Bonus material: Bootstraps and Permutations

In this material:

- Go over another "recipe" for creating confidence intervals (bootstrap)
- Touch on permutations

**note, bootstraping and permutations are not in your book and will be included in the exam only as extra credit

Confidence interval recap

- So far we've (mostly) been using formulas of the form estimate $\pm$ margin of error to generate confidence intervals.
- We've applied these formulas to the mean $\bar{x}$ and the proportion $\hat{p}$. These "large sample" procedures use the Central Limit Theorem to calculate the standard error of the sampling distribution for $\bar{x}$ and $\hat{p}$

Confidence interval recap

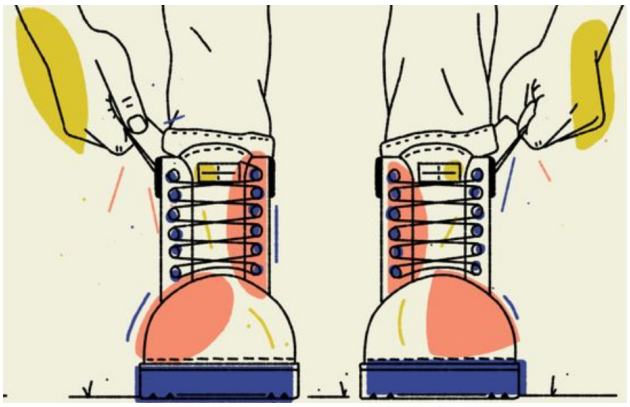
- We did this when the data met certain assumptions, one of which being that the data were drawn from a population that is Normally distributed.
- Though there was some leeway around this assumption, what would happen if you were sure the assumption was violated? How would you calculate a 95% confidence interval in that case?
- **We need a procedure to calculate confidence intervals that do not rely on the assumption of Normality.**

Confidence intervals for other parameters

- What if we wanted a confidence interval for the median? Or a confidence interval for the first quartile, Q_1, or some other parameter?
- We couldn't use the same procedure because the CLT does not apply to the sampling distribution of the median or $Q1$, or to many other statistics.
- **We need a procedure to calculate confidence intervals for other parameters, such as the median or any other quartile or percentile.**

The Bootstrap approach

Enter: The Bootstrap Confidence Interval



- It was so-named because it uses only the information from the sample you have to estimate the CI, such that the sample is "pulling itself up by its bootstraps".

Enter: The Bootstrap Confidence Interval

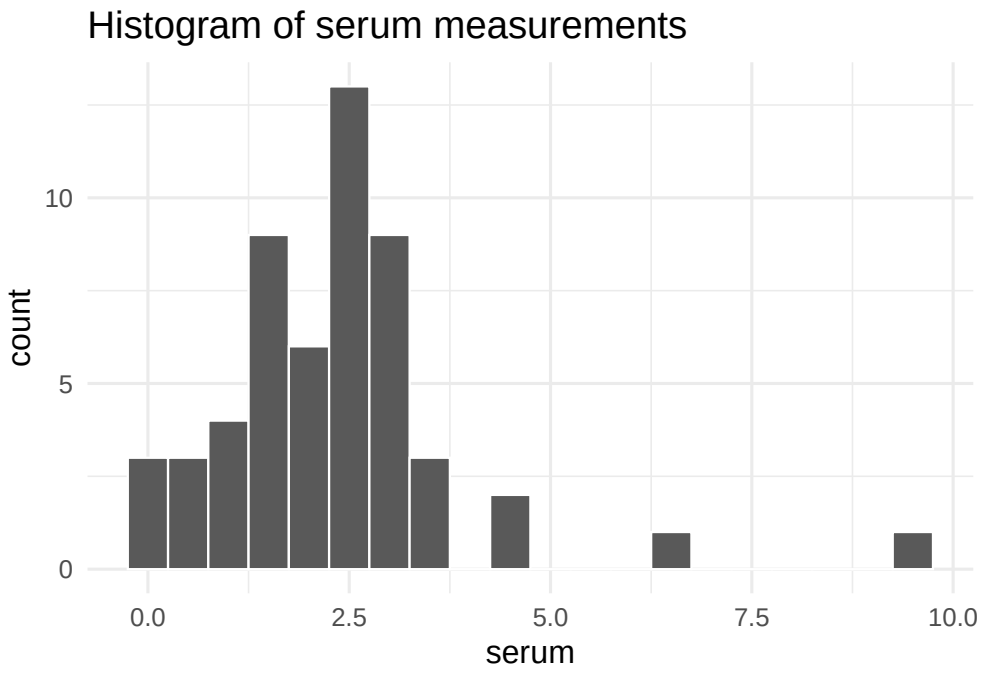
- The Bootstrap CI does not require the underlying distribution to be Normally distributed.
- The Bootstrap CI can be made for any parameter (based on its sample statistic).
- This method gained popularity in 1979 when popularized by Bradley Efron, and harnesses current computing power.
- **It is a very popular method used to compute confidence intervals today**

Bootstrap example

This example came from bootstrap's advocate, Bradley Efron. Suppose we have measures of serum from 54 patients:

```
serum_data <- data.frame(serum = c(0.1, 0.1, 0.2, 0.4, 0.4, 0.6, 0.8, 0.8,
                                   0.9, 0.9, 1.3, 1.3, 1.4, 1.5, 1.6, 1.6, 1.7,
                                   1.7, 1.7, 1.8, 2.0, 2.0, 2.2, 2.2, 2.2, 2.3,
                                   2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.5, 2.5,
                                   2.5, 2.7, 2.7, 2.8, 2.9, 2.9, 2.9, 3.0, 3.1,
                                   3.1, 3.2, 3.2, 3.3, 3.3, 3.5, 4.4, 4.5, 6.4,
                                   9.4))
```

Serum



First, calculate the theory-based 95% CI: dplyr()

```
t_star <- qt(0.975, df = 53) # find t* to use to calculate the 95% CI

serum_data %>% summarise(mean_serum = mean(serum),
                         se_serum = sd(serum)/sqrt(n()),
                         lower_CI = mean_serum - t_star * se_serum,
                         upper_CI = mean_serum + t_star * se_serum)
```

```
##   mean_serum  se_serum lower_CI upper_CI
## 1   2.318519 0.2078991 1.901526 2.735511
```

First, calculate the theory-based 95% CI: t.test()

```
t.test(serum_data %>% pull(serum))
```

```
##
##  One Sample t-test
##
## data:  serum_data %>% pull(serum)
## t = 11.152, df = 53, p-value = 1.62e-15
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  1.901526 2.735511
## sample estimates:
## mean of x
##  2.318519
```

The 95% CI provides our best guess of where the true proportion lies. The 95% CI for μ is 1.90 to 2.74. We found this interval using a method that gives an interval that captures μ 19 times out of 20.

95% CI for the median

- We can't use the above method to compute a 95% CI for the median because the CLT does not apply to the median.
- However, we can use a method that relies on the assumption that the sample is a SRS from the underlying population

How the Bootstrap CI is made

- If we truly have a SRS from the underlying population, this means that the distribution of serum in the sample should *approximate* the distribution of serum in the population
- Graphically, this means the shape of the histogram for the sample data should approximate the shape of the density plot for the entire population
- **The key: If we take repeated samples (with replacement) from our sample, we can approximate the sampling distribution for any statistic we'd like**
- This is the process of bootstrapping
- Let's apply this method to calculate the 95% CI for the median using the serum data

Boostrap confidence interval for the median

1. Calculate the median for the original sample of size 54. Denote this value by m. This is our estimate of the median for the underlying population. We need to create a 95% CI around m.

```
median_serum <- serum_data %>% summarise(median_serum = median(serum))
median_serum
```

```
##   median_serum
## 1         2.35
```

2. Resample with replacement from the original sample a new sample, also of size 54.

Boostrap confidence interval for the median

3. Calculate the median based on resample #1. Call this median m_1^*.
4. Resample again. Calculate the median based on resample #2. Call this median m_2^*. ... repeat this resampling procedure several thousand times.
5. Make a histogram of m_1^*, m_2^*, ... , m_{1000}^*. **This histogram approximates the sampling distribution for the median**

Boostrap confidence interval for the median

6. Calculate the bounds such that the middle 95% of the observations are between the lower and upper bounds. In R, we can do this using `quantile(sample_median, 0.025)` and `quantile(sample_median, 0.975)` to locate the 2.5th and 97.5th percentiles of the variable `sample_median`.

Boostrap confidence interval for the median

The code embeded in the lecture here resamples the data 1000 times and calculates the median for each resample. You do not need to know how to do this in R. It stores the median in a data frame called `many_sample_medians`

```
head(many_sample_medians)
```

```
##   sample_median sample.id
## 1           2.4         1
## 2           2.4         2
## 3           2.5         3
```

```
## 4           2.4         4
## 5           2.4         5
## 6           2.4         6
```

Calculate the lower and upper bounds of the 95% bootstrap CI

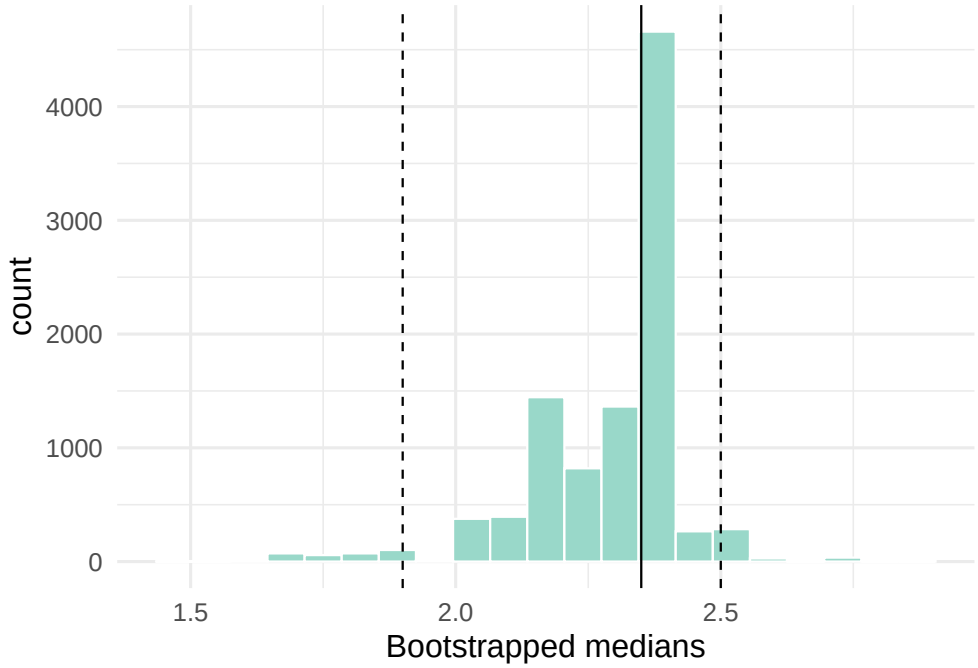
```
# Understand this code. It takes the data frame containing the bootstrap
# sample medians and calculates the lower and upper CI using the quantile function.

bounds <- many_sample_medians %>%
  summarise(lower_CI = quantile(sample_median, 0.025),
            upper_CI = quantile(sample_median, 0.975))

bounds
```

```
##   lower_CI upper_CI
## 1      1.9      2.5
```

```
# how would the code change if we were interested in the 99% CI?
```

Thus, our best estimate of the median is 2.35. The bootstrapped 95% confidence interval for the median is 1.9 to 2.5.

Plot the histogram of the bootstrapped medians and denote the 95% confidence interval



- Note that the sampling distribution for the median is not symmetric. It is skewed left (for these data).
- The CI is not symmetric around our best guess for the sample mean (2.35).

Summary on Bootstrap CIs

- The bootstrap is a method that we use to estimate confidence intervals
- It is particularly useful when:
  - We don't have a nice formula to calculate the CI, or we don't know what the formula is

  - The underlying assumptions of using a "large sample" formula are not satisfied
- We can make bootstrap CIs around many statistics we've learnt about: the median, the quartiles, the correlation coefficient.
- The basic method is to sample our data with replacement, and generate a statistic for each sample
- The distribution of these estimates is used to construct our CI

Permutation

Permutation tests

The majority of the methods we've used so far for hypothesis testing (z-tests, t-tests, and chi-square tests) have depended on having large enough sample sizes for the inference to be valid. They have also required that the sample was a SRS from some larger population.

Today we will talk about another method for conducting hypothesis tests that do not require either assumption.

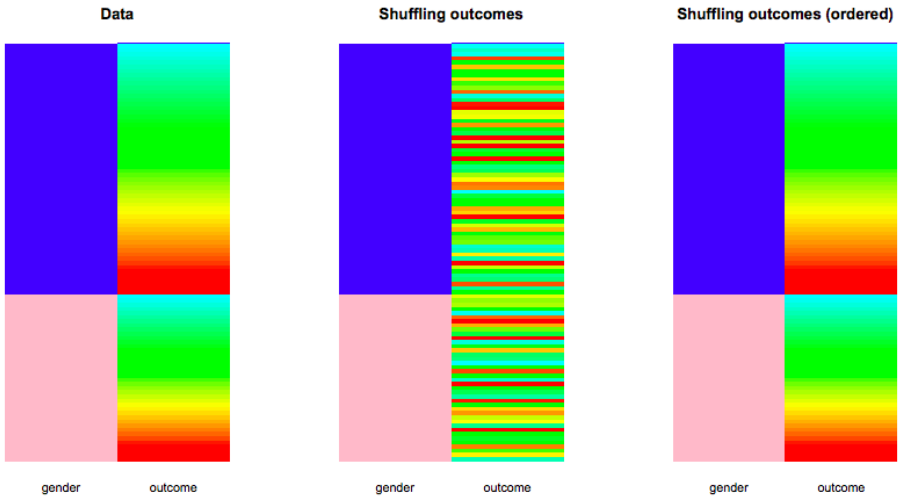
It might remind you of our bootstrapping lecture, but remember, bootstrapping was for confidence intervals, whereas permutation tests are for hypothesis testing.

Permutation tests

- Permutation tests are another way to get p-values for hypothesis tests.
- These tests are based on reshuffling (or permuting) the data to break any relationship between the two variables.

Permutation test, shown visually

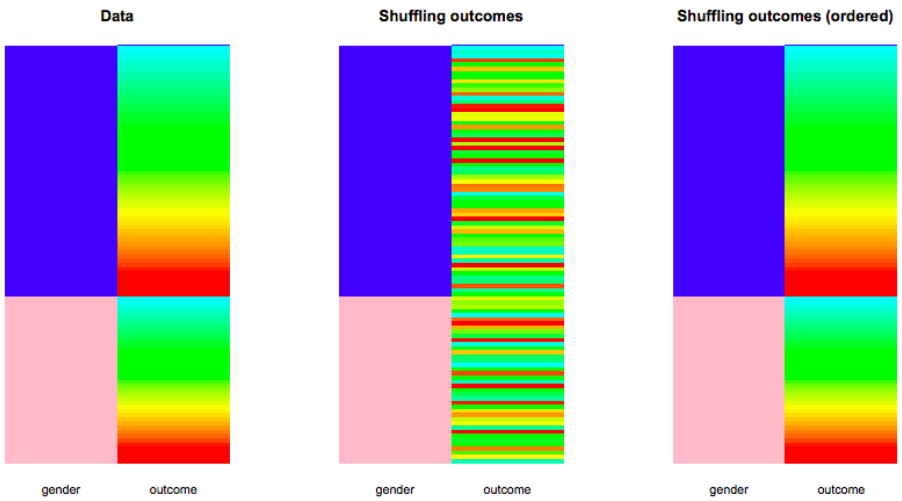
Example: null is true



If the null is *true* then the distribution of the response variable is the same for each level of the explanatory variable. This is shown by the entire spectrum of colors for both levels of the explanatory variable in this plot.

Permutation test, shown visually

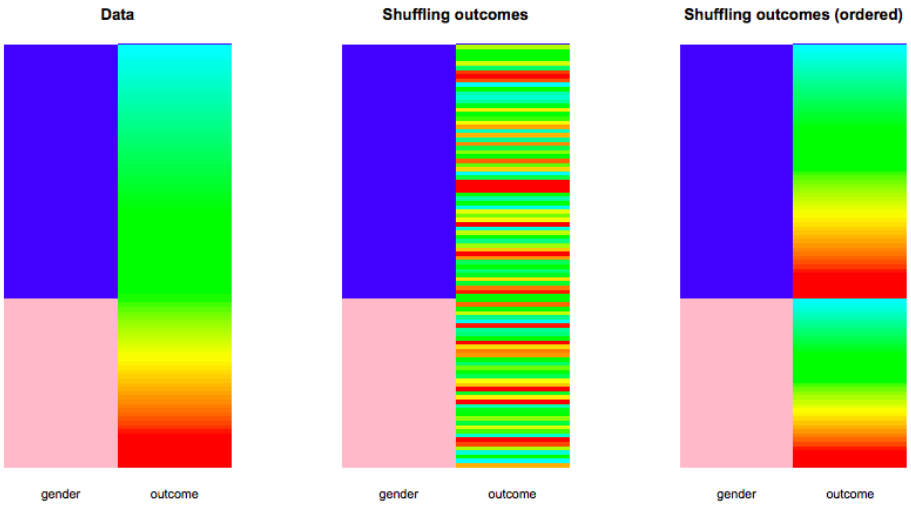
Example: null is true



After reshuffling, the distribution comes out the same. This illustrates that if the null is true, your observed statistic will look like a random reshuffle. reference: http://faculty.washington.edu/kenrice/sisg/SISG-08-06.pdf

Permutation test, shown visually

Example: null is false



If the null is *false* the distribution of the response variable varies for each level of the explanatory variable.

Permutation tests

For a test looking at difference between 2 groups:

- We summarize the outcome in each of our two groups (2 means, 2 proportions etc)
- The null hypothesis is that there is no difference between these two groups.
- Consider the number of possible configuration of the experimental group and the observed response.
- Generate either a full set of all possible configurations or a random set of a desired # of possible outcomes
- For each possible configuration, compute the outcome in each group and our desired measure (difference)

- Create the distribution of all computed outcomes
- Calculate a p-value based upon our observed response relative to all of the potential experimental groups.

Example: Beer consumption and mosquito attraction to humans

Background: Malaria and alcohol consumption both represent major public health problems. Alcohol consumption is rising in developing countries and, as efforts to manage malaria are expanded, understanding the links between malaria and alcohol consumption becomes crucial. Our aim was to ascertain the effect of beer consumption on human attractiveness to malaria mosquitoes in semi field conditions in Burkina Faso. - Lefevre et al, 2010, in *PLOS One*

Example: Beer consumption and mosquito attraction to humans

- Volunteers were randomly assigned to drink either beer or water
- Batches of mosquitoes were inside a device and could choose to fly towards the human participant or towards the open air
- The number of mosquito flying towards the human were counted for each participant

Example: Beer consumption and mosquito attraction to humans

The data:

```
beer <- c(27, 19, 20, 20, 23, 17, 21, 24, 31, 26, 28, 20,
          27, 19, 25, 31, 24, 28, 24, 29, 21, 21, 18, 27,
          20)

water <- c(21, 19, 13, 22, 15, 22, 15, 22, 20, 12, 24, 24,
           21, 19, 18, 16, 23, 20)

mosq_data <- data.frame(num_mosquitos = c(beer, water),
                        treatment = c(rep("beer", 25),
                                      rep("water", 18)))
```

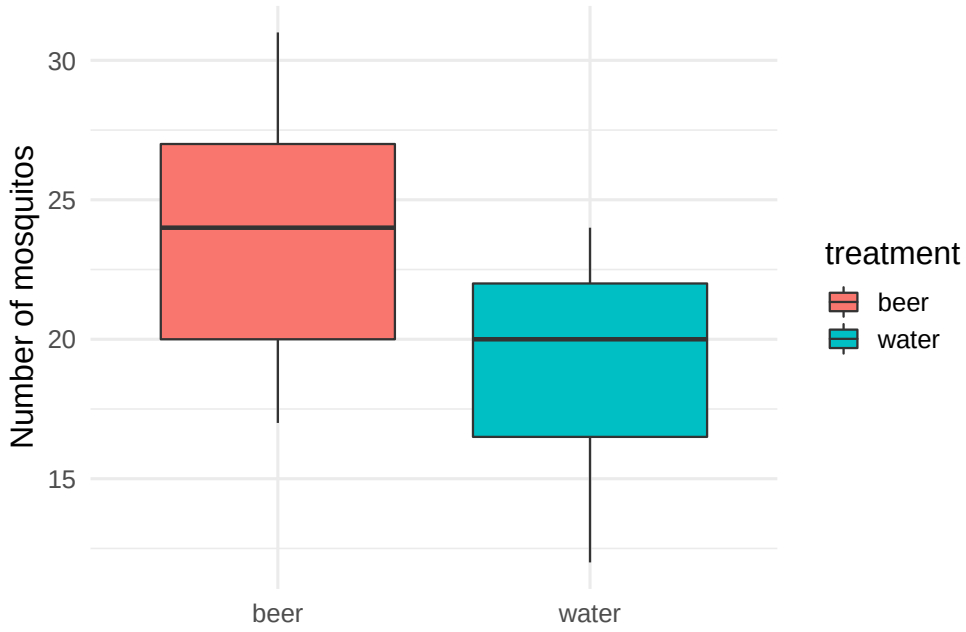
Example: Beer consumption and mosquito attraction to humans

`beer` is a vector of the count of mosquitoes the flew towards the participant for the 25 people randomized to beer

`water` is the same thing but for those 18 people randomized to drink water

Example: Beer consumption and mosquito attraction to humans

Descriptives: Does there look to be a difference between the groups?



Example: Beer consumption and mosquito attraction to humans

What type of variable is the exposure here?

What type of variable is the outcome?

How might we approach this with a parametric test?

Example: Beer consumption and mosquito attraction to humans

```
t.test(beer, water, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  beer and water
## t = 3.6582, df = 39.113, p-value = 0.0007474
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.957472 6.798084
## sample estimates:
## mean of x mean of y
##  23.60000  19.22222
```

The average number of mosquitoes attracted to beer drinkers was 23.6 vs. 19.22 attracted to water drinkers. What is the p-value?

Example: Beer consumption and mosquito attraction to humans

There is another way to perform this test. Consider the null hypothesis:

$$H_0 : \mu_1 = \mu_2$$

If the two means are the same, then we would expect no difference between the number of mosquitoes attracted to beer drinkers vs. water drinkers.

Example: Beer consumption and mosquito attraction to humans

Permutation approach:

Assuming the null is true:

We could mix up the labels of who drank beer and who had water and re-compute the difference between beer- and water-drinkers in the number of mosquitoes.

We could do this many times. For each shuffling of the labels, we could re-compute the difference and mark it on a histogram.

Example: Beer consumption and mosquito attraction to human

Watch this clip from 8:13-9:52: https://youtu.be/5Dnw46eC-0o?t=492.

- It shows the sampling distribution being built for this example under the null hypothesis of no difference.
- It shows how the labels can be shuffled at random, and after each re-shuffling, the mean difference is computed and plotted on an evolving histogram.
- Then a vertical line is added at the **observed** value of the difference (based on the data from the sample).
- An observed value in the tails of the distribution implies that it is unlikely to occur under the null hypothesis.

The 'infer' package in R

The `infer` package

The `infer` package is relatively new to the tidyverse (which includes `ggplot2`, `readr`, `dplyr`, among others)

It is **awesome** because it interjects the steps of hypothesis testing directly into the code. It also keeps things "tidy" meaning that the output is often returned in a nice little data frame.

You can use `infer` to conduct permutation tests, but if you're interested you could also learn more here about doing all your testing using this package.

Let's have a look!

The `infer` package for permutation tests

First use the `infer` functions `specify()`, `hypothesize()`, `generate()`, and `calculate` to create the histogram of the sampling distribution for the mean difference:

```
null_distn <- mosq_data %>%
  specify(response = num_mosquitos, explanatory = treatment) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000, type = "permute") %>%
  calculate(stat = "diff in means", order = c("beer", "water"))

head(null_distn)
```

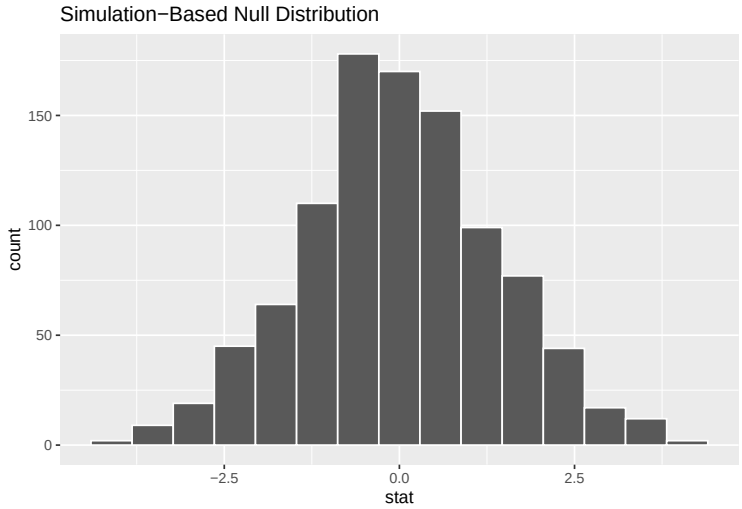
```
## Response: num_mosquitos (numeric)
## Explanatory: treatment (factor)
## Null Hypothesis: independence
## # A tibble: 6 x 2
##   replicate   stat
##       <int>  <dbl>
```

```
## 1         1 -2.02
## 2         2 -0.209
## 3         3 -2.12
## 4         4 -2.02
## 5         5 -0.496
## 6         6  0.842
```

The `infer` package for permutation tests

Then, use the `infer` function `visualize` to plot the sampling distribution, add a line at the observed mean difference, and shade the region corresponding to the p-value:

```
null_distn %>% visualize(obs_stat = 23.6-19.22, direction = "two_sided")
```



The `infer` package for permutation tests

Finally, calculate the p-value by using the `get_pvalue()` function:

```
null_distn %>% get_pvalue(obs_stat = 23.6-19.22, direction = "two_sided")
```

```
## Warning: Please be cautious in reporting a p-value of 0. This result is an
## approximation based on the number of `reps` chosen in the `generate()` step. See
## `?get_p_value()` for more information.
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1       0
```

In summary

- Permutation tests are another way to get p-values for hypothesis tests.
- There is a permutation test equivalent for all the two sample tests that we've covered.
- Permutation tests rely on reshuffling (or permuting) the data to break any relationship between the two variables.
- The `infer` package is a good way to conduct and visualize permutation tests in R.