size for gender differences in height (d = 1.95) is huge by reference to our table above. We can also see this by looking at the distributions of male and female heights in our sample. Figure 10.2 shows that the two distributions are quite well separated, though still overlapping, highlighting the fact that even when there is a very large effect size for the difference between two groups, there will be individuals from each group that are more like the other group.

It is also worth noting that we rarely encounter effects of this magnitude in science, in part because they are such obvious effects that we don't need scientific research to find them. As we will see in Chapter 17 on reproducibility, very large reported effects in scientific research often reflect the use of questionable research practices rather than truly huge effects in nature. It is also worth noting that even for such a huge effect, the two distributions still overlap - there will be some females who are taller than the average male, and vice versa. For most interesting scientific effects, the degree of overlap will be much greater, so we shouldn't immediately jump to strong conclusions about different populations based on even a large effect size.

10.2.2 Pearson's r

Pearson's r, also known as the *correlation coefficient*, is a measure of the strength of the linear relationship between two continuous variables. We will discuss correlation in much more detail in Chapter 13, so we will save the details for that chapter; here, we simply introduce r as a way to quantify the relation between two variables.

r is a measure that varies from -1 to 1, where a value of 1 represents a perfect positive relationship between the variables, 0 represents no relationship, and -1 represents a perfect negative relationship. Figure 10.3 shows examples of various levels of correlation using randomly generated data.

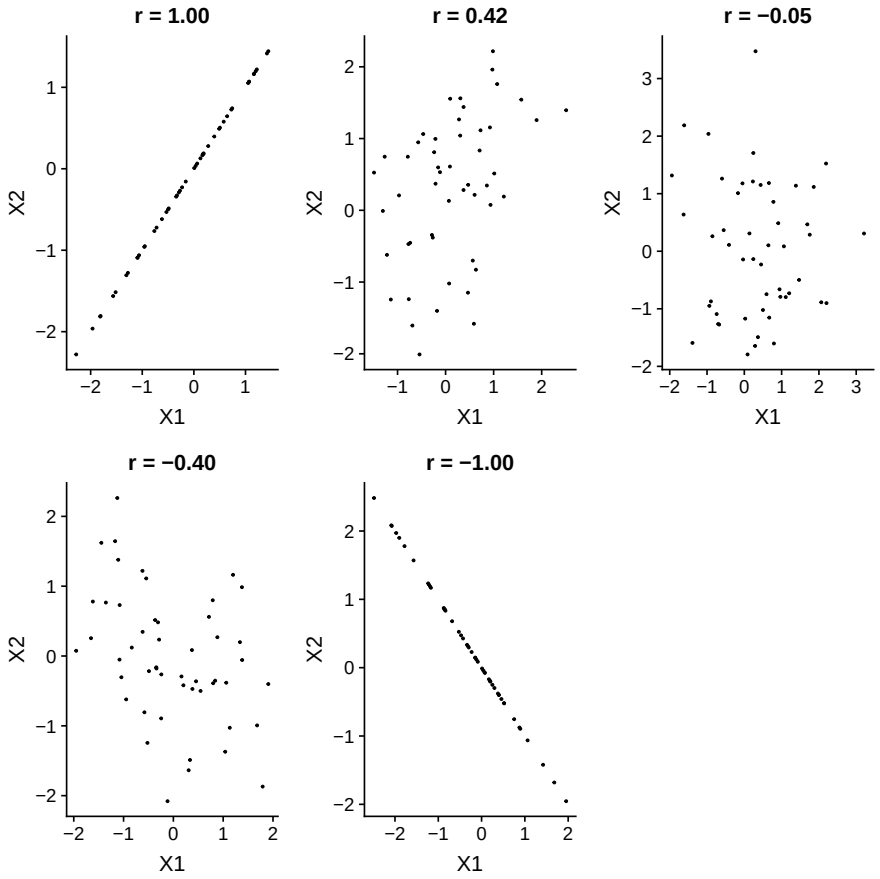

Figure 10.3: Examples of various levels of Pearson's r.

10.2.3 Odds ratio

In our earlier discussion of probability we discussed the concept of odds – that is, the relative likelihood of some event happening versus not happening:

$$odds\ of\ A = \frac{P(A)}{P(\neg A)}$$

The odds ratio is simply the ratio of two odds. For example, let's take the case of smoking and lung cancer. A study published in the International Journal of Cancer in 2012 (Pesch et al. 2012) combined data regarding the occurrence of lung cancer in smokers and individuals who have never smoked across a number of different studies. Note that these data come from case-control studies, which means that participants in the studies were recruited because they either did or did not have cancer; their smoking status was then examined. These numbers thus do not represent the prevalence of cancer amongst smokers in the general population – but they can tell us about the relationship between cancer and smoking.

```
# create table for cancer occurrence depending on smoking status
smokingDf <- tibble(
  NeverSmoked = c(2883, 220),
  CurrentSmoker = c(3829, 6784),
  row.names = c("NoCancer", "Cancer")
)
pander(smokingDf)
```

NeverSmoked	CurrentSmoker	row.names
2883	3829	NoCancer
220	6784	Cancer

We can convert these numbers to odds ratios for each of the groups:

```
# convert smoking data to odds

smokingDf <-
  smokingDf %>%
  mutate(
    pNeverSmoked = NeverSmoked / sum(NeverSmoked),
    pCurrentSmoker = CurrentSmoker / sum(CurrentSmoker)
  )

oddsCancerNeverSmoked <- smokingDf$NeverSmoked[2] / smokingDf$NeverSmoked[1]
oddsCancerCurrentSmoker <- smokingDf$CurrentSmoker[2] / smokingDf$CurrentSmoker[1]
```

The odds of someone having lung cancer who has never smoked is 0.08 whereas the odds of a current smoker having lung cancer is 1.77. The ratio of these odds tells us about the relative likelihood of cancer between the two groups:

```
#compute odds ratio

oddsRatio <- oddsCancerCurrentSmoker/oddsCancerNeverSmoked
sprintf('odds ratio of cancer for smokers vs. nonsmokers: %0.3f',oddsRatio)
```

```
## [1] "odds ratio of cancer for smokers vs. nonsmokers: 23.218"
```

The odds ratio of 23.22 tells us that the odds of cancer in smokers are roughly 23 times higher than never-smokers.

10.3 Statistical power

Remember from the previous chapter that under the Neyman-Pearson hypothesis testing approach, we have to specify our level of tolerance for two kinds of errors: False positives (which they called *Type I error*) and false negatives (which they called *Type II error*). People often focus heavily on Type I error, because making a false positive claim is generally viewed as a very bad thing; for example, the now discredited claims by Wakefield (1999) that autism was associated with vaccination led to anti-vaccine sentiment that has resulted in substantial increases in childhood diseases such as measles. Similarly, we don't want to claim that a drug cures a disease if it really doesn't. That's why the tolerance for Type I errors is generally set fairly low, usually at $\alpha = 0.05$. But what about Type II errors?

The concept of *statistical power* is the complement of Type II error – that is, it is the likelihood of finding a positive result given that it exists:

$$power = 1 - \beta$$

Another important aspect of the Neyman-Pearson model that we didn't discuss above is the fact that in addition to specifying the acceptable levels of Type I and Type II errors, we also have to describe a specific alternative hypothesis – that is, what is the size of the effect that we wish to detect? Otherwise, we can't interpret β – the likelihood of finding a large effect is always going to be higher than finding a small effect, so β will be different depending on the size of effect we are trying to detect.

There are three factors that can affect power:

- Sample size: Larger samples provide greater statistical power
- Effect size: A given design will always have greater power to find a large effect than a small effect (because finding large effects is easier)
- Type I error rate: There is a relationship between Type I error and power such that (all else being equal) decreasing Type I error will also decrease power.

We can see this through simulation. First let's simulate a single experiment, in which we compare the means of two groups using a standard t-test. We will vary the size of the effect (specified in terms of Cohen's d), the Type I error rate, and the sample size, and for each of these we will examine how the proportion of significant results (i.e. power) is affected. Figure 10.4 shows an example of how power changes as a function of these factors.

```
# Simulate power as a function of sample size, effect size, and alpha

# create a set of functions to generate simulated results
powerDf <-
  expand.grid(
    sampSizePerGroup = c(12, 24, 48, 96),
    effectSize = c(.2, .5, .8),
    alpha = c(0.005, 0.05)
  ) %>%
  tidyr::expand(effectSize, sampSizePerGroup, alpha) %>%
  group_by(effectSize, sampSizePerGroup, alpha)

runPowerSim <- function(df, nsims = 1000) {
  p <- array(NA, dim = nsims)
  for (s in 1:nsims) {
    data <- data.frame(
      y = rnorm(df$sampSizePerGroup * 2),
      group = array(0, dim = df$sampSizePerGroup * 2)
    )
```

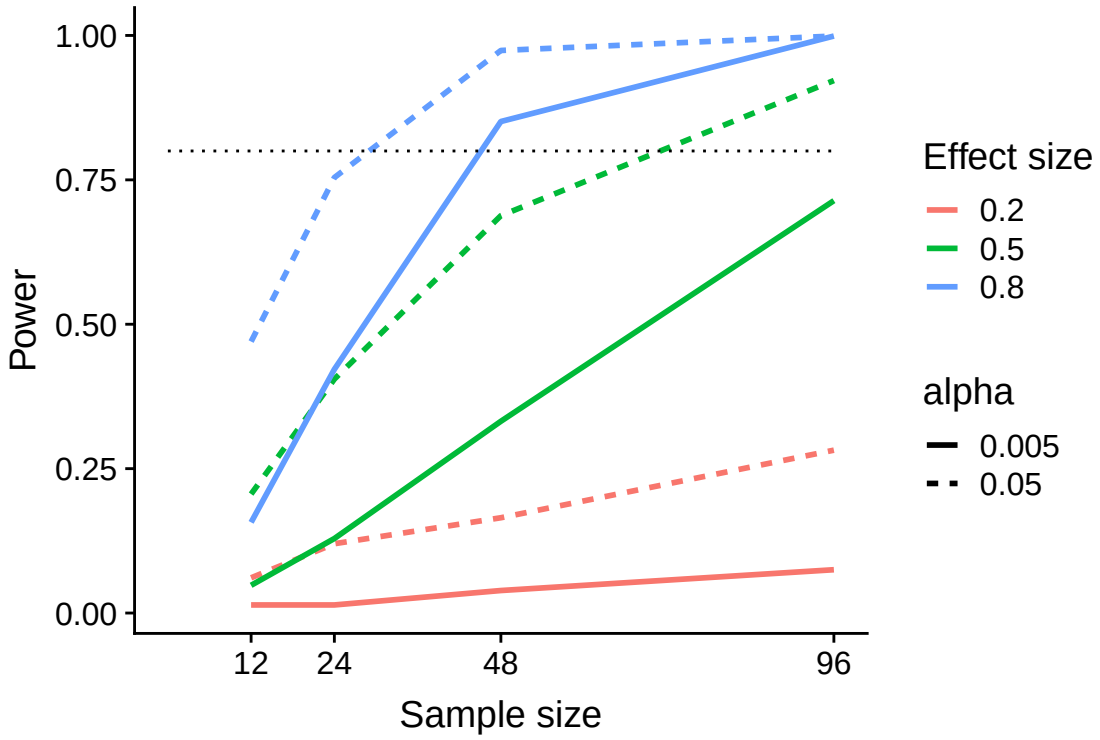

Figure 10.4: Results from power simulation, showing power as a function of sample size, with effect sizes shown as different colors, and alpha shown as line type. The standard criterion of 80 percent power is shown by the dotted black line.

```
    data$group[1:df$sampSizePerGroup] <- 1
    data$y[data$group == 1] <- data$y[data$group == 1] + df$effectSize
    tt <- t.test(y ~ group, data = data)
    p[s] <- tt$p.value
  }
  return(data.frame(power = mean(p < df$alpha)))
}

# run the simulation
powerSimResults <- powerDf %>%
  do(runPowerSim(.))
```

This simulation shows us that even with a sample size of 96, we will have relatively little power to find a small effect ($d = 0.2$) with $\alpha = 0.005$. This means that a study designed to do this would be *futile* – that is, it is almost guaranteed to find nothing even if a true effect of that size exists.

There are at least two important reasons to care about statistical power, one of which we discuss here and the other of which we will return to in Chapter 17. If you are a researcher, you probably don't want to spend your time doing futile experiments. Running an underpowered study is essentially futile, because it means that there is a very low likelihood that one will find an effect, even if it exists.

10.3.1 Power analysis

Fortunately, there are tools available that allow us to determine the statistical power of an experiment. The most common use of these tools is in planning an experiment, when we would like to determine how large our

sample needs to be in order to have sufficient power to find our effect of interest.

Let's say that we are interested in running a study of how a particular personality trait differs between users of iOS versus Android devices. Our plan is collect two groups of individuals and measure them on the personality trait, and then compare the two groups using a t-test. In order to determine the necessary sample size, we can use the `pwr.t.test()` function from the `pwr` library.

```
# power analysis for Cohen's d = 0.5, for 80% power with alpha = 0.05
pwr.t.test(d = 0.5, power = 0.8, sig.level = 0.05)
```

```
##
##      Two-sample t test power calculation
##
##               n = 64
##               d = 0.5
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

This tells us that we would need at least 64 subjects in each group in order to have sufficient power to find a medium-sized effect. It's always important to run a power analysis before one starts a new study, to make sure that the study won't be futile due to a sample that is too small.

It might have occurred to you that if the effect size is large enough, then the necessary sample will be very small. For example, if we run the same power analysis with an effect size of d=3, then we will see that we only need about 3 subjects in each group to have sufficient power to find the difference.

```
##
##      Two-sample t test power calculation
##
##               n = 3.1
##               d = 3
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

However, it's rare in science to be doing an experiment where we expect to find such a large effect – just as we don't need statistics to tell us that 16-year-olds are taller than than 6-year-olds. When we run a power analysis, we need to specify an effect size that is plausible for our study, which would usually come from previous research. However, in Chapter 17 we will discuss a phenomenon known as the "winner's curse" that likely results in published effect sizes being larger than the true effect size, so this should also be kept in mind.

10.4 Suggested readings

- Robust misinterpretation of confidence intervals, by Hoekstra et al.

Chapter 11

Bayesian statistics

In this chapter we will take up the approach to statistical modeling and inference that stands in contrast to the null hypothesis testing framework that you encountered in Chapter 9. This is known as "Bayesian statistics" after the Reverend Thomas Bayes, whose theorem you have already encountered in Chapter 3. In this chapter you will learn how Bayes' theorem provides a way of understanding data that solves many of the conceptual problems that we discussed regarding null hypothesis testing.

11.1 Generative models

Say you are walking down the street and a friend of yours walks right by but doesn't say hello. You would probably try to decide why this happened – Did they not see you? Are they mad at you? Are you suddenly cloaked in a magic invisibility shield? One of the basic ideas behind Bayesian statistics is that we want to infer the details of how the data are being generated, based on the data themselves. In this case, you want to use the data (i.e. the fact that your friend did not say hello) to infer the process that generated the data (e.g. whether or not they actually saw you, how they feel about you, etc).

The idea behind a generative model is that we observe data that are generated by a *latent* (unseen) process, usually with some amount of randomness in the process. In fact, when we take a sample of data from a population and estimate a parameter from the sample, what we are doing in essence is trying to learn the value of a latent variable (the population mean) that gives rise through sampling to the observed data (the sample mean).

If we know the value of the latent variable, then it's easy to reconstruct what the observed data should look like. For example, let's say that we are flipping a coin that we know to be fair. We can describe the coin by a binomial distribution with a value of p=0.5, and then we could generate random samples from such a distribution in order to see what the observed data should look like. However, in general we are in the opposite situation: We don't know the value of the latent variable of interest, but we have some data that we would like to use to estimate it.

11.2 Bayes' theorem and inverse inference

The reason that Bayesian statistics has its name is because it takes advantage of Bayes' theorem to make inferences from data back to some features of the (latent) model that generated the data. Let's say that we want to know whether a coin is fair. To test this, we flip the coin 10 times and come up with 7 heads. Before this test we were pretty sure that the coin was fair (i.e., that $p_{heads} = 0.5$), but these data certainly give us pause. We already know how to compute the conditional probability that we would flip 7 or more heads out of 10 if the coin is really fair ($P(n \geq 7|p_{heads} = 0.5)$), using the binomial distribution:

```
# compute the conditional probability of 7 or more heads when p(heads)=0.5
sprintf(
  "p(7 or more heads | p(heads) = 0.5) = %.3f",
  pbinom(7, 10, .5, lower.tail = FALSE)
)
```

```
## [1] "p(7 or more heads | p(heads) = 0.5) = 0.055"
```

That is a fairly small number, but this number doesn't really answer the question that we are asking – it is telling us about the likelihood of 7 or more heads given some particular probability of heads, whereas what we really want to know is the probability of heads. This should sound familiar, as it's exactly the situation that we were in with null hypothesis testing, which told us about the likelihood of data rather than the likelihood of hypotheses.

Remember that Bayes' theorem provides us with the tool that we need to invert a conditional probability:

$$P(H|D) = \frac{P(D|H) * P(H)}{P(D)}$$

We can think of this theorem as having four parts:

- prior ($P(H)$): Our degree of belief about hypothesis H before seeing the data D
- likelihood ($P(D|H)$): How likely are the observed data D under hypothesis H?
- marginal likelihood ($P(D)$): How likely are the observed data, combining over all possible hypotheses?
- posterior ($P(H|D)$): Our updated belief about hypothesis H, given the data D

Here we see one of the primary differences between frequentist and Bayesian statsistics. Frequentists do not believe in the idea of a probability of a hypothesis (i.e., our degree of belief about a hypothesis) – for them, a hypothesis is either true or it isn't. Another way to say this is that for the frequentist, the hypothesis is fixed and the data are random, which is why frequentist inference focuses on describing the probability of data given a hypothesis (i.e. the p-value). Bayesians, on the other hand, are comfortable making probability statements about both data and hypotheses.

11.3 Doing Bayesian estimation

We ultimately want to use Bayesian statistics to test hypotheses, but before we do that we need to estimate the parameters that are necessary to test the hypothesis. Here we will walk through the process of Bayesian estimation. Let's use another screening example: Airport security screening. If you fly a lot like I do, it's just a matter of time until one of the random explosive screenings comes back positive; I had the particularly unfortunate experience of this happening soon after September 11, 2001, when airport security staff were especially on edge.

What the security staff want to know is what is the likelihood that a person is carrying an explosive, given that the machine has given a positive test. Let's walk through how to calculate this value using Bayesian analysis.

11.3.1 Specifying the prior

To use Bayes' theorem, we first need to specify the prior probability for the hypothesis. In this case, we don't know the real number but we can assume that it's quite small. According to the FAA, there were 971,595,898 air passengers in the U.S. in 2017. For this example, let's say that one out of those travelers was carrying an explosive in their bag

```
prior <- 1/971595898
```

11.3.2 Collect some data

The data are composed of the results of the explosive screening test. Let's say that the security staff runs the bag through their testing apparatus 10 times, and it gives a positive reading on 9 of the 10 tests.

```
nTests <- 10
nPositives <- 9
```

11.3.3 Computing the likelihood

We want to compute the likelihood of the data under the hypothesis that there is an explosive in the bag. Let's say that we know that the sensitivity of the test is 0.99 – that is, when a device is present, it will detect it 99% of the time. To determine the likelihood of our data under the hypothesis that a device is present, we can treat each test as a Bernoulli trial (that is, a trial with an outcome of true or false) with a probability of success of 0.99, which we can model using a binomial distribution.

```
likelihood <- dbinom(nPositives, nTests, 0.99)
likelihood
```

```
## [1] 0.091
```

11.3.4 Computing the marginal likelihood

We also need to know the overall likelihood of the data – that is, finding 9 positives out of 10 tests. Computing the marginal likelihood is often one of the most difficult aspects of Bayesian analysis, but for our example it's simple because we can take advantage of the specific form of Bayes' theorem that we introduced in Section 3.7:

$$P(B|A) = \frac{P(A|B) * P(B)}{P(A|B) * P(B) + P(A|\neg B) * P(\neg B)}$$

The marginal likelihood in this case is a weighted average of the likelihood of the data under either presence or absence of the explosive, multiplied by the probability of the explosive being present (i.e. the prior). In this case, let's say that we know that the specificity of the test is 0.9, such that the likelihood of a positive result when there is no explosive is 0.1.

We can compute this in R as follows:

```
marginal_likelihood <-
  dbinom(
    x = nPositives,
    size = nTests,
    prob = 0.99
  ) * prior +
  dbinom(
    x = nPositives,
    size = nTests,
    prob = .1
  ) *
  (1 - prior)
```

```
sprintf("marginal likelihood = %.3e", marginal_likelihood)
```

```
## [1] "marginal likelihood = 9.094e-09"
```

11.3.5 Computing the posterior

We now have all of the parts that we need to compute the posterior probability of an explosive being present, given the observed 9 positive outcomes out of 10 tests.

```
posterior <- (likelihood * prior) / marginal_likelihood
posterior
```

```
## [1] 0.01
```

This result shows us that the probability of an explosive in the bag is much higher than the prior, but nowhere near certainty, again highlighting the fact that testing for rare events is almost always liable to produce high numbers of false positives.

11.4 Estimating posterior distributions

In the previous example there were only two possible outcomes – the explosive is either there or it's not – and we wanted to know which outcome was most likely given the data. However, in other cases we want to use Bayesian estimation to estimate the numeric value of a parameter. Let's say that we want to know about the effectiveness of a new drug for pain; to test this, we can administer the drug to a group of patients and then ask them whether their pain improved or not after taking the drug. We can use Bayesian analysis to estimate the proportion of people for whom the drug will be effective using these data.

11.4.1 Specifying the prior

In this case, we don't have any prior information about the effectiveness of the drug, so we will use a *uniform distribution* as our prior, since all values are equally likely under a uniform distribution. In order to simplify the example, we will only look at a subset of 99 possible values of effectiveness (from .01 to .99, in steps of .01). Therefore, each possible value has a prior probability of 1/99.

11.4.2 Collect some data

We need some data in order to estimate the effect of the drug. Let's say that we administer the drug to 100 individuals, and get the following results:

```
# create a table with results
nResponders <- 64
nTested <- 100

drugDf <- tibble(
  outcome = c("improved", "not improved"),
  number = c(nResponders, nTested - nResponders)
)
pander(drugDf)
```

outcome	number
improved	64

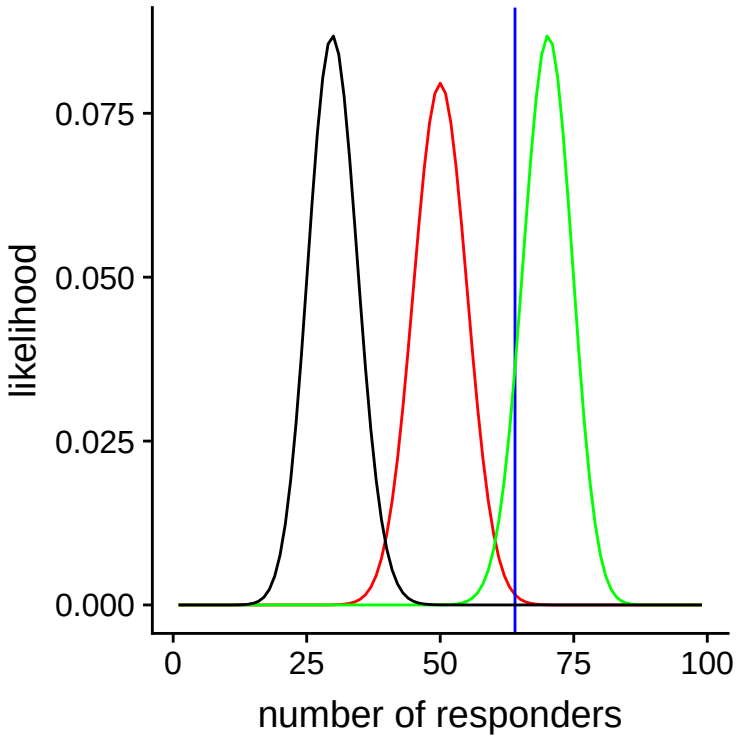

Figure 11.1: Likelihood of each possible number of responders under several different hypotheses (p(respond)=0.5 (red), 0.7 (green), 0.3 (black). Observed value shown in blue.

outcome	number
not improved	36

11.4.3 Computing the likelihood

We can compute the likelihood of the data under any particular value of the effectiveness parameter using the `dbinom()` function in R. In Figure 11.1 you can see the likelihood curves over numbers of responders for several different values of $p_{respond}$. Looking at this, it seems that our observed data are relatively more likely under the hypothesis of $p_{respond} = 0.7$, somewhat less likely under the hypothesis of $p_{respond} = 0.5$, and quite unlikely under the hypothesis of $p_{respond} = 0.3$. One of the fundamental ideas of Bayesian inference is that we will try to find the value of our parameter of interest that makes the data most likely, while also taking into account our prior knowledge.

11.4.4 Computing the marginal likelihood

In addition to the likelihood of the data under different hypotheses, we need to know the overall likelihood of the data, combining across all hypotheses (i.e., the marginal likelihood). This marginal likelihood is primarily important beacuse it helps to ensure that the posterior values are true probabilities. In this case, our use of a set of discrete possible parameter values makes it easy to compute the marginal likelihood, because we can just compute the likelihood of each parameter value under each hypothesis and add them up.

```
# compute marginal likelihood
likeDf <-
  likeDf %>%
```

```
  mutate(uniform_prior = array(1 / n()))

# multiply each likelihood by prior and add them up
marginal_likelihood <-
  sum(
    dbinom(
      x = nResponders, # the number who responded to the drug
      size = 100, # the number tested
      likeDf$presp # the likelihood of each response
    ) * likeDf$uniform_prior
  )

sprintf("marginal likelihood = %0.4f", marginal_likelihood)
```

```
## [1] "marginal likelihood = 0.0100"
```

11.4.5 Computing the posterior

We now have all of the parts that we need to compute the posterior probability distribution across all possible values of $p_{respond}$, as shown in Figure 11.2.

```
# Create data for use in figure
bayesDf <-
  tibble(
    steps = seq(from = 0.01, to = 0.99, by = 0.01)
  ) %>%
  mutate(
    likelihoods = dbinom(
      x = nResponders,
      size = 100,
      prob = steps
    ),
    priors = dunif(steps) / length(steps),
    posteriors = (likelihoods * priors) / marginal_likelihood
  )
```

11.4.6 Maximum a posteriori (MAP) estimation

Given our data we would like to obtain an estimate of $p_{respond}$ for our sample. One way to do this is to find the value of $p_{respond}$ for which the posterior probability is the highest, which we refer to as the *maximum a posteriori* (MAP) estimate. We can find this from the data in 11.2:

```
# compute MAP estimate
MAP_estimate <-
  bayesDf %>%
  arrange(desc(posteriors)) %>%
  slice(1) %>%
  pull(steps)

sprintf("MAP estimate = %0.4f", MAP_estimate)
```

```
## [1] "MAP estimate = 0.6400"
```

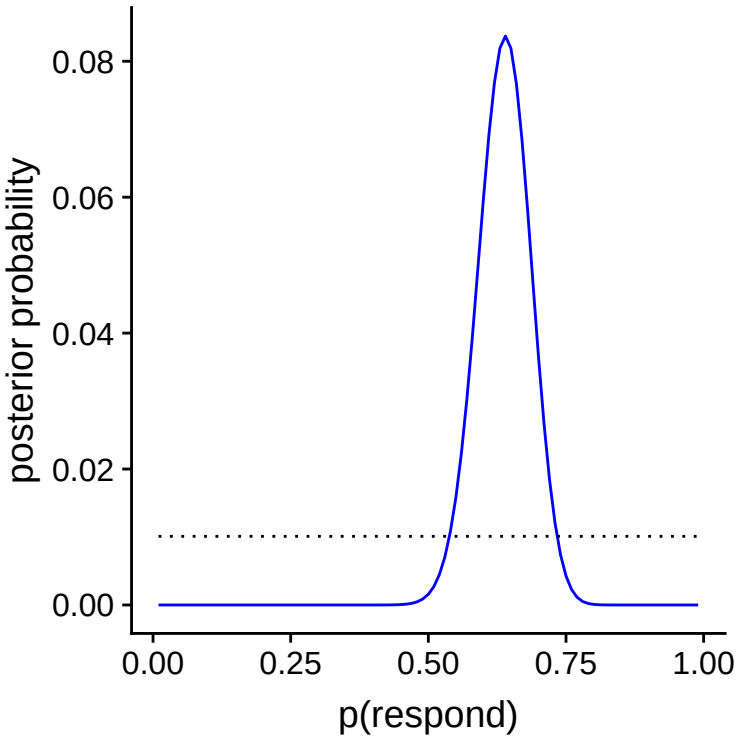

Figure 11.2: Posterior probability distribution plotted in blue against uniform prior distribution (dotted black line).

Note that this is simply the proportion of responders from our sample – this occurs because the prior was uniform and thus didn't bias our response.

11.4.7 Credible intervals

Often we would like to know not just a single estimate for the posterior, but an interval in which we are confident that the posterior falls. We previously discussed the concept of confidence intervals in the context of frequentist inference, and you may remember that the interpretation of confidence intervals was particularly convoluted. What we really want is an interval in which we are confident that the true parameter falls, and Bayesian statistics can give us such an interval, which we call a *credible interval*.

In some cases the credible interval can be computed *numerically* based on a known distribution, but it's more common to generate a credible interval by sampling from the posterior distribution and then to compute quantiles of the samples. This is particularly useful when we don't have an easy way to express the posterior distribution numerically, which is often the case in real Bayesian data analysis.

We will generate samples from our posterior distribution using a simple algorithm known as *rejection sampling*. The idea is that we choose a random value of x (in this case $p_{respond}$) and a random value of y (in this case, the posterior probability of $p_{respond}$) each from a uniform distribution. We then only accept the sample if $y < f(x)$ - in this case, if the randomly selected value of y is less than the actual posterior probability of y. Figure 11.3 shows an example of a histogram of samples using rejection sampling, along with the 95% credible intervals obtained using this method.

```
# Compute credible intervals for example

nsamples <- 100000
```

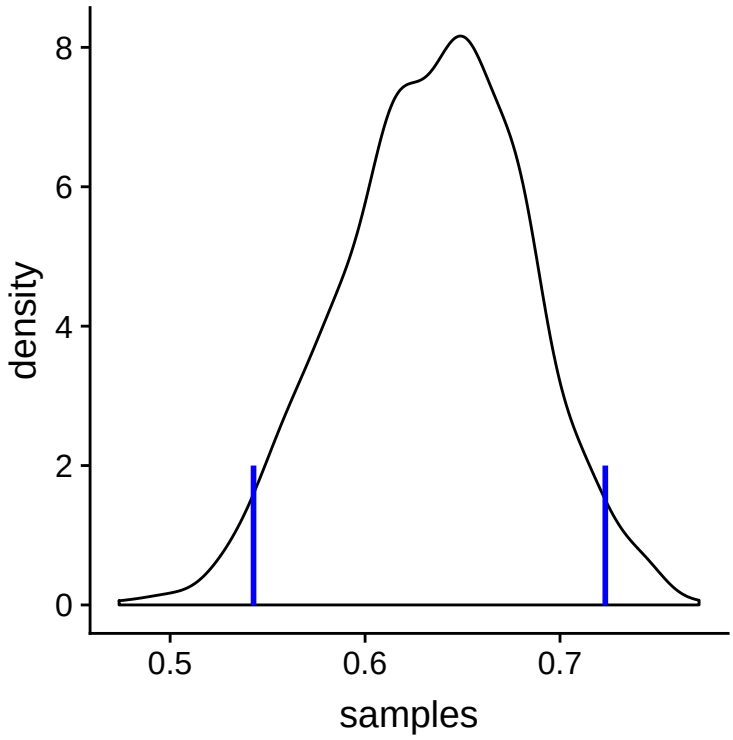

Figure 11.3: Rejection sampling example.The black line shows the density of all possible values of p(respond); the blue lines show the 2.5th and 97.5th percentiles of the distribution, which represent the 95 percent credible interval for the estimate of p(respond).

```
# create random uniform variates for x and y
x <- runif(nsamples)
y <- runif(nsamples)

# create f(x)
fx <- dbinom(x = nResponders, size = 100, prob = x)

# accept samples where y < f(x)
accept <- which(y < fx)
accepted_samples <- x[accept]

credible_interval <- quantile(x = accepted_samples, probs = c(0.025, 0.975))
pander(credible_interval)
```

2.5%	98%
0.54	0.72

The interpretation of this credible interval is much closer to what we had hoped we could get from a confidence interval (but could not): It tells us that there is a 95% probability that the value of $p_{respond}$ falls between these two values. Importantly, it shows that we have high confidence that $p_{respond} > 0.0$, meaning that the drug seems to have a positive effect.

11.4.8 Effects of different priors

In the previous example we used a *flat prior*, meaning that we didn't have any reason to believe that any particular value of $p_{respond}$ was more or less likely. However, let's say that we had instead started with some previous data: In a previous study, researchers had tested 20 people and found that 10 of them had responded positively. This would have lead us to start with a prior belief that the treatment has an effect in 50% of people. We can do the same computation as above, but using the information from our previous study to inform our prior (see Figure 11.4).

```
# compute likelihoods for data under all values of p(heads)
# using a flat or empirical prior.
# here we use the quantized values from .01 to .99 in steps of 0.01

df <-
  tibble(
    steps = seq(from = 0.01, to = 0.99, by = 0.01)
  ) %>%
  mutate(
    likelihoods = dbinom(nResponders, 100, steps),
    priors_flat = dunif(steps) / sum(dunif(steps)),
    priors_empirical = dbinom(10, 20, steps) / sum(dbinom(10, 20, steps))
  )

marginal_likelihood_flat <-
  sum(dbinom(nResponders, 100, df$steps) * df$priors_flat)

marginal_likelihood_empirical <-
  sum(dbinom(nResponders, 100, df$steps) * df$priors_empirical)

df <-
  df %>%
  mutate(
    posteriors_flat =
      (likelihoods * priors_flat) / marginal_likelihood_flat,
    posteriors_empirical =
      (likelihoods * priors_empirical) / marginal_likelihood_empirical
  )
```

Note that the likelihood and marginal likelihood did not change - only the prior changed. The effect of the change in prior to was to pull the posterior closer to the mass of the new prior, which is centered at 0.5.

Now let's see what happens if we come to the analysis with an even stronger prior belief. Let's say that instead of having previously observed 10 responders out of 20 people, the prior study had instead tested 500 people and found 250 responders. This should in principle give us a much stronger prior, and as we see in Figure 11.5, that's what happens: The prior is much more concentrated around 0.5, and the posterior is also much closer to the prior. The general idea is that Bayesian inference combines the information from the prior and the likelihood, weighting the relative strength of each.

```
# compute likelihoods for data under all values of p(heads) using strong prior.

df <-
  df %>%
  mutate(
    priors_strong = dbinom(250, 500, steps) / sum(dbinom(250, 500, steps))
  )
```

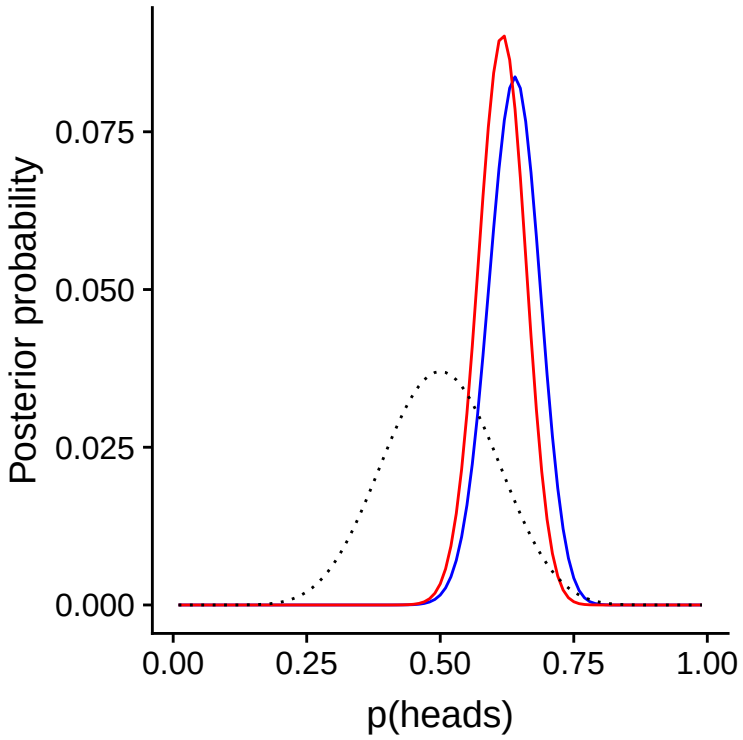

Figure 11.4: Effects of priors on the posterior distribution. The original posterior distribution based on a flat prior is plotted in blue. The prior based on the observation of 10 responders out of 20 people is plotted in the dotted black line, and the posterior using this prior is plotted in red.

```
marginal_likelihood_strong <-
  sum(dbinom(nResponders, 100, df$steps) * df$priors_strong)

df <-
  df %>%
  mutate(
    posteriors_strongprior = (likelihoods * priors_strong) / marginal_likelihood_strong
  )
```

This example also highlights the sequential nature of Bayesian analysis – the posterior from one analysis can become the prior for the next analysis.

Finally, it is important to realize that if the priors are strong enough, they can completely overwhelm the data. Let's say that you have an absolute prior that $p_{respond}$ is 0.8 or greater, such that you set the prior likelihood of all other values to zero. What happens if we then compute the posterior?

```
# compute likelihoods for data under all values of p(respond) using absolute prior.
df <-
  df %>%
  mutate(
    priors_absolute = array(data = 0, dim = length(steps)),
    priors_absolute = if_else(
      steps >= 0.8,
      1, priors_absolute
    ),
    priors_absolute = priors_absolute / sum(priors_absolute)
```

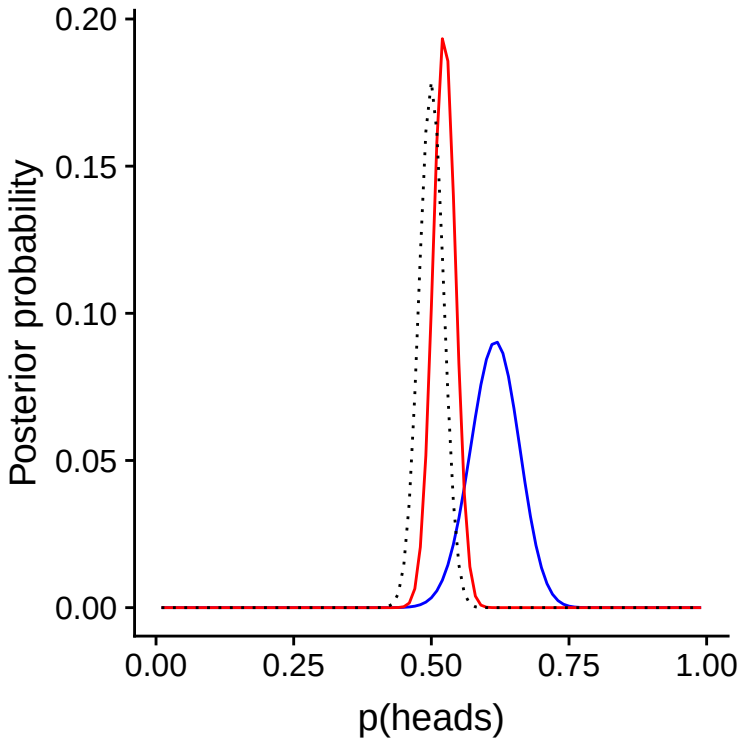

Figure 11.5: Effects of the strength of the prior on the posterior distribution. The blue line shows the posterior obtained using the prior based on 50 heads out of 100 people. The dotted black line shows the prior based on 250 heads out of 500 flips, and the red line shows the posterior based on that prior.

```
  )

marginal_likelihood_absolute <-
  sum(dbinom(nResponders, 100, df$steps) * df$priors_absolute)

df <-
  df %>%
  mutate(
    posteriors_absolute =
      (likelihoods * priors_absolute) / marginal_likelihood_absolute
  )
```

In Figure 11.6 we see that there is zero density in the posterior for any of the values where the prior was set to zero - the data are overwhelmed by the absolute prior.

11.5 Choosing a prior

The impact of priors on the resulting inferences are the most controversial aspect of Bayesian statistics. There are various ways to choose one's priors, which (as we saw above) can impact the resulting inferences. *Uninformative priors* attempt to bias the resulting posterior as little as possible, as we saw in the example of the uniform prior above. It's also common to use *weakly informative priors* (or *default priors*), which bias the result only very slightly. For example, if we had used a binomial distribution based on one heads out of two coin flips, the prior would have been centered around 0.5 but fairly flat, biasing the posterior only slightly.

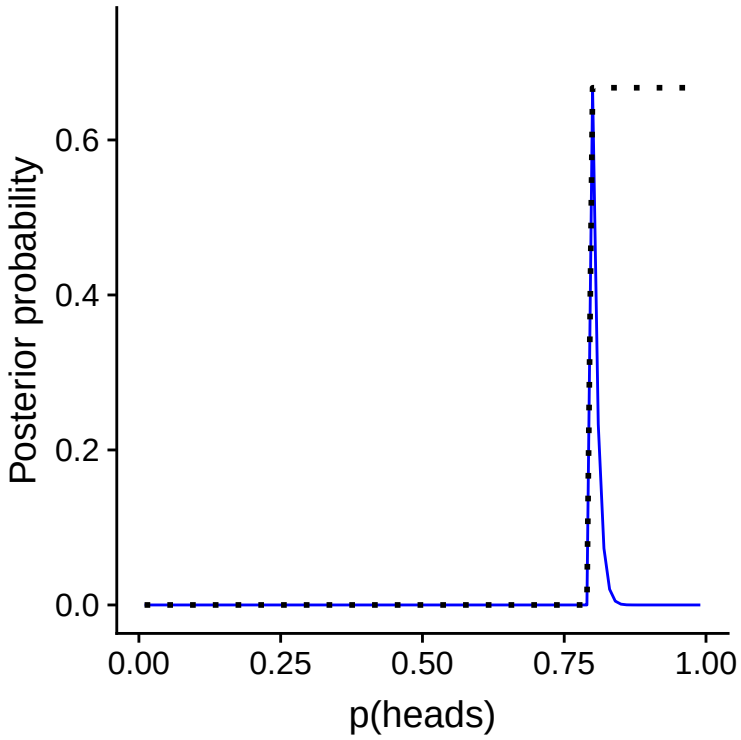

Figure 11.6: Effects of the strength of the prior on the posterior distribution. The blue line shows the posterior obtained using an absolute prior which states that p(respond) is 0.8 or greater. The prior is shown in the dotted black line.

It is also possible to use priors based on the scientific literature or pre-existing data, which we would call *empirical priors*. In general, however, we will stick to the use of uninformative/weakly informative priors, since they raise the least concern about biasing our results. In general it is a good idea to try any Bayesian analysis using multiple reasonable priors, and make sure that the results don't change in an important way based on the prior.

11.6 Bayesian hypothesis testing

Having learned how to perform Bayesian estimation, we now turn to the use of Bayesian methods for hypothesis testing. Let's say that there are two politicians who differ in their beliefs about whether the public supports the death penalty. Senator Smith thinks that only 40% of people support the death penalty, whereas Senator Jones thinks that 60% of people support the death penalty. They arrange to have a poll done to test this, which asks 1000 randomly selected people whether they support the death penalty. The results are that 490 of the people in the polled sample support the death penalty. Based on these data, we would like to know: Do the data support the claims of one senator over the other? We can test this using a concept known as the Bayes factor.

11.6.1 Bayes factors

The Bayes factor characterizes the relative likelihood of the data under two different hypotheses. It is defined as:

$$BF = \frac{p(data|H_1)}{p(data|H_2)}$$

for two hypotheses H_1 and H_2. In the case of our two senators, we know how to compute the likelihood of the data under each hypothesis using the binomial distribution. We will put Senator Smith in the numerator and Senator Jones in the denominator, so that a value greater than one will reflect greater evidence for Senator Smith, and a value less than one will reflect greater evidence for Senator Jones.

```
# compute Bayes factor for Smith vs. Jones

bf <-
  dbinom(
    x = 490,
    size = 1000,
    prob = 0.4 #Smith's hypothesis
  ) / dbinom(
    x = 490,
    size = 1000,
    prob = 0.6 #Jones' hypothesis
  )

sprintf("Bayes factor = %0.2f", bf)
```

```
## [1] "Bayes factor = 3325.26"
```

This number provides a measure of the evidence that the data provides regarding the two hypotheses - in this case, it tells us the data support Senator Smith more than 3000 times more strongly than they support Senator Jones.

11.6.2 Bayes factors for statistical hypotheses

In the previous example we had specific predictions from each senator, whose likelihood we could quantify using the binomial distribution. However, in real data analysis we generally must deal with uncertainty about our parameters, which complicates the Bayes factor. However, in exchange we gain the ability to quantify the relative amount of evidence in favor of the null versus alternative hypotheses.

Let's say that we are a medical researcher performing a clinical trial for the treatment of diabetes, and we wish to know whether a particular drug reduces blood glucose compared to placebo. We recruit a set of volunteers and randomly assign them to either drug or placebo group, and we measure the change in hemoglobin A1C (a marker for blood glucose levels) in each group over the period in which the drug or placebo was administered. What we want to know is: Is there a difference between the drug and placebo?

First, let's generate some data and analyze them using null hypothesis testing (see Figure 11.7).

```
# create simulated data for drug trial example

set.seed(123456)
nsubs <- 40
effect_size <- 0.1

# randomize indiviuals to drug (1) or placebo (0)
drugDf <-
  tibble(
    group = as.integer(runif(nsubs) > 0.5)
  ) %>%
  mutate(
```

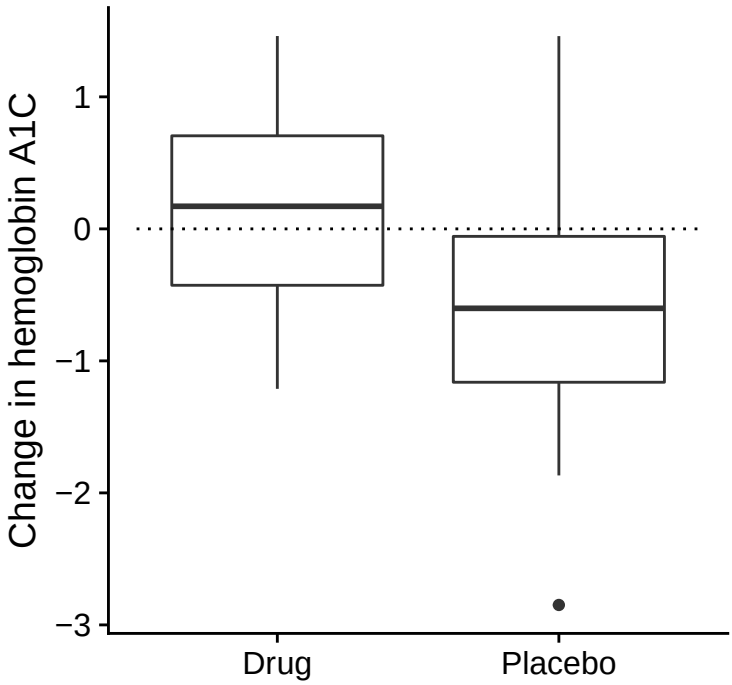

Figure 11.7: Box plots showing data for drug and placebo groups.

```
    hbchange = rnorm(nsubs) - group * effect_size
  )
```

Let's perform an independent-samples t-test, which shows that there is a significant difference between the groups:

```
# compute t-test for drug example
drugTT <- t.test(hbchange ~ group, alternative = "greater", data = drugDf)
print(drugTT)
```

```
##
##  Welch Two Sample t-test
##
## data:  hbchange by group
## t = 2, df = 40, p-value = 0.03
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.096   Inf
## sample estimates:
## mean in group 0 mean in group 1
##            0.12           -0.48
```

This test tells us that there is a significant difference between the groups, but it doesn't quantify how strongly the evidence supports the null versus alternative hypotheses. To measure that, we can compute a Bayes factor using `ttestBF` function from the BayesFactor package in R:

```
# compute Bayes factor for drug data
bf_drug <- ttestBF(
  formula = hbchange ~ group, data = drugDf,
  nullInterval = c(0, Inf)
)
```

```
bf_drug
```

```
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 0<d<Inf    : 2.4  ±0%
## [2] Alt., r=0.707 !(0<d<Inf) : 0.12 ±0%
##
## Against denominator:
##   Null, mu1-mu2 = 0
## ---
## Bayes factor type: BFindepSample, JZS
```

The Bayes factor here tells us that the alternative hypothesis (i.e. that the difference is greater than zero) is about 2.4 times more likely than the point null hypothesis (i.e. a mean difference of exactly zero) given the data.

11.6.2.1 One-sided tests

We generally are less interested in testing against the null hypothesis of a specific point value (e.g. mean difference = 0) than we are in testing against a directional null hypothesis (e.g. that the difference is less than or equal to zero). We can also perform a directional (or *one-sided*) test using the results from `ttestBF` analysis, since it provides two Bayes factors: one for the alternative hypothesis that the mean difference is greater than zero, and one for the alternative hypothesis that the mean difference is less than zero. If we want to assess the relative evidence for a positive effect, we can compute a Bayes factor comparing the relative evidence for a positive versus a negative effect by simply dividing the two Bayes factors returned by the function:

```
bf_drug[1]/bf_drug[2]
```

```
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 0<d<Inf : 20 ±0%
##
## Against denominator:
##   Alternative, r = 0.707106781186548, mu =/= 0 !(0<d<Inf)
## ---
## Bayes factor type: BFindepSample, JZS
```

Now we see that the Bayes factor for a positive effect versus a negative effect is substantially larger (almost 20).

11.6.2.2 Interpreting Bayes Factors

How do we know whether a Bayes factor of 2 or 20 is good or bad? There is a general guideline for interpretation of Bayes factors suggested by Kass & Rafferty (1995):

BF	Strength of evidence
1 to 3	not worth more than a bare mention
3 to 20	positive
20 to 150	strong
>150	very strong

Based on this, even though the statisical result is significant, the amount of evidence in favor of the alternative vs. the point null hypothesis is weak enough that it's not worth even mentioning, whereas the evidence for the directional hypothesis is positive, but not quite strong.

11.6.3 Assessing evidence for the null hypothesis

Because the Bayes factor is comparing evidence for two hypotheses, it also allows us to assess whether there is evidence in favor of the null hypothesis, which we couldn't do with standard null hypothesis testing (because it starts with the assumption that the null is true). This can be very useful for determining whether a non-significant result really provides strong evidence that there is no effect, or instead just reflects weak evidence overall.

11.7 Suggested readings

- *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy*, by Sharon Bertsch McGrayne
- *Doing Bayesian Data Analysis: A Tutorial Introduction with R*, by John K. Kruschke