Homework 7: Confidence intervals and the dance of the p-values and p-hacking

name and student ID

Today's date

Run this chunk of code to load the autograder package!

Instructions

- Solutions will be posted on Sunday, October 18th.

Helpful hints:

- Every function you need to use was taught during lecture! So you may need to revisit the lecture code to help you along by opening the relevant files on Datahub. Alternatively, you may wish to view the code in the condensed PDFs posted on the course website. Good luck!
- Knit your file early and often to minimize knitting errors! If you copy and paste code for the slides, you are bound to get an error that is hard to diagnose. Typing out the code is the way to smooth knitting! We recommend knitting your file each time after you write a few sentences/add a new code chunk, so you can detect the source of knitting errors more easily. This will save you and the GSIs from frustration!
- To avoid code running off the page, have a look at your knitted PDF and ensure all the code fits in the file. If it doesn't look right, go back to your .Rmd file and add spaces (new lines) using the return or enter key so that the code runs onto the next line.
- Useful mathematical notation in markdown:

$$\mu$$

$$\sigma$$

Part 1: Confidence intervals

Recall the Alameda dataset from prior labs: suppose you had a data frame containing the **entire population** of all residents of Alameda County. There are data on four variables:

1. Born either out (=1) versus in (=0) the county.
2. Number of siblings (integer)
3. Number of visits to the hospital last year
4. Height (in inches)

Today we will focus on the height variable to construct confidence intervals from many samples.

Read in the data, `HW07_Alameda.csv` (it lives in the data folder) and assign it to the name `alameda_pop`.

```
library(dplyr)
library(ggplot2)
library(readr)

alameda_pop <- read_csv("data/HW07_Alameda.csv")
```

1. [1 point] Calculate the true (population) mean and standard deviation of the `height` variable in the Alameda population dataset. To do this, use a `dplyr` function to make a dataframe called `height_mean_sd` with the first column called `mean_height` and the second column called `sd_height`.

```
height_mean_sd <- alameda_pop %>% summarize(mean_height = mean(height), sd_height = sd(height))

height_mean_sd
```

```
## # A tibble: 1 x 2
##   mean_height sd_height
##         <dbl>     <dbl>
## 1        70.0      2.79
```

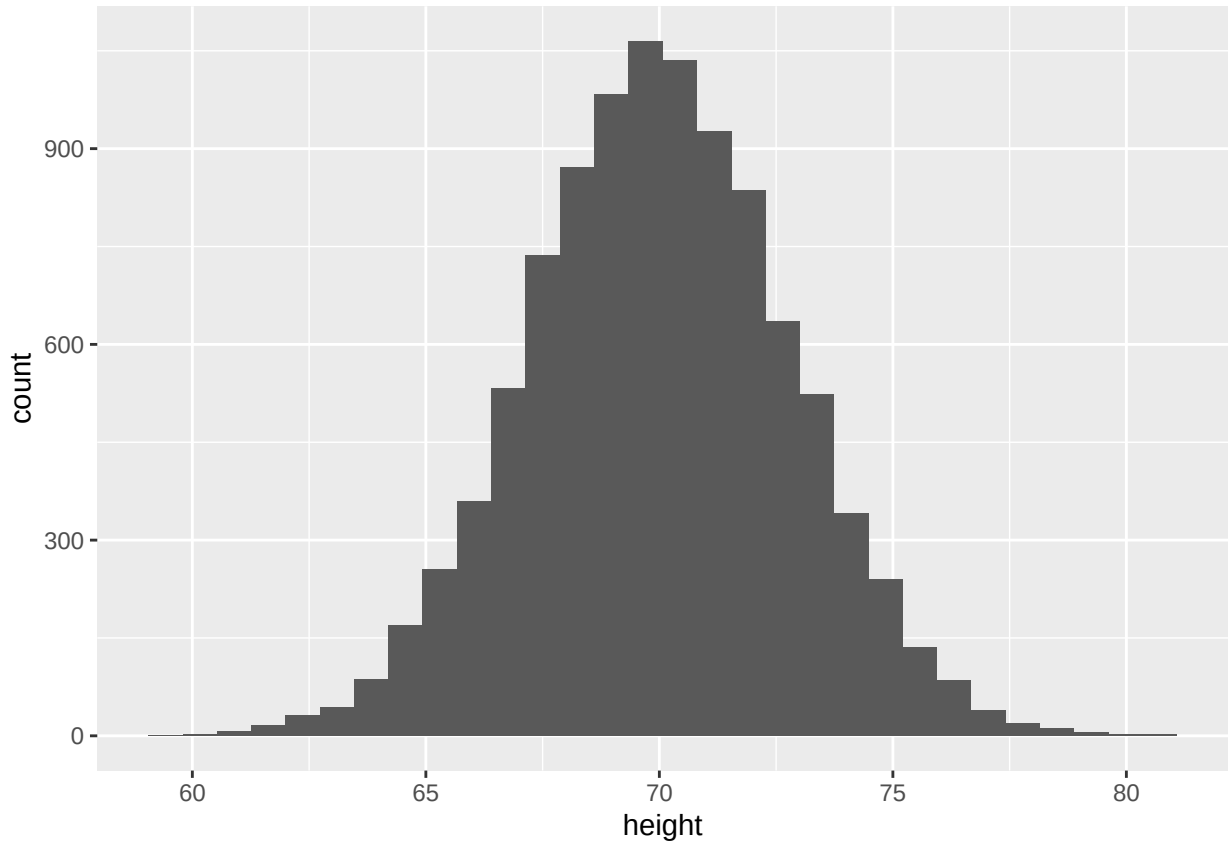
```
. = ottr::check("tests/height_mean_sd.R")
```

```
##
## All tests passed!
```

2. [1 point] Use `ggplot()` to make a histogram of `height`, assign it to the object `p2`, and comment on its distribution. Does it look Normally distributed?

```
p2 <- ggplot(alameda_pop, aes(x = height)) +
  geom_histogram()
p2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
. = ottr::check("tests/p2.R")
```

```
##
## All tests passed!
```

3. [1 point] Save the known population standard deviation of the `height` variable to the object called `known_pop_sd` below.

```
known_pop_sd <- 2.786314
known_pop_sd
```

```
## [1] 2.786314
```

```
. = ottr::check("tests/known_pop_sd.R")
```

```
##
## All tests passed!
```

Now suppose you *do not know* the population mean, but wanted to estimate it based on a sample. In this lab, we actually know the true value because we calculated it above. This way, we can see how well any one sample estimates the population mean and see how often the confidence intervals contain the mean across several repeated samples.

Calculating the CI and looking at its performance

We are now going to compute 95% confidence intervals (CI) for sampling means of different sizes. For this lab we:

- Have a variable with an underlying Normal distribution
- Will take simple random samples (SRS) from this distribution
- Know the value of the population mean (from your calculation in the first code chunk)

Thus, we satisfy the three conditions discussed in lecture for calculating a confidence interval when the underlying SD is known.

Recall the formula for the 95% confidence interval in this setting (hover your mouse within the double-dollar signs to see the formula or knit the file to read it more easily):

$$\bar{x} \pm 1.96 \times \frac{\sigma}{\sqrt{n}}$$

Where:

- $\bar{x}$ is the estimated mean based on your sample
- σ is the known standard deviation `known_pop_sd`, that you saved earlier for the distribution of `height`
- $\sigma/\sqrt{n}$ is the standard error of the sampling distribution for $\bar{x}$
- 1.96 is the critical value for a 95% CI

You will take a few samples from the distribution of heights from the Alameda population dataset and calculate the mean of your sampled heights and its confidence interval using the above formula. We will then simulate samples of different sizes and plot all the CIs.

Your task

4. Randomly generate 3 simple random samples of size $n = 10$ from the population. Calculate the average height for each of your samples. We wrote code to start you off, but you need to replace the three instances of NULL with calculations to compute the sample mean (`sample_mean_heights`), the lower confidence interval (`lower_CI`) and the upper confidence interval (`upper_CI`).

Hint: Review the above section for tips on how to calculate the CI if you forget. Once you do this, you can simply copy and paste 3 times (or bonus: use a for loop) to generate three randomly-drawn samples, the sample means, and confidence intervals.

```
sample_size <- 10
critical_value <- 1.96
size_10 <- sample_n(alameda_pop, sample_size, replace = FALSE)
p8 <- size_10 %>% summarise(mean_heights = mean(height)) %>%
mutate(lower_CI = mean_heights - critical_value*(known_pop_sd/sqrt(sample_size)),
       upper_CI = mean_heights + critical_value*(known_pop_sd/sqrt(sample_size))
)

# solution using for loop

for(i in 1:3){
size_10 <- sample_n(alameda_pop, sample_size, replace = FALSE)
print(p8 <- size_10 %>% summarise(mean_heights = mean(height)) %>%
mutate(lower_CI = mean_heights - critical_value*(known_pop_sd/sqrt(sample_size)),
       upper_CI = mean_heights + critical_value*(known_pop_sd/sqrt(sample_size)))
)
}
```

```
## # A tibble: 1 x 3
##   mean_heights lower_CI upper_CI
##          <dbl>    <dbl>    <dbl>
## 1         69.4     67.7     71.2
## # A tibble: 1 x 3
##   mean_heights lower_CI upper_CI
##          <dbl>    <dbl>    <dbl>
## 1         70.0     68.3     71.8
```

```
## # A tibble: 1 x 3
##   mean_heights lower_CI upper_CI
##          <dbl>    <dbl>    <dbl>
## 1         70.5     68.8     72.2
```

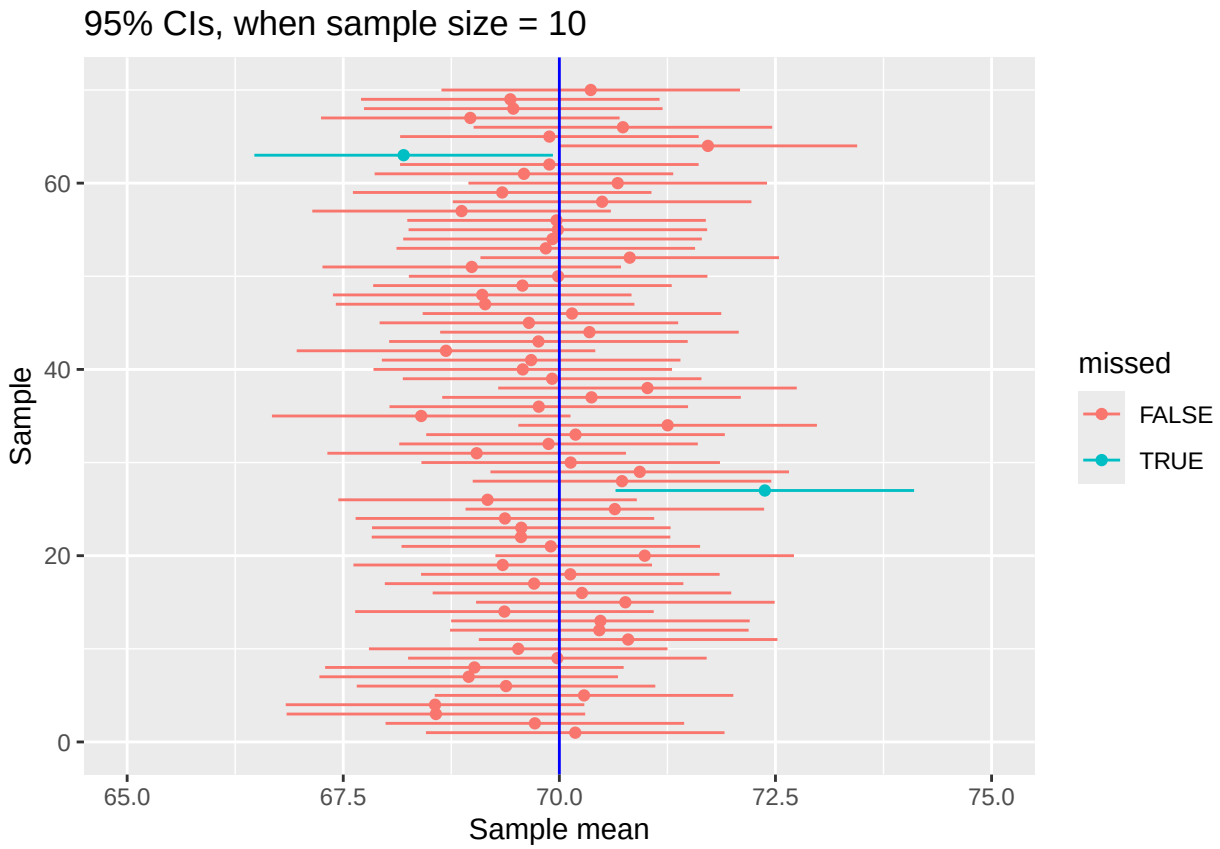
Now, let's make a plot as if we sampled for 100 CIs and see how many contain the true value for the mean. Try to understand what is going on within the code and experiment by changing the number of samples!

Plot sample size 10

```
alameda_pop <- read.csv("data/HW07_Alameda.csv")
known_pop_sd <- 2.786314


sample_size <- 10
calc_sample_stats <- function(df) {
  df %>%
    summarize(mean_heights = mean(height)) %>%
    mutate(lower_CI = mean_heights - 1.96 * known_pop_sd / sqrt(sample_size),
           upper_CI = mean_heights + 1.96 * known_pop_sd / sqrt(sample_size))

}
many.sample.stats <- replicate(100, sample_n(alameda_pop, sample_size), simplify = F) %>%
  lapply(., calc_sample_stats) %>%
  bind_rows() %>%
  mutate(sample.id = 1:n())
many.sample.stats <- many.sample.stats %>% mutate(missed = ((lower_CI > 70) & (upper_CI > 70)) |
                                                    ((lower_CI < 70) & (upper_CI < 70)))
ggplot(many.sample.stats %>% filter(sample.id < 100), aes(x = mean_heights, y = sample.id)) +
  geom_point(aes(col = missed)) +
  geom_segment(aes(x = lower_CI, xend = upper_CI, yend = sample.id, col = missed)) +
  geom_vline(xintercept = 70, col = "blue") +
  labs(y = "Sample", x = "Sample mean",
       title = paste0("95% CIs, when sample size = ", sample_size)) +
  scale_x_continuous(limits = c(65, 75)) +
  scale_y_continuous(limits = c(0, 70))
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```



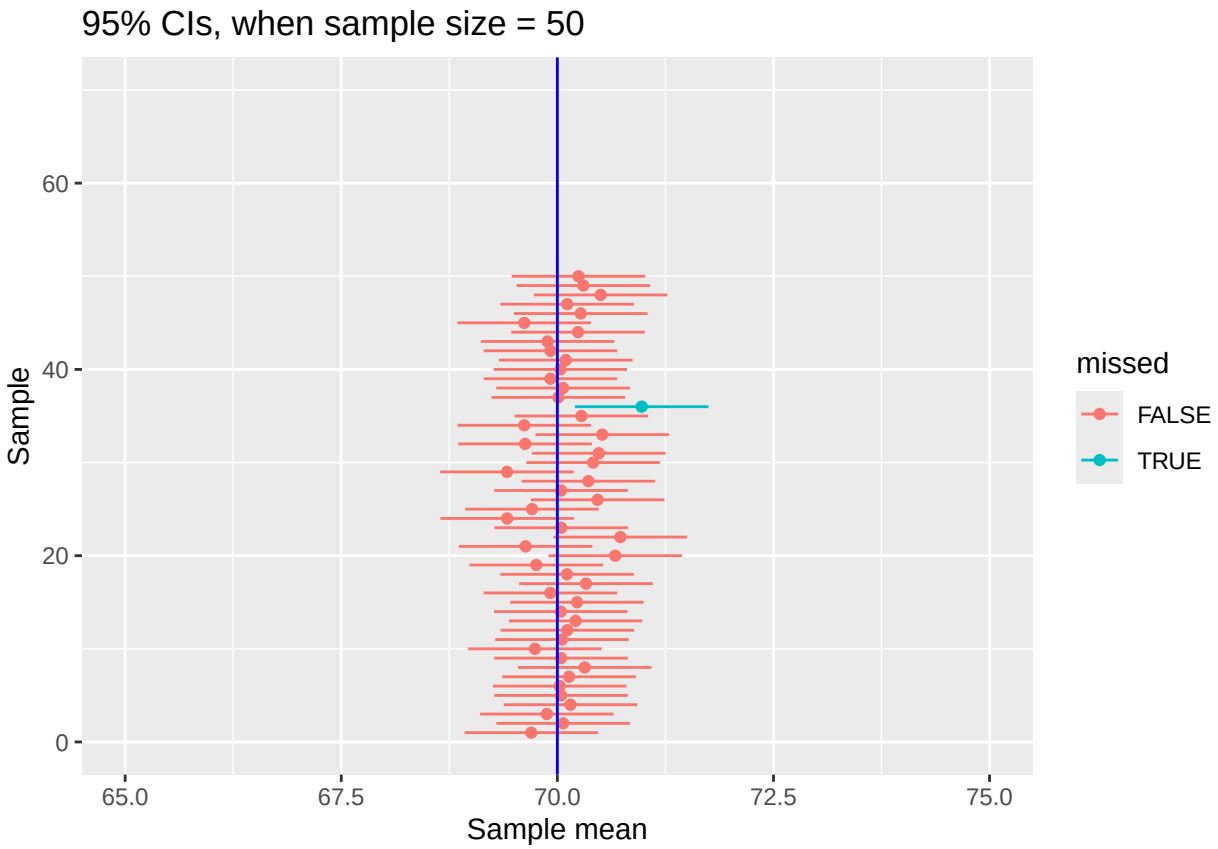
Based on this plot:

- What proportion of the confidence intervals contain the mean?

Answers will vary: 97/100 or 0.97 of the confidence intervals contain the mean.

Repeat this for a sample size of $n = 50$. In the code chunk below, generalize your code from the previous chunk to create three samples, this time of size n = 50:

```
sample_size <- 50
many.sample.stats <- replicate(50, sample_n(alameda_pop, sample_size), simplify = F) %>%
  lapply(., calc_sample_stats) %>%
  bind_rows() %>%
  mutate(sample.id = 1:n())
many.sample.stats <- many.sample.stats %>% mutate(missed = ((lower_CI > 70) & (upper_CI > 70)) |
                                                    ((lower_CI < 70) & (upper_CI < 70)))
ggplot(many.sample.stats %>% filter(sample.id < 100), aes(x = mean_heights, y = sample.id)) +
  geom_point(aes(col = missed)) +
  geom_segment(aes(x = lower_CI, xend = upper_CI, yend = sample.id, col = missed)) +
  geom_vline(xintercept = 70, col = "blue") +
  labs(y = "Sample", x = "Sample mean",
       title = paste0("95% CIs, when sample size = ", sample_size)) +
  scale_x_continuous(limits = c(65, 75)) +
  scale_y_continuous(limits = c(0, 70))
```



Once this is done, plot a sample for 100 CIs. Again, try to understand what is going on within the code and experiment by changing the number of samples!

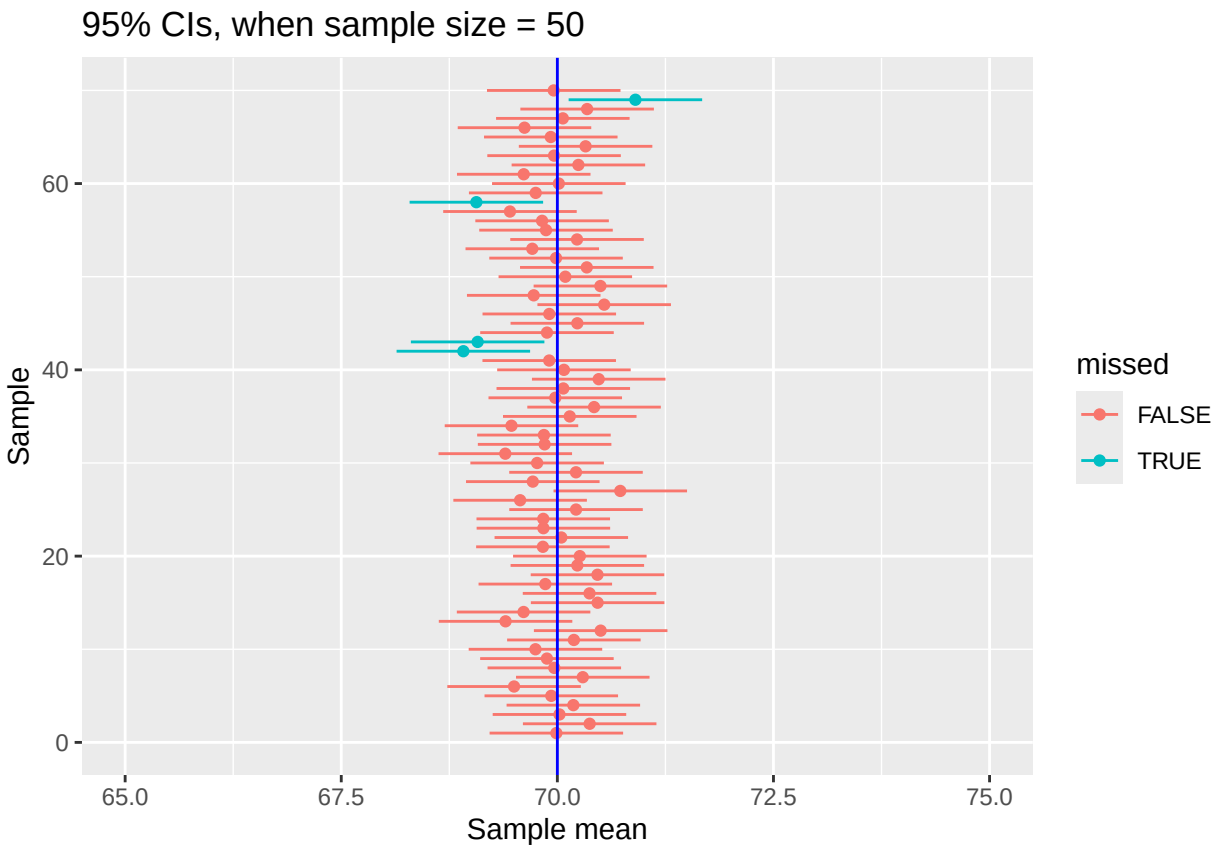
Plot sample size 50

```
alameda_pop <- read.csv("data/HW07_Alameda.csv")
known_pop_sd <- 2.786314


sample_size <- 50
calc_sample_stats <- function(df) {
  df %>%
    summarize(mean_heights = mean(height)) %>%
    mutate(lower_CI = mean_heights - 1.96 * known_pop_sd / sqrt(sample_size),
           upper_CI = mean_heights + 1.96 * known_pop_sd / sqrt(sample_size))

}
many.sample.stats <- replicate(100, sample_n(alameda_pop, sample_size), simplify = F) %>%
  lapply(., calc_sample_stats) %>%
  bind_rows() %>%
  mutate(sample.id = 1:n())
many.sample.stats <- many.sample.stats %>% mutate(missed = ((lower_CI > 70) & (upper_CI > 70)) |
                                                    ((lower_CI < 70) & (upper_CI < 70)))
ggplot(many.sample.stats %>% filter(sample.id < 100), aes(x = mean_heights, y = sample.id)) +
  geom_point(aes(col = missed)) +
  geom_segment(aes(x = lower_CI, xend = upper_CI, yend = sample.id, col = missed)) +
  geom_vline(xintercept = 70, col = "blue") +
  labs(y = "Sample", x = "Sample mean",
```

```
      title = paste0("95% CIs, when sample size = ", sample_size)) +
  scale_x_continuous(limits = c(65, 75)) +
  scale_y_continuous(limits = c(0, 70))
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```



- What proportion of the confidence intervals contain the mean?

Answers will vary: 95/100 or 0.95 of the confidence intervals contain the mean.

- How do the average widths of the CI's compare for $n = 50$ versus $n = 10$?

As n increases from 10 to 50, the width of the confidence interval gets narrower.

- What would happen to the confidence intervals if $n = 500$?

If $n = 500$, the confidence intervals would become even more narrow around the value of the sample mean.

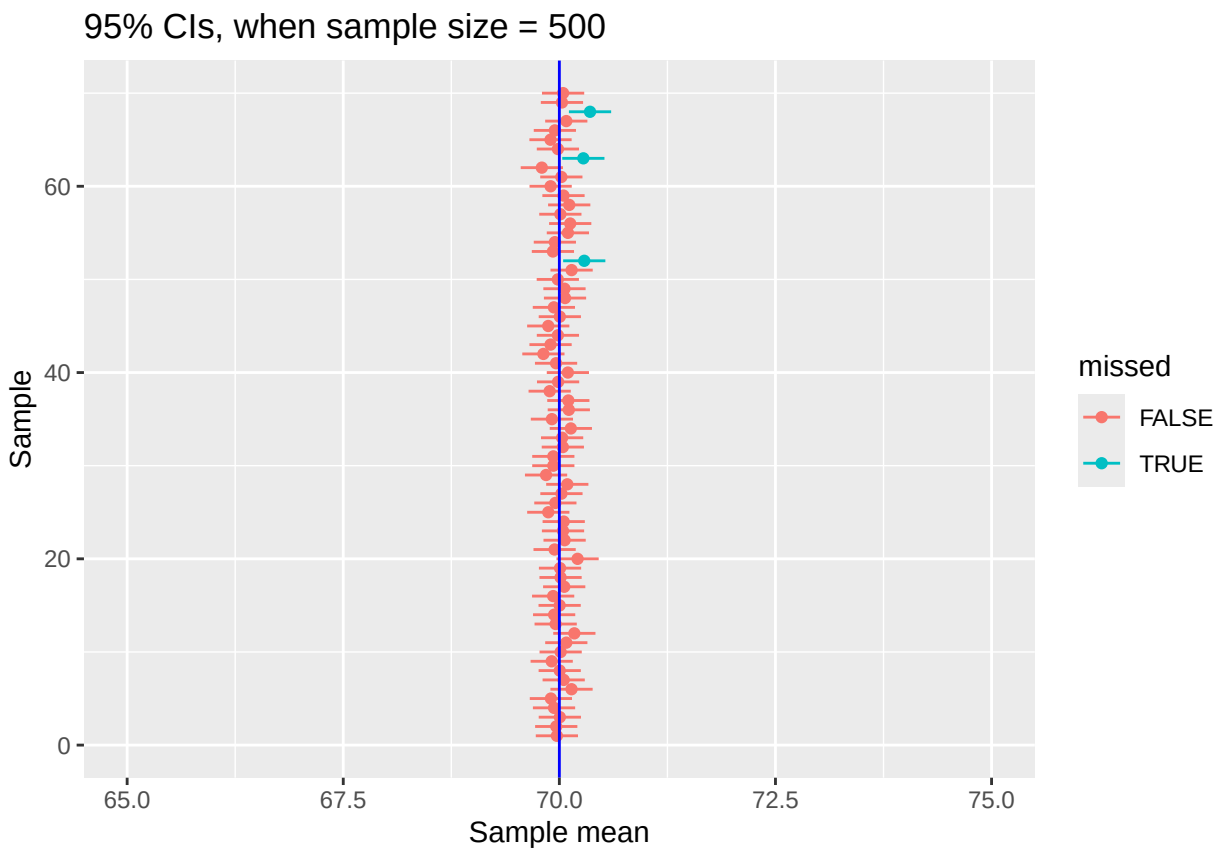
Bonus! Run this code to view what happens to the confidence intervals if $n = 500$ and if you want to simulate on your own samples of different sizes:

```
alameda_pop <- read.csv("data/HW07_Alameda.csv")
known_pop_sd <- 2.786314


sample_size <- 500
calc_sample_stats <- function(df) {
```

```
  df %>%
    summarize(mean_heights = mean(height)) %>%
    mutate(lower_CI = mean_heights - 1.96 * known_pop_sd / sqrt(sample_size),
           upper_CI = mean_heights + 1.96 * known_pop_sd / sqrt(sample_size))

}
many.sample.stats <- replicate(100, sample_n(alameda_pop, sample_size), simplify = F) %>%
  lapply(., calc_sample_stats) %>%
  bind_rows() %>%
  mutate(sample.id = 1:n())
many.sample.stats <- many.sample.stats %>% mutate(missed = ((lower_CI > 70) & (upper_CI > 70)) |
                                                    ((lower_CI < 70) & (upper_CI < 70)))
ggplot(many.sample.stats %>% filter(sample.id < 100), aes(x = mean_heights, y = sample.id)) +
  geom_point(aes(col = missed)) +
  geom_segment(aes(x = lower_CI, xend = upper_CI, yend = sample.id, col = missed)) +
  geom_vline(xintercept = 70, col = "blue") +
  labs(y = "Sample", x = "Sample mean",
       title = paste0("95% CIs, when sample size = ", sample_size)) +
  scale_x_continuous(limits = c(65, 75)) +
  scale_y_continuous(limits = c(0, 70))
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 29 rows containing missing values or values outside the scale range
## (`geom_segment()`).
```



Part 2

In this next section you will read articles from the internet about differences in p-values and confidence intervals. You will also review information about p-hacking and data dredging to bring you up-to-speed on some of the language used to talk about bad scientific practice around the misuse of p-values.

Section 1:

To do:

- Watch this 11-min Youtube video on p-hacking: https://www.youtube.com/watch?v=Gx0fAjNHb1M

(Captioned video will be linked on on Lab07 thread on Piazza when available)

- Read this Wikipedia article on data dredging: https://en.wikipedia.org/wiki/Data_dredging
- Read this Vox article about the Cornell food researcher: https://www.vox.com/science-and-health/2018/9/19/17879102/brian-wansink-cornell-food-brand-lab-retractions-jama
- Read this two-page ASA brief on statistical significance and p-values: https://www.amstat.org/asa/files/pdfs/P-ValueStatement.pdf

5. In your own words, what is p-hacking?

Manipulating your data to artificially manufacture significant p-values by choosing analysis strategies that are designed to make the p-values significant rather than the best analysis plan for the question.

6. In your own words, what is data dredging?

Performing many tests on the same data to find statistical signficance and only reporting the ones that come back significant without the context of all the tests that were insignificant.

7. One of these sources provides an example of p-hacking in epidemiology related to cancer clusters. Describe in your own words what the problem is.

A researcher has access to a bunch of demographic information about a particular town identified as a cancer cluster. In actuality, none of these variables may have any correlation with the increased risk of cancer. If you were to set a significance level of .01 then perform 100 tests on each of these 100 covariates to find its connection to cancer, it is likely that at least 1 of these tests will be reveal a significant relationship just by chance.

8. What are three practices noted in one of the articles to reduce p-hacking? Name each one and describe them in 1-2 sentences.

Pre-registration of study designs: Having scientisists commit to a design and analysis plan before the data is collected.

Open data sharing: Sharing data publicly (when possible) to allow for analyses to be repeated and checked even when the results have already been peer-reviewed.

Registered replication reports: Replicating (exactly or conceptually) already established findings to ensure and scrutinize the result.

9. One of the sources give a correction method for calculating p-values when you are going to conduct multiple tests. What is the name of the method? Write the equation for this correction.

Bonferroni Correction: Significant level / number of tests

Please watch the video here:

https://www.youtube.com/watch?reload=9&v=5OL1RqHrZQ8

With captions: https://youtu.be/hes_5Xds8_U

10. Which p-value is mentioned as leading to "Elation"?

0.001

11. How big was the "true" difference in the imaginary experiment described?

10 (or .5 sd)

12. Which measure gave a better estimate of the variability in results over multiple simulated studies?

Confidence intervals

END