Lab07: Confidence intervals and the dance of the p-values and p-hacking

Nimita Jugal Gaggar (3039677409)

07/27/2023

Run this chunk of code to load the autograder package!

Instructions

- Due date: Thursday, July 28th at 10:00pm PST with 2 hour grace period.
- Late penalty: 50% late penalty if submitted within 24 hours of due date, no marks for assignments submitted thereafter.
- Don't delete or add any `\newpage` tags.

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! You must knit correctly before submitting.*
- If your code runs off the page of the knitted PDF then you will LOSE POINTS! To avoid this, have a look at your knitted PDF and ensure all the code fits in the file (you can easily view it on Gradescope via the provided link after submitting). 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 last week's lab: 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, `L07_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/L07_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 %>% summarise(mean_height = mean(height), sd_height = sd(height) )
height_mean_sd
```

```
##   mean_height sd_height
## 1    69.97705  2.786314
```

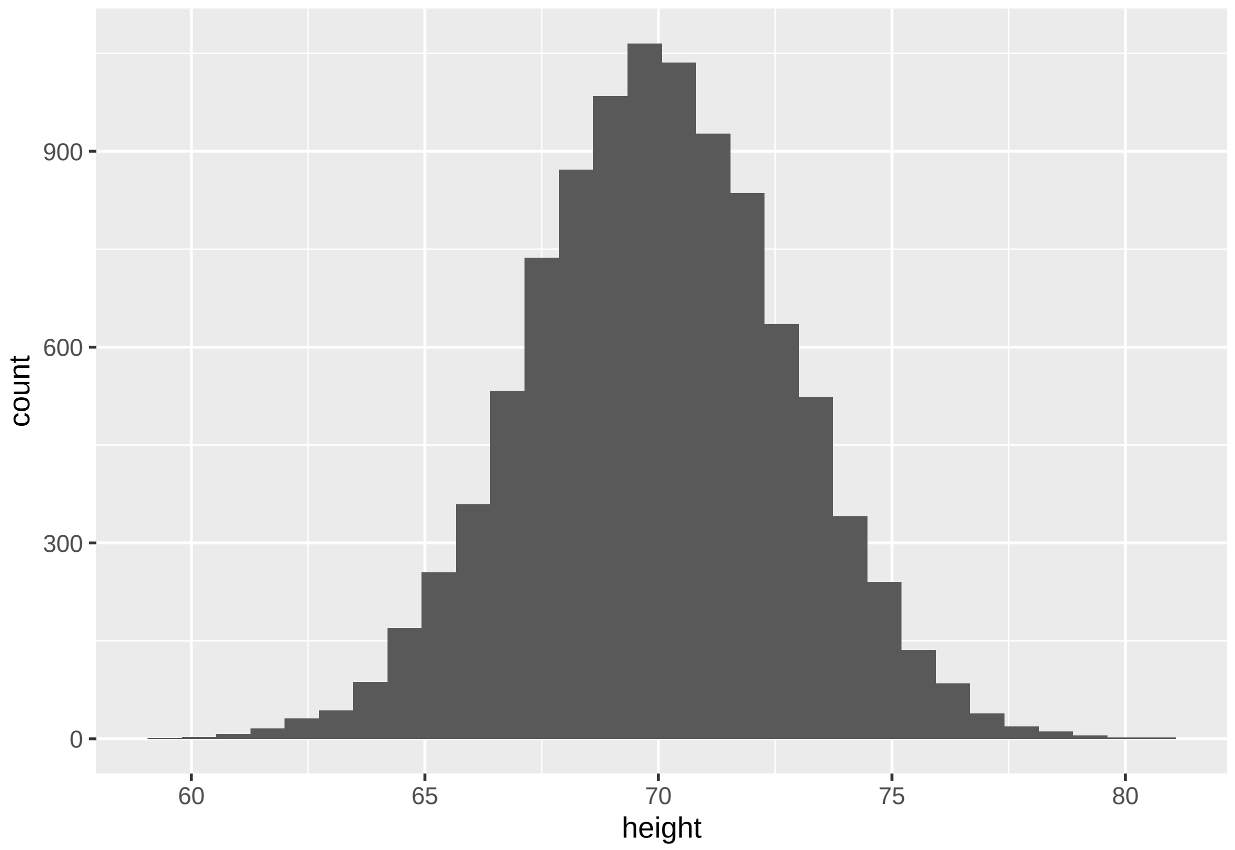
```
. = ottr::check("tests/height_mean_sd.R")
```

```
##
## 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'.
```



```r
. = 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.**

```r
known_pop_sd <- height_mean_sd $sd_height
known_pop_sd
```

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

```r
. = 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.

```
set.seed (142)

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))
)

for (i in 1:3) {sample_size <- 10
critical_value <- 1.96
size_10 <- sample_n(alameda_pop, sample_size, replace = FALSE)
```

```r
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))
)
print(p8)}
```

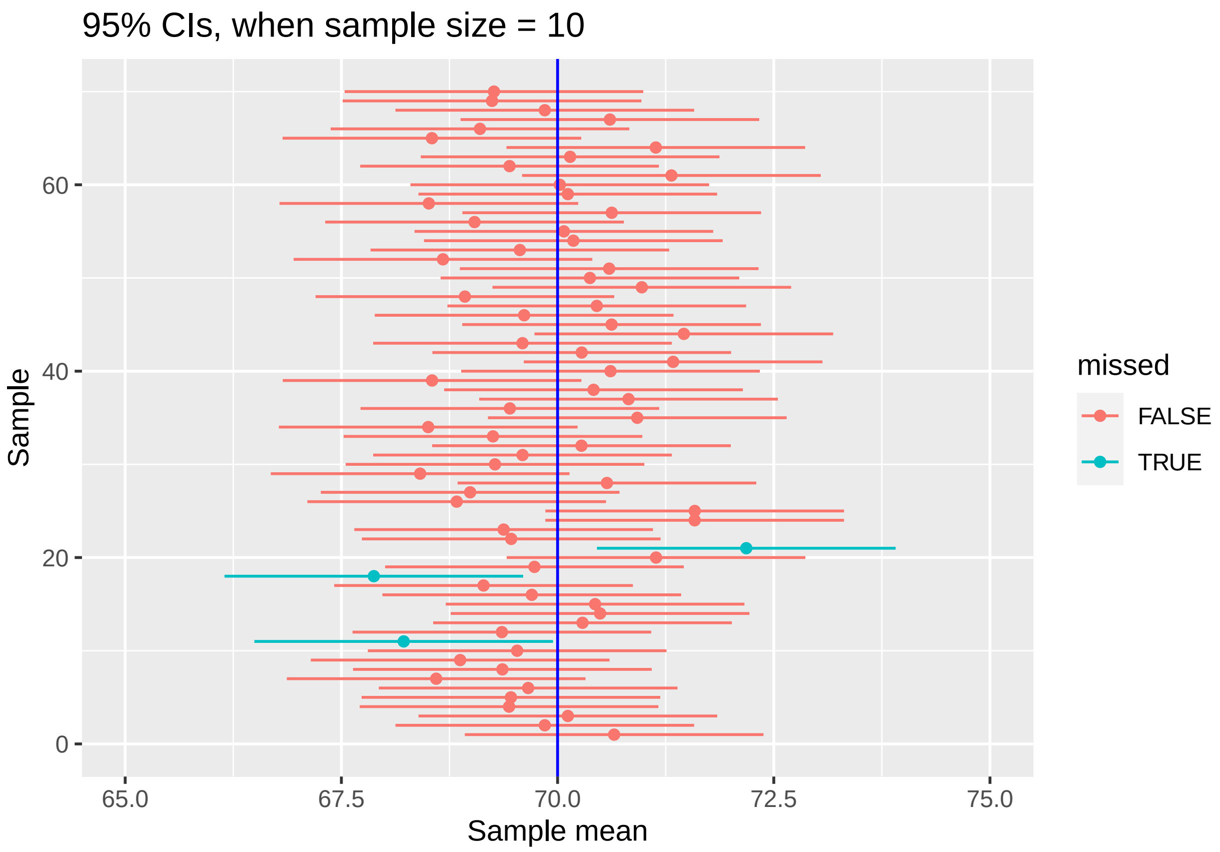
```
##   mean_heights lower_CI upper_CI
## 1     71.01983 69.29285  72.7468
##   mean_heights lower_CI upper_CI
## 1     70.25223 68.52525  71.9792
##   mean_heights lower_CI upper_CI
## 1     69.79864 68.07167 71.52562
```

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

```r
#read in data
alameda_pop <- read.csv("data/L07_Alameda.csv")
known_pop_sd <- 2.786314

#set sample size
sample_size <- 10

#make a function to calculate 95% CI for pop mean
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))

}

#excecute above function 100 times and glue together to make a new data frame
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 (‘geom_point()‘).
## Warning: Removed 29 rows containing missing values (`geom_segment()`).
```



Based on this plot:

- What proportion of the confidence intervals contain the mean?

95% of the confidence intervals contains the true pop 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:

```r
sample_size <- 50
critical_value <- 1.96
size_50 <- sample_n(alameda_pop, sample_size, replace = FALSE)
p9 <- 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))
)

for (i in 1:3) {sample_size <- 10
critical_value <- 1.96
size_50 <- sample_n(alameda_pop, sample_size, replace = FALSE)
p9 <- size_50 %>% 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))
)
print(p9)}
```

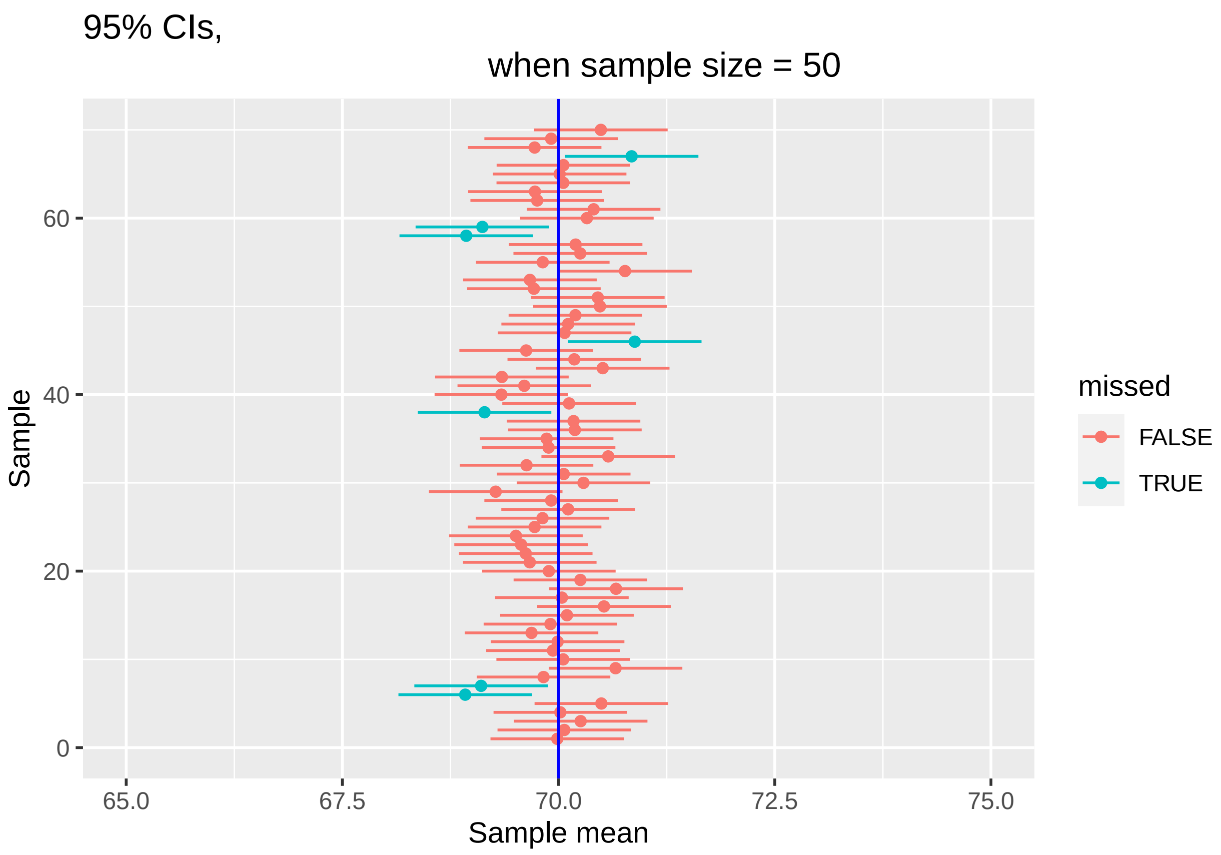
```
##   mean_heights lower_CI upper_CI
## 1      69.7457 68.01872 71.47267
##   mean_heights lower_CI upper_CI
## 1     68.49491 66.76793 70.22188
##   mean_heights lower_CI upper_CI
## 1     70.81219 69.08521 72.53917
```

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

```r
alameda_pop <- read.csv("data/L07_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 (‘geom_point()‘).

## Warning: Removed 29 rows containing missing values (‘geom_segment()‘).
```



- What proportion of the confidence intervals contain the mean?

97 % of the confidence intervals contain the mean.

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

The average widths of the CI's for $n = 50$ is narrower versus $n = 10$

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

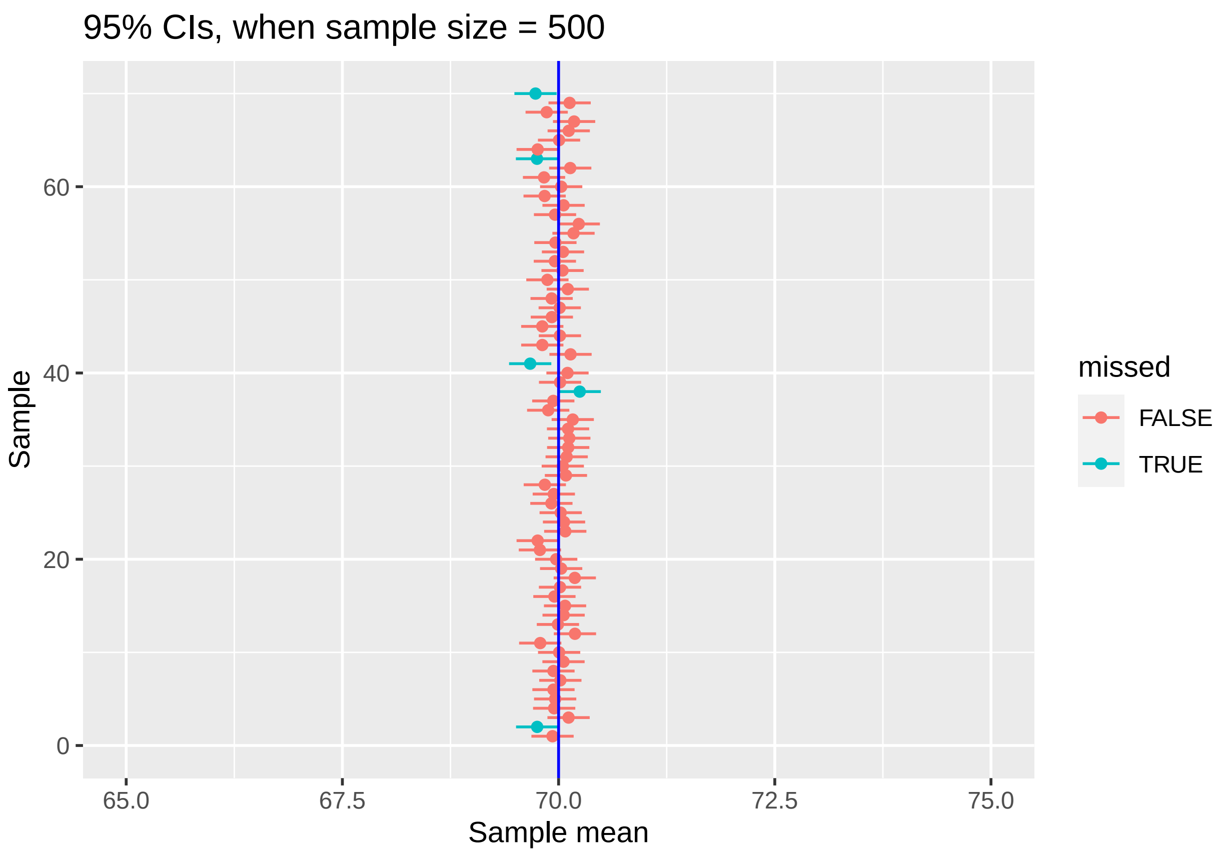
The average width of the CI will be narrower.

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/L07_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 (‘geom_point()‘).
```

```
## Warning: Removed 29 rows containing missing values (‘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?

p-hacking refers to the practice of manipulating or analyzing data in a way that increases the likelihood of obtaining statistically significant p-values that are below the threshold (often set at 0.05).

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

Data dredging is the improper use of data analysis to identify patterns in data that can be presented as statistically significant, thereby substantially increasing the risk of false positives while understating it.

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.

Another example is a town with a cancer cluster but no clear explanation. They have access to a lot of demographic data on the town and surrounding area, including hundreds or thousands of uncorrelated factors. Even if all these variables are independent of the cancer incidence rate, at least one is expected to correlate significantly with the cancer rate in the area. This suggests a hypothesis, but more testing using the same factors and data from a different place is needed to confirm. If hundreds or thousands of hypotheses with mutually relatively uncorrelated independent variables are tested, many null hypotheses will have p-values less than 0.01.

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. 1. Preregistration of study designs: Preregistration means that scientists publicly commit to an experiment's design before they start collecting data. This makes it much harder to cherry-pick results.

2. Open data sharing: Increasingly, scientists are calling on their colleagues to make all the data from their experiments available for anyone to scrutinize (there are exceptions, of course, for particularly sensitive information). This ensures that shoddy research that makes it through peer review can still be double-checked.
3. Registered replication reports: Scientists are hungry to see if previously reporting findings in the academic literature hold up under more intense scrutiny. There are many efforts underway to replicate (exactly or conceptually) research findings with rigor.

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.

The correction method for calculating p-values when conducting multiple tests is called the "Bonferroni correction."

Equation : Corrected significance level (alpha_corrected) = alpha / Number of tests

Where:

alpha = desired significance level for individual tests (commonly used as 0.05). Number of tests refers to the total number of statistical tests conducted.

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"?

p-value 0.001 is mentioned as leading to Elation

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

In the imaginary experiment described, the "true" difference was assumed to be 10 points on a well-being score.This difference was equivalent to half a standard deviation, which is represented as Cohen's delta of 0.5 or standardized mean difference of 0.5.

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

Confidence interval measure gave a better estimate of the variability in results over multiple simulated studies.

Submission

For assignments in this class, you'll be submitting using the **Terminal** tab in the pane below. In order for the submission to work properly, make sure that:

1. Any image files you add that are needed to knit the file are in the `src` folder and file paths are specified accordingly.
2. You **have not changed the file name** of the assignment.
3. The file knits properly.

Once you have checked these items, you can proceed to submit your assignment.

1. Click on the **Terminal** tab in the pane below.
2. Copy-paste the following line of code into the terminal and press enter.

cd; cd ph142-su23/lab/lab07; python3 turn_in.py

3. Follow the prompts to enter your Gradescope username and password.
4. If the submission is successful, you should see "Submission successful!" appear as the output. **Check your submission on the Gradescope website to ensure that the autograder worked properly and you received credit for your correct answers. If you think the autograder is incorrectly grading your work, please post on piazza!**
5. If the submission fails, try to diagnose the issue using the error messages–if you have problems, post on Piazza under the post "Datahub Issues".

The late policy will be strictly enforced, **no matter the reason**, including submission issues, so be sure to submit early enough to have time to diagnose issues if problems arise.

END