Data Analysis
Week 7: Confidence Intervals

1 Introduction

In previous weeks we have seen many examples of calculating *sample statistics* such as means, percentiles, standard deviations and regression coefficients. These *sample statistics* are used as *point estimates* of *population parameters* which describe the *population* from which the *sample* of data was taken. That last sentence assumes you're familiar with concepts and terminology about sampling (e.g. from the *Statistical Inference* course in 1st Semester) so here is a summary of some key terms:

1. **Population**: The population is a set of N observations of interest.
2. **Population parameter**: A population parameter is a numerical summary value about the population. In most settings, this is a value that's unknown and you wish you knew it.
3. **Census**: An exhaustive enumeration/counting of all observations in the population in order to compute the population parameter's numerical value *exactly*.
    - When N is small, a census is feasible. However, when N is large, a census can get very expensive, either in terms of time, energy, or money.
4. **Sampling**: Collecting a sample of size n of observations from the population. Typically the sample size n is much smaller than the population size N, thereby making sampling a much cheaper procedure than a census.
    - It is important to remember that the lowercase n corresponds to the sample size and uppercase N corresponds to the population size, thus $n \leq N$.
5. **Point estimates/sample statistics**: A summary statistic based on the sample of size n that *estimates* the unknown population parameter.
6. **Representative sampling**: A sample is said be a *representative sample* if it "looks like the population". In other words, the sample's characteristics are a good representation of the population's characteristics.
7. **Generalisability**: We say a sample is *generalisable* if any results based on the sample can generalise to the population.
8. **Bias**: In a statistical sense, we say *bias* occurs if certain observations in a population have a higher chance of being sampled than others. We say a sampling procedure is *unbiased* if every observation in a population had an equal chance of being sampled.
9. **Random sampling**: We say a sampling procedure is *random* if we sample randomly from the population in an unbiased fashion.

1.1 Inference via sampling

The logic of inference via sampling is:

- If the sampling of a sample of size n is done at **random**, then
- The sample is **unbiased** and **representative** of the population, thus
- Any result based on the sample can **generalise** to the population, thus
- The **point estimate/sample statistic** is an *estimate* of the unknown population parameter of interest

and thus we have **inferred** something about the population based on our sample.

Task: To ground the above concepts, consider the following:

In 2013 National Public Radio in the USA reported a poll of President Obama's approval rating among young Americans aged 18-29 in an article Poll: Support For Obama Among Young Americans Eroding. A quote from the article:

> After voting for him in large numbers in 2008 and 2012, young Americans are souring on President Obama.
>
> According to a new Harvard University Institute of Politics poll, just 41 percent of millennials (adults ages 18-29) approve of Obama's job performance, his lowest-ever standing among the group and an 11-point drop from April.

Identify each of the following terms 1-9 above in this context. The solution is given below.

Solution:

1. **Population**: Who is the population of N observations of interest?
   - Obama poll: $N = ?$ young Americans aged 18-29
2. **Population parameter**: What is the population parameter?
   - Obama poll: The true population proportion p of young Americans who approve of Obama's job performance.
3. **Census**: What would a census be in this case?
   - Obama poll: Locating all $N = ?$ young Americans (which is in the millions) and asking them if they approve of Obama's job performance. This would be quite expensive to do!
4. **Sampling**: How do you acquire the sample of size n observations?
   - Obama poll: One way would be to get phone records from a database and pick out n phone numbers. In the case of the above poll, the sample was of size $n = 2089$ young adults.
5. **Point estimates/sample statistics**: What is the summary statistic based on the sample of size n that *estimates* the unknown population parameter?
   - Key: The sample proportion $\widehat{p}$ of young Americans in the sample of size $n = 2089$ that approve of Obama's job performance. In this study's case, $\widehat{p} = 0.41$ which is the quoted 41% figure in the article.
6. **Representative sampling**: Is the sample procedure *representative*? In other words, do the resulting samples "look like" the population?
   - Obama poll: Does our sample of $n = 2089$ young Americans "look like" the population of all young Americans aged 18-29?
7. **Generalisability**: Are the samples *generalisable* to the greater population?
   - Obama poll: Is $\widehat{p} = 0.41$ a "good guess" of p? In other words, can we confidently say that 41% of *all* young Americans approve of Obama.
8. **Bias**: Is the sampling procedure unbiased? In other words, do all observations have an equal chance of being included in the sample?
   - Obama poll: Did all young Americans have an equal chance at being represented in this poll? For example, if this was conducted using a database of only mobile phone numbers, would people without mobile phones be included? What about if this were an internet poll on a certain news website? Would non-readers of this this website be included?
9. **Random sampling**: Was the sampling random?
   - Obama poll: Random sampling is a necessary assumption for all of the above to work. Most articles reporting on polls take this assumption as granted. In our Obama poll, you'd have to ask the group that conducted the poll: The Harvard University Institute of Politics.

Following "the logic of inference via sampling" above, in the Obama poll example:

- If we had a way of contacting a randomly chosen sample of 2089 young Americans and poll their approval of Obama, then
- These 2089 young Americans would "look like" the population of all young Americans, thus
- Any results based on this sample of 2089 young Americans can generalise to the entire population of all young Americans, thus
- The reported sample approval rating of 41% of these 2089 young Americans is an *estimate* of the true approval rating amongst *all* young Americans.

So this poll's *estimate* of Obama's approval rating amongst millennials was 41%. However is this the end of the story when understanding the results of a poll? If you read further in the article, it states:

Note the term *margin of error*, which here is plus or minus 2.1 percentage points. This is saying that a typical range of errors for polls of this type is about ±2.1%. These errors are caused by *sampling variation*, i.e. the fact that sample statistics vary from sample to sample.

When speaking about estimating population parameters using sample statistics the term "error" can be misleading. Any variation from the true population parameter value is called "error". It doesn't mean a mistake has been made, it's just acknowledging the fact that an estimate based on a sample is highly likely to be different from the true population parameter it is estimating. A reasonable range of "errors" to expect is called the "margin of error". We'll see this week that this is what's known as a 95% confidence interval (CI) for the unknown approval rating. We'll study confidence intervals (CIs) using a new package for our data science and statistical toolbox: the `infer` package for statistical inference.

Required R packages

Before we proceed, load all the packages needed for this week:

```
library(dplyr)
library(ggplot2)
library(janitor)
library(moderndive)
library(infer)
```

2 Inference using sample statistics

The table below lists a variety of contexts where sample statistics can be used to estimate population parameters. In all 6 cases, the point estimate/sample statistic *estimates* the unknown population parameter. It does so by computing summary statistics based on a sample of size n. We'll cover the first four scenarios this week and next week we'll cover Scenarios 5 & 6 about the regression line.

Table 1: Scenarios of sample statistics for inference.

Scenario	Population Parameter	Population Notation	Sample Statistic	Sample Notation
1	Population proportion	p	Sample proportion	$\widehat{p}$
2	Population mean	μ	Sample mean	$\bar{x}$
3	Diff.in pop. props	$p_1 - p_2$	Diff. in sample props	$\widehat{p}_1 - \widehat{p}_2$
4	Diff. in pop. means	$\mu_1 - \mu_2$	Diff. in sample means	$\bar{x}_1 - \bar{x}_2$
5	Pop. intercept	β_0	Sample intercept	$\widehat{\beta}_0$ or b_0
6	Pop. slope	β_1	Sample slope	$\widehat{\beta}_1$ or b_1

In reality, we don't have access to the population parameter values (if we did, why would we need to estimate them?) we only have a single sample of data from a larger population. We'd like to be able to make some reasonable guesses about population parameters using that single sample to create a range of plausible values for a population parameter. This range of plausible values is known as a **confidence interval**.

There are theoretical ways of defining confidence intervals for these different scenarios (such as you saw in 'Statistical Inference' in Semester 1). But we can also use a single sample to get some idea of how other samples might vary in terms of their sample statistics, i.e. to estimate the sampling distributions of sample statistics. One common way this is done is via a process known as **bootstrapping**.

3 Bootstrapping

The `moderndive` package contains a sample of 40 pennies collected and minted in the United States. Let's explore this sample data first:

```
orig_pennies_sample
```

```
# A tibble: 40 x 2
    year age_in_2011
   <int>       <int>
 1  2005           6
 2  1981          30
 3  1977          34
 4  1992          19
 5  2005           6
 6  2006           5
 7  2000          11
 8  1992          19
 9  1988          23
10  1996          15
# ... with 30 more rows
```

The `orig_pennies_sample` data frame has rows corresponding to a single penny with two variables:

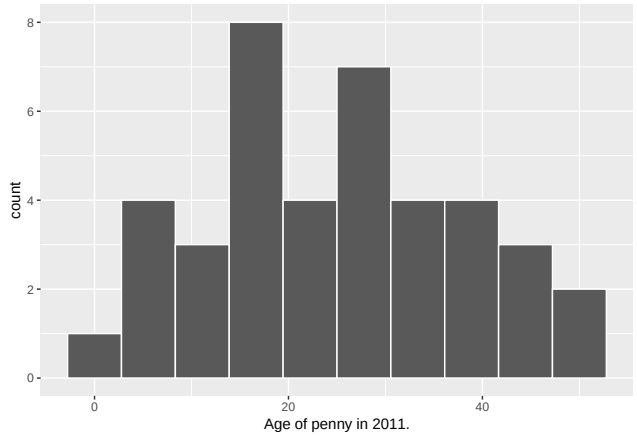
- `year` of minting as shown on the penny and
- `age_in_2011` giving the years the penny had been in circulation from 2011 as an integer, e.g. 15, 2, etc.

Suppose we are interested in understanding some properties of the mean age of **all** US pennies from this data collected in 2011. How might we go about that? Let's begin by understanding some of the properties of `orig_pennies_sample` using data wrangling from Week 2 and data visualisation from Week 1.

3.1 Exploratory data analysis

First, let's visualise the values in this sample as a histogram:

```
ggplot(orig_pennies_sample, aes(x = age_in_2011)) +
  geom_histogram(bins = 10, color = "white") +
  labs(x = "Age of penny in 2011.")
```



We see a roughly symmetric distribution here that has quite a few values near 20 years in age with only a few larger than 40 years or smaller than 5 years. If `orig_pennies_sample` is a representative sample from

the population, we'd expect the age of all US pennies collected in 2011 to have a similar shape, a similar spread, and similar measures of central tendency like the mean.

So where does the mean value fall for this sample? This point will be known as our **point estimate** and provides us with a single number that could serve as the guess to what the true population mean age might be. Recall how to find this using the `dplyr` package:

```
x_bar <- orig_pennies_sample %>%
  summarize(stat = mean(age_in_2011))
```

```
# A tibble: 1 x 1
   stat
  <dbl>
1  25.1
```

We've denoted this *sample mean* as $\bar{x}$, which is the standard notation for denoting the mean of a sample. Our point estimate is, thus, $\bar{x} = 25.1$. Note that this is just one sample though providing just one sample mean to estimate the population mean. To construct a *confidence interval* (and to do any sort of *statistical inference* for that matter) we need to know about the **sampling distribution** of this sample mean, i.e. how would its values vary if many samples of the same size were drawn from the same population.

The process of **bootstrapping** allows us to use a single sample to generate many different samples that will act as our way of approximating a sampling distribution using a created **bootstrap distribution** instead. We will "pull ourselves up by our bootstraps" (as the saying goes in English, see here) using a single sample (`orig_pennies_sample`) to get an idea of the **sampling distribution**.

3.2 The Bootstrapping Process

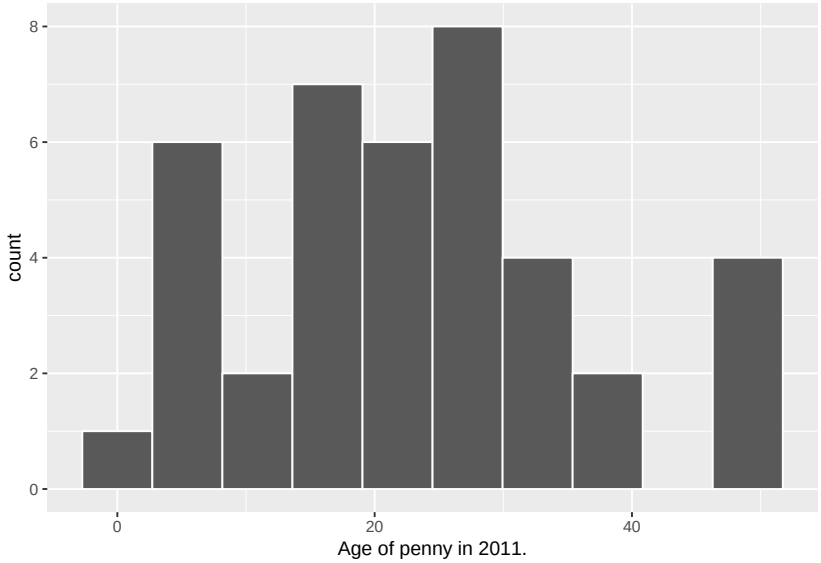
Bootstrapping uses a process of sampling **with replacement** from our original sample to create new **bootstrap samples** of the *same* size as our original sample. We can use the `rep_sample_n()` function in the `infer` package to explore what one such bootstrap sample would look like. Remember that we are randomly sampling from the original sample here **with replacement** and that we always use the same sample size for the bootstrap samples as the size of the original sample (`orig_pennies_sample`).

```
bootstrap_sample1 <- orig_pennies_sample %>%
  rep_sample_n(size = 40, replace = TRUE, reps = 1)
```

```
# A tibble: 40 x 3
# Groups:   replicate [1]
   replicate  year age_in_2011
       <int> <int>       <int>
 1         1  1983          28
 2         1  2000          11
 3         1  2004           7
 4         1  1981          30
 5         1  1993          18
 6         1  2006           5
 7         1  1981          30
 8         1  2004           7
 9         1  1992          19
10         1  1994          17
# ... with 30 more rows
```

Let's visualise what this new bootstrap sample looks like:

```
ggplot(bootstrap_sample1, aes(x = age_in_2011)) +
  geom_histogram(bins = 10, color = "white") +
  labs(x = "Age of penny in 2011.")
```



We now have another sample from what we could assume comes from the population of interest. We can similarly calculate the sample mean of this bootstrap sample, called a **bootstrap statistic**.

```
bootstrap_sample1 %>%
  summarize(stat = mean(age_in_2011))
```

```
# A tibble: 1 x 2
  replicate  stat
      <int> <dbl>
1         1  23.2
```

We can see that this sample mean is different to the `x_bar` value we calculated earlier for the `orig_pennies_sample` data. We'll come back to analysing the different bootstrap statistic values shortly.

Let's recap what was done to get to this bootstrap sample using a tactile explanation:

1. First, pretend that each of the 40 values of `age_in_2011` in `orig_pennies_sample` were written on a small piece of paper. Recall that these values were 6, 30, 34, 19, 6, etc.
2. Now, put the 40 small pieces of paper into a receptacle such as a baseball cap.
3. Shake up the pieces of paper.
4. Draw "at random" from the cap to select one piece of paper.
5. Write down the value on this piece of paper. Say that it is 28.
6. Now, place this piece of paper containing 28 back into the cap.
7. Draw "at random" again from the cap to select a piece of paper. Note that this is the *sampling with replacement* part since you may draw 28 again.
8. Repeat this process until you have drawn 40 pieces of paper and written down the values on these 40 pieces of paper. Completing this repetition produces ONE bootstrap sample.

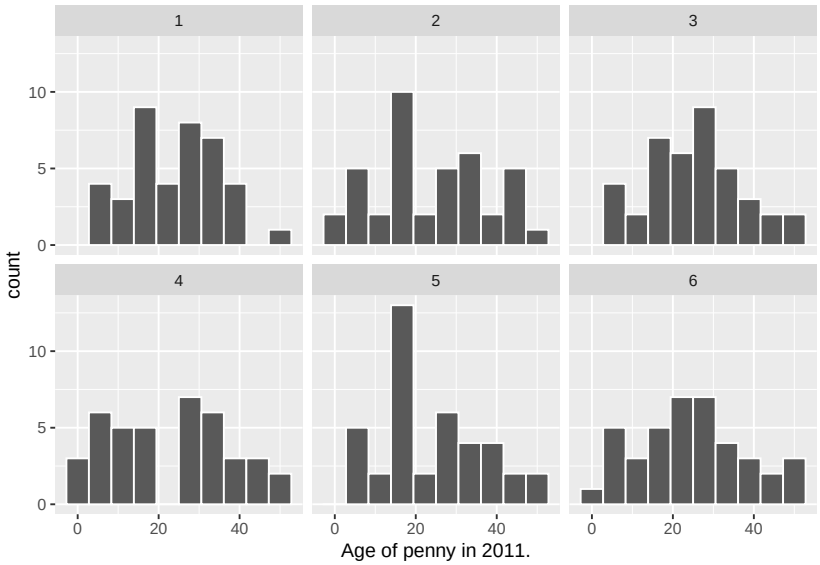
If you look at the values in `bootstrap_sample1`, you can see how this process plays out. We originally drew 28, then we drew 11, then 7, and so on. Of course, we didn't actually use pieces of paper and a cap here. We just had the computer perform this process for us to produce `bootstrap_sample1` using `rep_sample_n()` with `replace = TRUE` set.

The process of *sampling with replacement* is how we can use the original sample to take a guess as to what other values in the population may be. Sometimes in these bootstrap samples, we will select lots of larger values from the original sample, sometimes we will select lots of smaller values, and most frequently we will select values that are near the center of the sample. Let's explore what the distribution of values of

age_in_2011 for six different bootstrap samples looks like to further understand this variability.

```
six_bootstrap_samples <- orig_pennies_sample %>%
  rep_sample_n(size = 40, replace = TRUE, reps = 6)
```

```
ggplot(six_bootstrap_samples, aes(x = age_in_2011)) +
  geom_histogram(bins = 10, color = "white") +
  facet_wrap(~ replicate) +
  labs(x = "Age of penny in 2011.")
```



We can also look at the six different means using dplyr syntax:

```
six_bootstrap_samples %>%
  group_by(replicate) %>%
  summarize(stat = mean(age_in_2011))
```

```
# A tibble: 6 x 2
  replicate  stat
      <int> <dbl>
1         1  23.6
2         2  24.1
3         3  25.2
4         4  23.1
5         5  24.0
6         6  24.7
```

Instead of doing this six times, we could do it 1000 times and then look at the distribution of stat across all 1000 of the replicates. This sets the stage for the infer R package (see documentation here or the "Cheat Sheet" on the Data Analysis Moodle page) that helps users perform statistical inference such as confidence intervals and hypothesis tests using verbs similar to what you've seen with dplyr. We'll walk through setting up each of the infer verbs for confidence intervals using this orig_pennies_sample example, while also explaining the purpose of the verbs in a general framework.

4 The infer package for statistical inference

The infer package makes great use of the "pipe" %>% to create a pipeline for statistical inference. The goal of the package is to provide a way for its users to explain the computational process of confidence intervals

and hypothesis tests using the code as a guide. The verbs build in order here, so you'll want to start with `specify` and then continue through the others as needed.

4.1 Specify variables

The `specify` function is used primarily to choose which variables will be the focus of the statistical inference. In addition, a setting of which variable will act as the `explanatory` and which acts as the `response` variable is done here. For proportion problems (i.e. Scenarios 1 & 3 in Table 1) we also specify which of the different levels we are calculating the proportion of (e.g. "females", "approval of Obama's job performance").

To begin to create a confidence interval for the population mean age of US pennies in 2011, we start by using `specify()` to choose which variable in our `orig_pennies_sample` data we'd like to work with. This can be done in one of two ways:

1. Using the `response` argument:

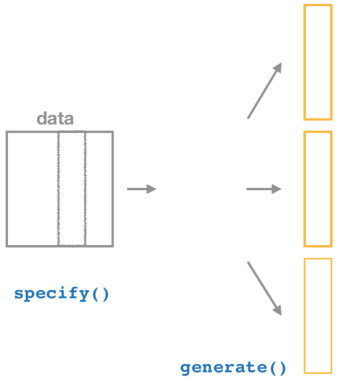
```
orig_pennies_sample %>%
  specify(response = age_in_2011)
```

2. Using `formula` notation:

```
orig_pennies_sample %>%
  specify(formula = age_in_2011 ~ NULL)
```

Note that the formula notation uses the common R methodology to include the response y variable on the left of the `~` and the explanatory x variable on the right of `~`. Recall that you used this notation frequently with the `lm` function in Weeks 4 and 6 when fitting regression models. Either notation works just fine, but a preference is usually given here for the `formula` notation to further build on the ideas from earlier chapters.

4.2 Generate replicates



After `specify`ing the variables we'd like in our inferential analysis, we next feed that into the `generate` verb. The `generate` verb's main argument is `reps`, which is used to give how many different repetitions one would like to perform. Another argument here is `type`, which is automatically determined by the kinds of variables passed into `specify`. We can also be explicit and set this `type` to be `type = "bootstrap"`. Make sure to check out `?generate` to see the options here and use the `?` operator to better understand other verbs as well.

Let's `generate` 1000 bootstrap samples:

```
thousand_bootstrap_samples <- orig_pennies_sample %>%
  specify(response = age_in_2011) %>%
  generate(reps = 1000)
```

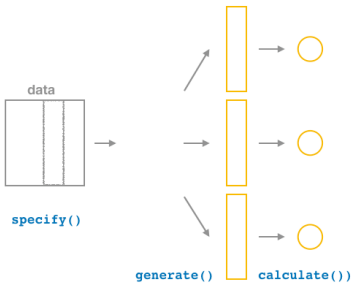
We can use the `dplyr` `count` function to help us understand what the `thousand_bootstrap_samples` data frame looks like:

```
thousand_bootstrap_samples %>%
  count(replicate)
```

```
# A tibble: 1,000 x 2
# Groups:   replicate [1,000]
   replicate     n
       <int> <int>
 1         1    40
 2         2    40
 3         3    40
 4         4    40
 5         5    40
 6         6    40
 7         7    40
 8         8    40
 9         9    40
10        10    40
# ... with 990 more rows
```

Notice that each `replicate` has 40 entries here. Now that we have 1000 different bootstrap samples, our next step is to `calculate` the bootstrap statistics for each sample.

4.3 Calculate summary statistics



After `generate`ing many different samples, we next want to condense those samples down into a single statistic for each `replicate`d sample. As seen in the diagram, the `calculate` function is helpful here.

As we did at the beginning of this chapter, we now want to calculate the mean `age_in_2011` for each bootstrap sample. To do so, we use the `stat` argument and set it to `"mean"` below. The `stat` argument has a variety of different options here and we will see further examples of this throughout the remaining chapters.

```
bootstrap_distribution <- orig_pennies_sample %>%
  specify(response = age_in_2011) %>%
  generate(reps = 1000) %>%
  calculate(stat = "mean")
```

```
# A tibble: 1,000 x 2
   replicate  stat
       <int> <dbl>
 1         1  26.5
 2         2  25.4
 3         3  26.0
 4         4  26
 5         5  25.2
 6         6  29.0
 7         7  22.8
 8         8  26.4
 9         9  24.9
10        10  28.1
# ... with 990 more rows
```

We see that the resulting data has 1000 rows and 2 columns corresponding to the 1000 replicates and the mean for each bootstrap sample.

Observed statistic / point estimate calculations

Just as `group_by() %>% summarize()` produces a useful workflow in `dplyr`, we can also use `specify() %>% calculate()` to compute summary measures on our original sample data. It's often helpful both in confidence interval calculations, but also in hypothesis testing to identify what the corresponding statistic is in the original data. For our example on penny age, we computed above a value of `x_bar` using the `summarize` verb in `dplyr`:

```
orig_pennies_sample %>%
  summarize(stat = mean(age_in_2011))
```

```
# A tibble: 1 x 1
   stat
  <dbl>
1  25.1
```

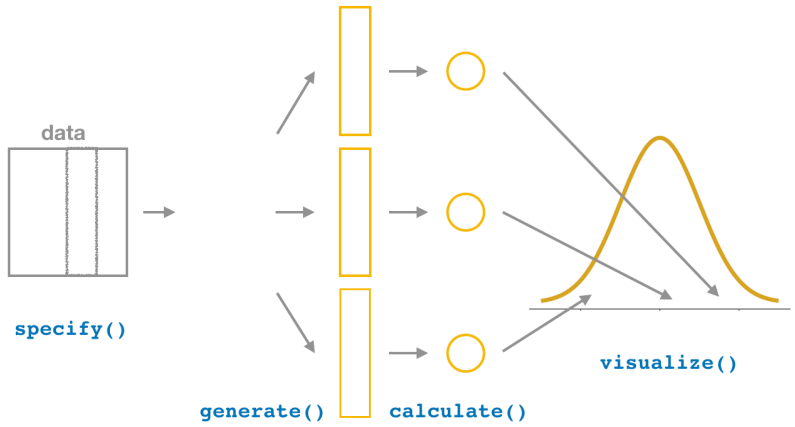
This can also be done by skipping the `generate` step in the pipeline feeding `specify` directly into `calculate`:

```
orig_pennies_sample %>%
  specify(response = age_in_2011) %>%
  calculate(stat = "mean")
```

```
# A tibble: 1 x 1
   stat
  <dbl>
1  25.1
```

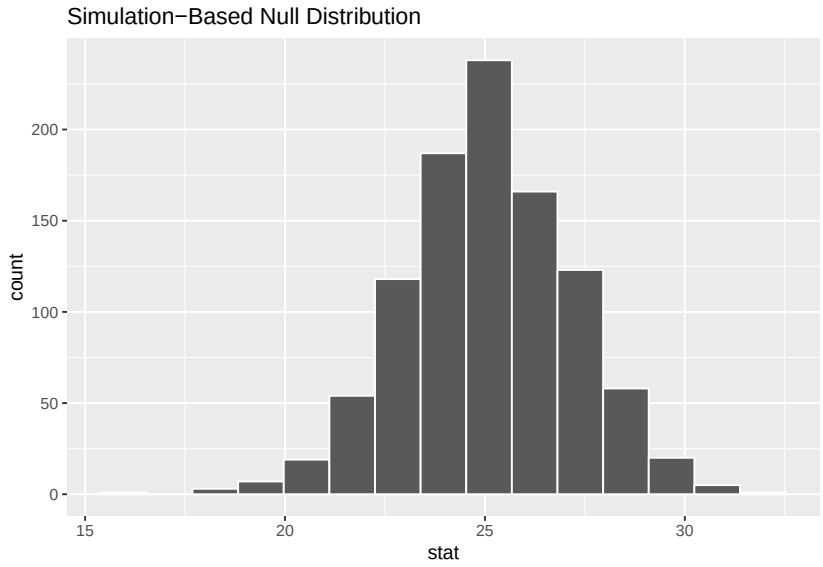
This shortcut will be particularly useful when the calculation of the observed statistic is tricky to do using `dplyr` alone. This is particularly the case when working with more than one variable.

4.4 Visualise the results



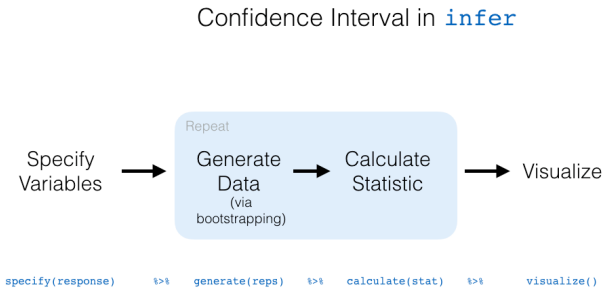
The `visualize` verb provides a simple way to view the bootstrap distribution as a histogram of the `stat` variable values. It has many other arguments that one can use as well including the shading of the histogram values corresponding to the confidence interval values.

```
bootstrap_distribution %>%
  visualize()
```



The shape of this resulting distribution may look familiar to you. It resembles the well-known normal (bell-shaped) curve.

The following diagram recaps the `infer` pipeline for creating a bootstrap distribution.



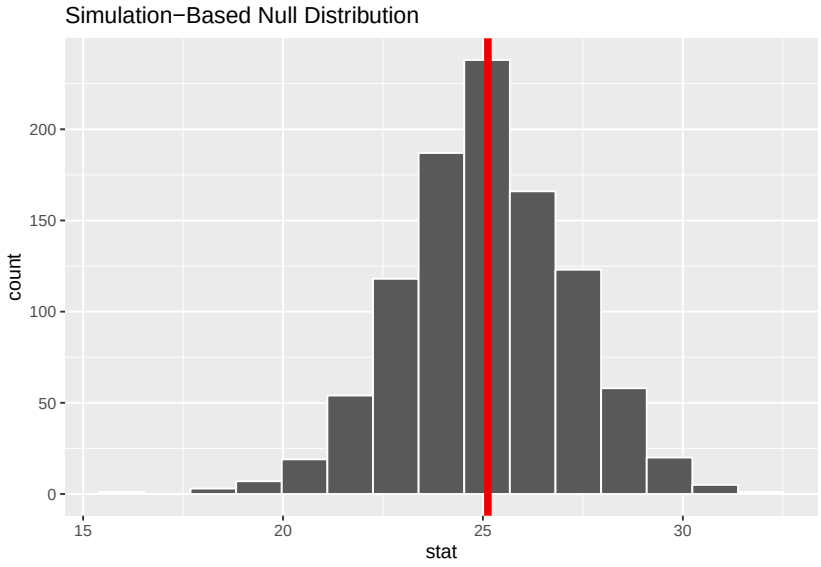
5 Constructing confidence intervals

A **confidence interval** gives a range of plausible values for a population parameter. It depends on a specified *confidence level* with higher confidence levels corresponding to wider confidence intervals and lower confidence levels corresponding to narrower confidence intervals. Common confidence levels include 90%, 95%, and 99%.

Confidence intervals are simple to define and play an important role in the sciences and any field that uses data. You can think of a confidence interval as playing the role of a net when fishing. Instead of just trying to catch a fish with a single spear (estimating an unknown parameter by using a single point estimate/sample statistic), we can use a net to try to provide a range of possible locations for the fish (use a range of possible values based around our sample statistic to make a plausible guess as to the location of the parameter).

The bootstrapping process will provide bootstrap statistics that have a bootstrap distribution with center at (or extremely close to) the mean of the original sample. This can be seen by giving the observed statistic `obs_stat` argument the value of the point estimate `x_bar`.

```
bootstrap_distribution %>%
  visualize(obs_stat = x_bar)
```



We can also compute the mean of the bootstrap distribution of means to see how it compares to `x_bar`:

```
bootstrap_distribution %>%
  summarize(mean_of_means = mean(stat))
```

```
# A tibble: 1 x 1
  mean_of_means
          <dbl>
1          25.1
```

In this case, we can see that the bootstrap distribution provides us a guess as to what the variability in different sample means may look like only using the original sample as our guide. We can quantify this variability in the form of a 95% confidence interval in two different ways.

5.1 The percentile method

One way to calculate a range of plausible values for the unknown mean age of coins in 2011 is to use the middle 95% of the `bootstrap_distribution` to determine our endpoints. Our endpoints are thus at the

2.5^{th} and 97.5^{th} percentiles. This can be done with `infer` using the `get_ci` function. (You can also use the `conf_int` or `get_confidence_interval` functions here as they are aliases that work the exact same way).

```
bootstrap_distribution %>%
  get_ci(level = 0.95, type = "percentile")
```

```
# A tibble: 1 x 2
  `2.5%` `97.5%`
   <dbl>   <dbl>
1   21.0    29.3
```

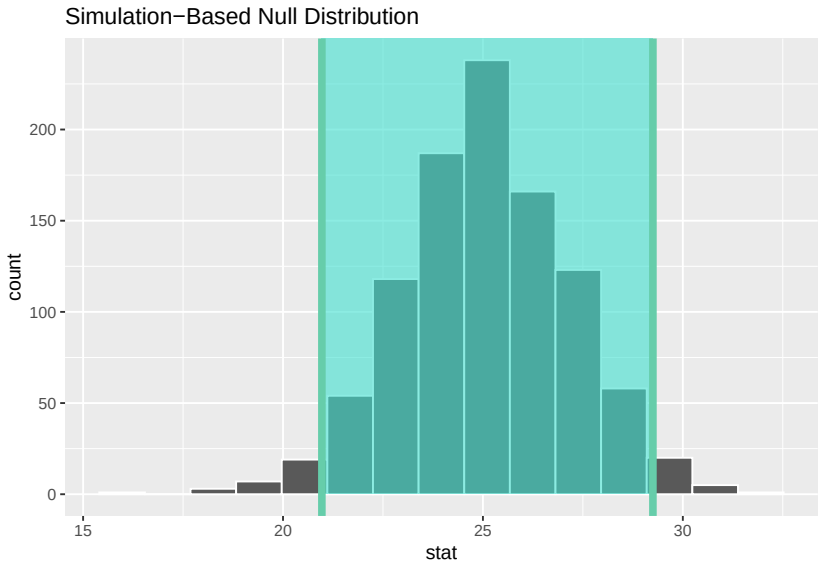
These options are the default values for `level` and `type` so we can also just do:

```
percentile_ci <- bootstrap_distribution %>%
  get_ci()
```

```
# A tibble: 1 x 2
  `2.5%` `97.5%`
   <dbl>   <dbl>
1   21.0    29.3
```

Using the percentile method, our range of plausible values for the mean age of US pennies in circulation in 2011 is 20.97 years to 29.25 years. We can use the `visualize` function to view this using the `endpoints` and `direction` arguments, setting `direction` to `"between"` (between the values) and `endpoints` to be those stored with name `percentile_ci`.

```
bootstrap_distribution %>%
  visualize(endpoints = percentile_ci, direction = "between")
```



You can see that 95% of the data stored in the `stat` variable in `bootstrap_distribution` falls between the two endpoints with 2.5% to the left outside of the shading and 2.5% to the right outside of the shading.

5.2 The standard error method

If the bootstrap distribution is close to symmetric and bell-shaped, we can also use a shortcut formula for determining the lower and upper endpoints of the confidence interval. This is done by using the formula $\bar{x} \pm (multiplier * SE)$, where $\bar{x}$ is our original sample mean and SE stands for **standard error** and corresponds to the standard deviation of the bootstrap/sampling distribution. The value of *multiplier* here is the appropriate percentile of the standard normal distribution.

These are automatically calculated when `level` is provided with `level = 0.95` being the default. (95% of the values in a standard normal distribution fall within 1.96 standard deviations of the mean, so $multiplier = 1.96$ for `level = 0.95`, for example). As mentioned, this formula assumes that the bootstrap distribution is symmetric and bell-shaped. This is often the case with bootstrap distributions, especially those in which the original distribution of the sample is not highly skewed.

The variability of the sampling distribution may be approximated by the variability of the bootstrap distribution. Traditional theory-based methodologies for inference also have formulas for standard errors, assuming some conditions are met (you will have seen some of these in Statistical Inference in Semester 1).

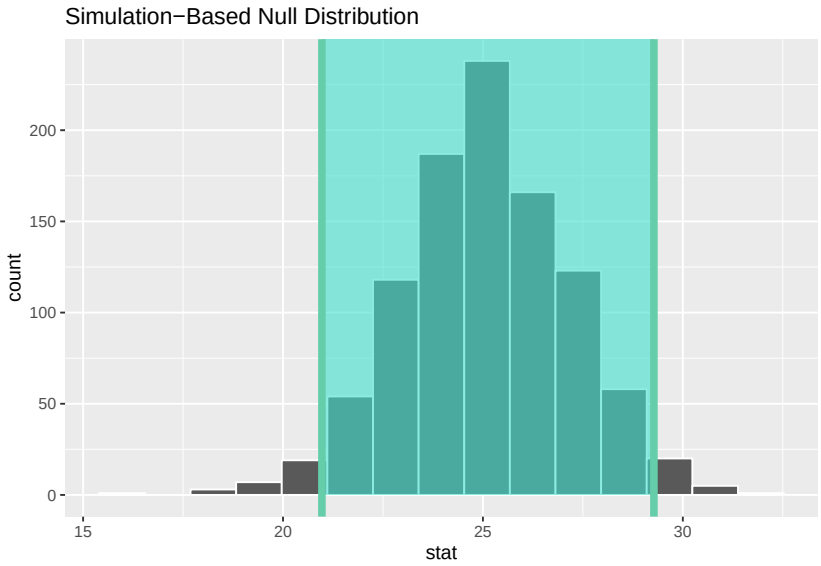
This $\bar{x} \pm (multiplier * SE)$ formula is implemented in the `get_ci()` function as shown with our pennies problem using the bootstrap distribution's variability as an approximation for the sampling distribution's variability. We'll see more on this approximation shortly.

Note that the center of the confidence interval (the `point_estimate`) must be provided for the standard error confidence interval.

```
standard_error_ci <- bootstrap_distribution %>%
  get_ci(type = "se", point_estimate = x_bar)
```

```
# A tibble: 1 x 2
  lower upper
  <dbl> <dbl>
1  21.0  29.3
```

```
bootstrap_distribution %>%
  visualize(endpoints = standard_error_ci, direction = "between")
```



We see that both methods produce nearly identical confidence intervals with the percentile method being $[20.97, 29.25]$ and the standard error method being $[20.97, 29.28]$.

6 Interpreting the confidence interval

Recall that the confidence intervals we've produced are based on bootstrapping using the single sample `orig_pennies_sample`. We have been claiming that this is a sample form all the pennies in circulation in 2011, but we can now reveal that it is actually a sample from a larger number of pennies stored as `pennies` in the `moderndive` package. The `pennies` data frame contains 800 rows of data and two columns pertaining to the same variables as `orig_pennies_sample`. It is important to stress that this is *very artificial*, i.e. we

would usually never have access to all the information about the larger group from which our sample is taken, but we have set up the data this way here to illustrate the properties of confidence intervals for the purpose of interpreting confidence intervals.

So let's assume that `pennies` is our population of interest (i.e. a population with $N = 800$ units). We can therefore calculate the population mean age of pennies in 2011, denoted by the Greek letter μ, by calculating the mean of `age_in_2011` for the `pennies` data frame.

```
pennies_mu <- pennies %>%
  summarize(overall_mean = mean(age_in_2011)) %>%
  pull() # we use this to extract a single value from the data frame
```

```
[1] 21.1525
```

As we saw at the end of the previous section, one range of plausible values for the population mean age of pennies in 2011 (μ), is $[20.97, 29.25]$. Note that the value $\mu = 21.15$ (i.e.the mean of `pennies` calculated above) **does** fall in this confidence interval. So in this instance, the confidence interval based on `orig_pennies_sample` was a good estimate of μ.

If we had a different sample of size 40 and constructed a confidence interval using the same method, would we be guaranteed that it contained the population parameter value μ as well? Let's try it out:

```
orig_pennies_sample2 <- pennies %>%
  sample_n(size = 40)
```

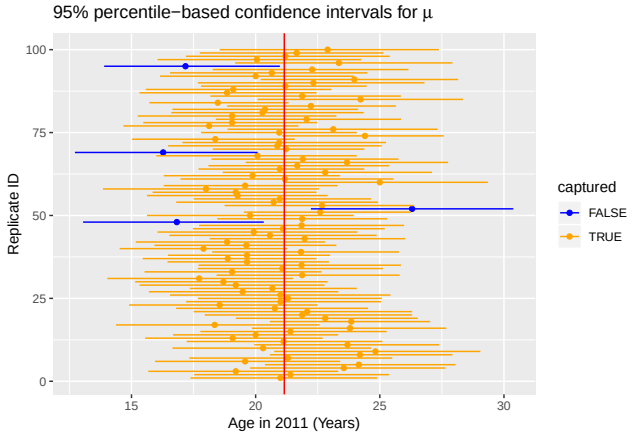
Note the use of the `sample_n` function in the `dplyr` package here. This does the same thing as `rep_sample_n(reps = 1)` but omits the extra `replicate` column.

We next create an `infer` pipeline to generate a percentile-based 95% confidence interval for μ:

```
percentile_ci2 <- orig_pennies_sample2 %>%
  specify(formula = age_in_2011 ~ NULL) %>%
  generate(reps = 1000) %>%
  calculate(stat = "mean") %>%
  get_ci()
```

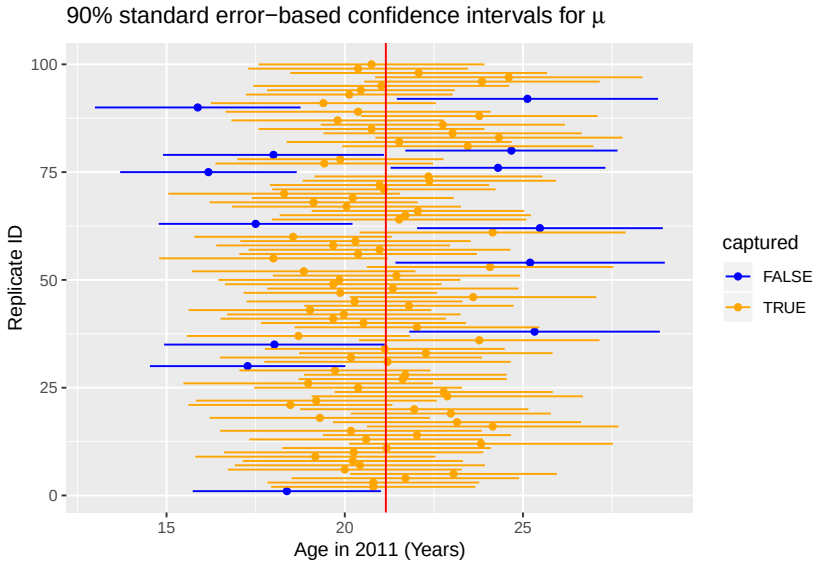
```
# A tibble: 1 x 2
  `2.5%` `97.5%`
   <dbl>   <dbl>
1   16.5    22.8
```

This new confidence interval also contains the value of μ. Let's further investigate by repeating this process 100 times to get 100 different confidence intervals derived from 100 different samples of `pennies`. Each sample will have size of 40 just as the original sample. We will plot each of these confidence intervals as horizontal lines. We will also show a line corresponding to the known population value of 21.15 years.



Of the 100 confidence intervals based on samples of size $n = 40$, 96 of them captured the population mean $\mu = 21.15$, whereas 4 of them did not include it. If we repeated this process of building confidence intervals more times with more samples, we'd expect 95% of them to contain the population mean. In other words, the procedure we have used to generate confidence intervals is "95% reliable" in that we can expect it to include the true population parameter 95% of the time if the process is repeated.

To further accentuate this point, let's perform a similar procedure using 90% confidence intervals instead. This time we will use the standard error method instead of the percentile method for computing the confidence intervals.



Of the 100 confidence intervals based on samples of size $n = 40$, 87 of them captured the population mean $\mu = 21.15$, whereas 13 of them did not include it. Repeating this process for more samples would result in us getting closer and closer to 90% of the confidence intervals including the true value. It is common to say while interpreting a confidence interval to be "95% confident" or "90% confident" that the true value falls within the range of the specified confidence interval. We will use this "confident" language throughout the rest of this chapter, but remember that it has more to do with a measure of reliability of the building process.

Back to our pennies example

After this elaboration on what the level corresponds to in a confidence interval, let's conclude by providing an interpretation of the original confidence interval result we found in the last section.

Interpretation: We are 95% confident that the true mean age of pennies in circulation in 2011 is between 20.97 and 29.25 years. This level of confidence is based on the percentile-based method including the true mean 95% of the time if many different samples (not just the one we used) were collected and confidence intervals were created.

7 Comparing two proportions

Table 1 (repeated): Scenarios of sample statistics for inference.

Scenario	Population Parameter	Population Notation	Sample Statistic	Sample Notation
1	Population proportion	p	Sample proportion	$\widehat{p}$
2	Population mean	μ	Sample mean	$\bar{x}$
3	Diff.in pop. props	$p_1 - p_2$	Diff. in sample props	$\widehat{p}_1 - \widehat{p}_2$
4	Diff. in pop. means	$\mu_1 - \mu_2$	Diff. in sample means	$\bar{x}_1 - \bar{x}_2$
5	Pop. intercept	β_0	Sample intercept	$\widehat{\beta}_0$ or b_0
6	Pop. slope	β_1	Sample slope	$\widehat{\beta}_1$ or b_1

In the previous sections we have considered Scenario 2 in Table 1 (reproduced here), i.e. constructing a confidence interval for a single population mean. Often, however, interest lies in comparing two populations, e.g. by constructing a confidence interval for the difference in the population means, as in Scenario 4. But it may be that the characteristic of interest is a population proportion (rather than a population mean) which is reflected in Scenarios 1 and 3. In this section we will focus on Scenario 3, i.e. constructing a confidence interval for the difference in two population proportions.

Let's start with an example. If you see someone else yawn, are you more likely to yawn? In an episode of the TV show *Mythbusters*, they tested the myth that yawning is contagious.

Fifty adults who thought they were being considered for an appearance on the show were interviewed by a show recruiter ("confederate") who either yawned or did not. Participants then sat by themselves in a large van and were asked to wait. While in the van, the Mythbusters watched via hidden camera to see if the unaware participants yawned. The data frame containing the results is available at `mythbusters_yawn` in the `moderndive` package. Let's check it out.

```
mythbusters_yawn
```

```
# A tibble: 50 x 3
    subj group   yawn
   <int> <chr>   <chr>
 1     1 seed    yes
 2     2 control yes
 3     3 seed    no
 4     4 seed    yes
 5     5 seed    no
 6     6 control no
 7     7 seed    yes
 8     8 control no
 9     9 control no
10    10 seed    no
# ... with 40 more rows
```

- The participant ID is stored in the `subj` variable with values of 1 to 50.
- The `group` variable is either `seed` for when a confederate was trying to influence the participant or `control` if a confederate did not interact with the participant.
- The `yawn` variable is either `yes` if the participant yawned or `no` if the participant did not yawn.

We can use the `janitor` package to get a glimpse into this data in a table format:

```
mythbusters_yawn %>%
  tabyl(group, yawn) %>%
  adorn_percentages() %>%
```

```
  adorn_pct_formatting() %>%
  # To show original counts
  adorn_ns()
```

```
   group         no        yes
 control 75.0% (12) 25.0%  (4)
    seed 70.6% (24) 29.4% (10)
```

We are interested in comparing the proportion of those that yawned after seeing a seed versus those that yawned with no seed interaction. We'd like to see if the difference between these two proportions is significantly larger than 0. If so, we'd have evidence to support the claim that yawning is contagious based on this study.

In looking over this problem, we can make note of some important details to include in our `infer` pipeline:

- We are calling a `success` having a `yawn` value of `yes`.
- Our response variable will always correspond to the variable used in the `success` so the response variable is `yawn`.
- The explanatory variable is the other variable of interest here: `group`.

To summarise, we are looking to examine the relationship between yawning and whether or not the participant saw a seed yawn or not.

7.1 Compute the point estimate

```
mythbusters_yawn %>%
  specify(formula = yawn ~ group)
```

Note that the `success` argument must be specified in situations such as this where the response variable has only two levels.

```
mythbusters_yawn %>%
  specify(formula = yawn ~ group, success = "yes")
```

```
Response: yawn (factor)
Explanatory: group (factor)
# A tibble: 50 x 2
   yawn  group
   <fct> <fct>
 1 yes   seed
 2 yes   control
 3 no    seed
 4 yes   seed
 5 no    seed
 6 no    control
 7 yes   seed
 8 no    control
 9 no    control
10 no    seed
# ... with 40 more rows
```

We next want to calculate the statistic of interest for our sample. This corresponds to the difference in the proportion of successes.

```
mythbusters_yawn %>%
  specify(formula = yawn ~ group, success = "yes") %>%
  calculate(stat = "diff in props")
```

We see another error here. To further check to make sure that R knows exactly what we are after, we need to provide the `order` in which R should subtract these proportions of successes. As the error message states, we'll want to put `"seed"` first after `c()` and then `"control"`: `order = c("seed", "control")`. Our point estimate is thus calculated:

```
obs_diff <- mythbusters_yawn %>%
  specify(formula = yawn ~ group, success = "yes") %>%
  calculate(stat = "diff in props", order = c("seed", "control"))
obs_diff
```

```
# A tibble: 1 x 1
    stat
   <dbl>
1 0.0441
```

This value represents the proportion of those that yawned after seeing a seed yawn (0.2941) minus the proportion of those that yawned with not seeing a seed (0.25).

7.2 Bootstrap distribution

Our next step in building a confidence interval is to create a bootstrap distribution of statistics (differences in proportions of successes). We saw how it works with a single variable in computing bootstrap means in the `pennies` example but we haven't yet worked with bootstrapping involving multiple variables, i.e. comparing two groups. In the `infer` package, bootstrapping with multiple variables means that each **row** is potentially resampled. Let's investigate this by looking at the first few rows of `mythbusters_yawn`:

```
head(mythbusters_yawn)
```

```
# A tibble: 6 x 3
   subj group   yawn
  <int> <chr>   <chr>
1     1 seed    yes
2     2 control yes
3     3 seed    no
4     4 seed    yes
5     5 seed    no
6     6 control no
```

When we bootstrap this data, we are potentially pulling the subject's readings multiple times. Thus, we could see the entries of `seed` for `group` and `no` for `yawn` together in a new row in a bootstrap sample. This is further seen by exploring the `sample_n` function in `dplyr` on this smaller 6 row data frame comprised of `head(mythbusters_yawn)`. The `sample_n` function can perform this bootstrapping procedure and is similar to the `rep_sample_n` function in `infer`, except that it is not `rep`eated but rather only performs one sample with or without replacement.

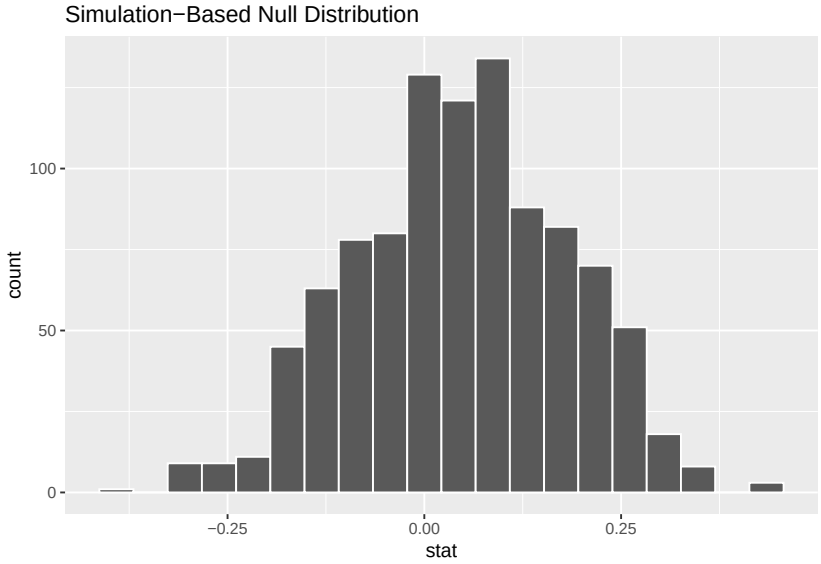
```
head(mythbusters_yawn) %>%
  sample_n(size = 6, replace = TRUE)
```

```
# A tibble: 6 x 3
   subj group   yawn
  <int> <chr>   <chr>
1     5 seed    no
2     5 seed    no
3     2 control yes
4     4 seed    yes
5     1 seed    yes
6     1 seed    yes
```

We can see that in this bootstrap sample generated from the first six rows of `mythbusters_yawn`, we have some rows repeated. The same is true when we perform the `generate` step in `infer`:

```
bootstrap_distribution <- mythbusters_yawn %>%
  specify(formula = yawn ~ group, success = "yes") %>%
  generate(reps = 1000) %>%
  calculate(stat = "diff in props", order = c("seed", "control"))
```

```
bootstrap_distribution %>%
  visualize(bins = 20)
```



This distribution is roughly symmetric and bell-shaped but isn't quite there. Let's use the percentile-based method to compute a 95% confidence interval for the true difference in the proportion of those that yawn with and without a seed presented. The arguments are explicitly listed here but remember they are the defaults and simply `get_ci` can be used.

```
bootstrap_distribution %>%
  get_ci(type = "percentile", level = 0.95)
```

```
# A tibble: 1 x 2
  `2.5%` `97.5%`
   <dbl>   <dbl>
1 -0.219   0.293
```

The confidence interval shown here is (-0.22, 0.29) and therefore includes the value of 0. The range of plausible values for the difference in the proportion of those that yawned with and without a seed is therefore between -0.22 and 0.29.

Therefore, we are not sure which proportion is larger. Some of the bootstrap statistics showed the proportion without a seed to be higher and others showed the proportion with a seed to be higher. If the confidence interval was entirely above zero, we would be relatively sure (about "95% confident") that the seed group had a higher proportion of yawning than the control group.

Note that this all relates to the importance of denoting the `order` argument in the `calculate` function. Since we specified `seed` and then `control` positive values for the statistic correspond to the `see"` proportion being higher, whereas negative values correspond to the `control` group being higher.

We, therefore, have evidence via this confidence interval suggesting that the conclusion from the Mythbusters show that "yawning is contagious" being "confirmed" is not statistically appropriate.

8 Further Tasks

You are encouraged to complete the following tasks by using RMarkdown to produce a single document which summarises all your work, i.e. the original questions, your R code, your comments and reflections, etc.

1. In the last section, we constructed a confidence interval for the difference in the proportion of people who yawned between the "seeded" group and the "control" group (Scenario 3).

   By modifying the code in the last section in light of how we constructed a confidence interval for the age of pennies in the section on "Constructing confidence intervals" (Scenario 2), use `orig_pennies_sample` data to constuct a confidence interval for the proportion of people who yawn when they see someone else yawn (Scenario 1). Does this overlap with the confidence interval for the proportion of people who yawn when they did not see someone else yawn (Scenario 1)? Are your findings here consistent with the findings in the last section?

2. Recall the data on 144 domestic male and female adult cats that we first saw in Week 4 (`cats` from the `MASS` library). Each cat had their heart weight in grams (`Hwt`) and body weight in kilograms (`Bwt`) measured, and interest lies in exploring difference between females and males.

   a. Construct bootstrap confidence intervals for the average heart weight of female and male cats separately? Interpret your results.

   b. Construct a bootstrap confidence interval for the difference in the average heart weights of female and male cats. Interpret your result.

   c. Repeat a. and b. for the body weight of cats.