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

Nimita Jugal Gaggar

07/27/2023

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
```

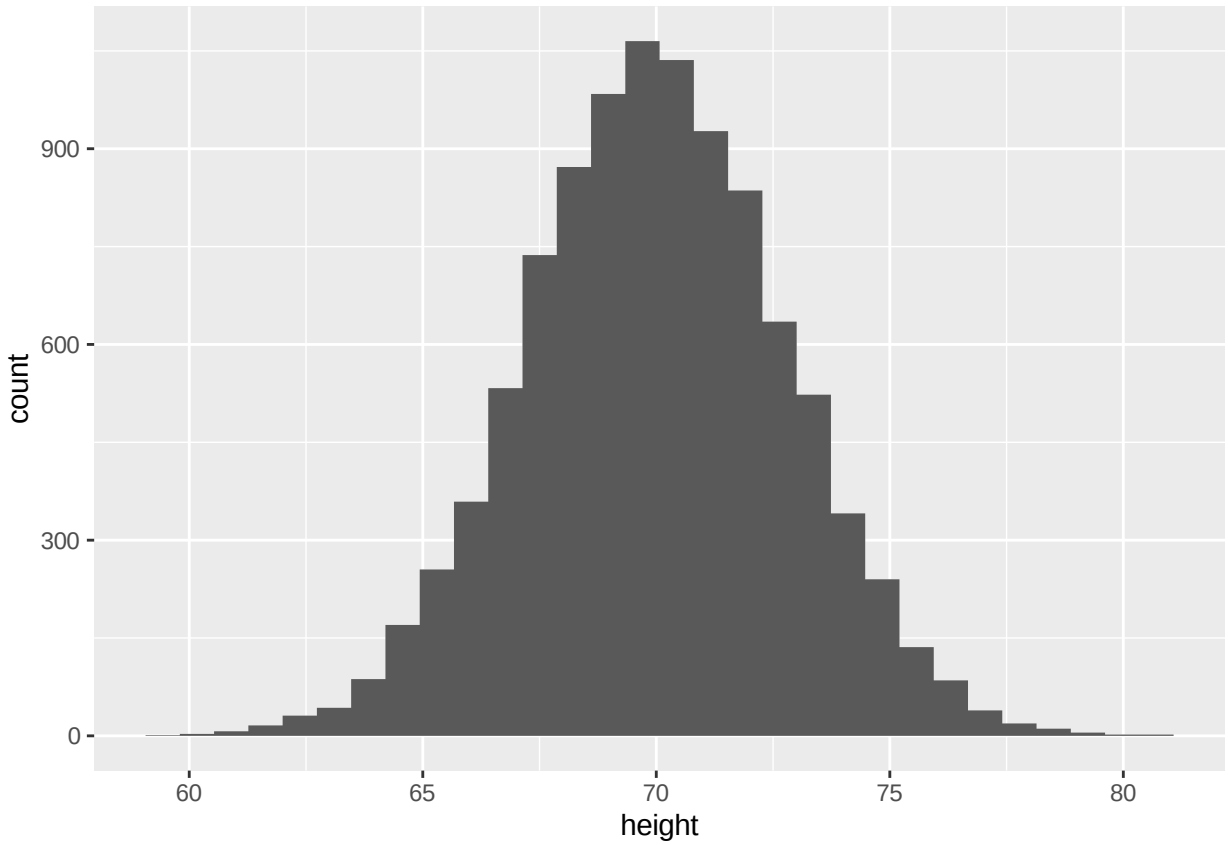
```
. = ottr::check("tests/height_mean_sd.R")
```

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




```
. = 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 <- height_mean_sd $sd_height
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.

```
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)
```

```
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
```

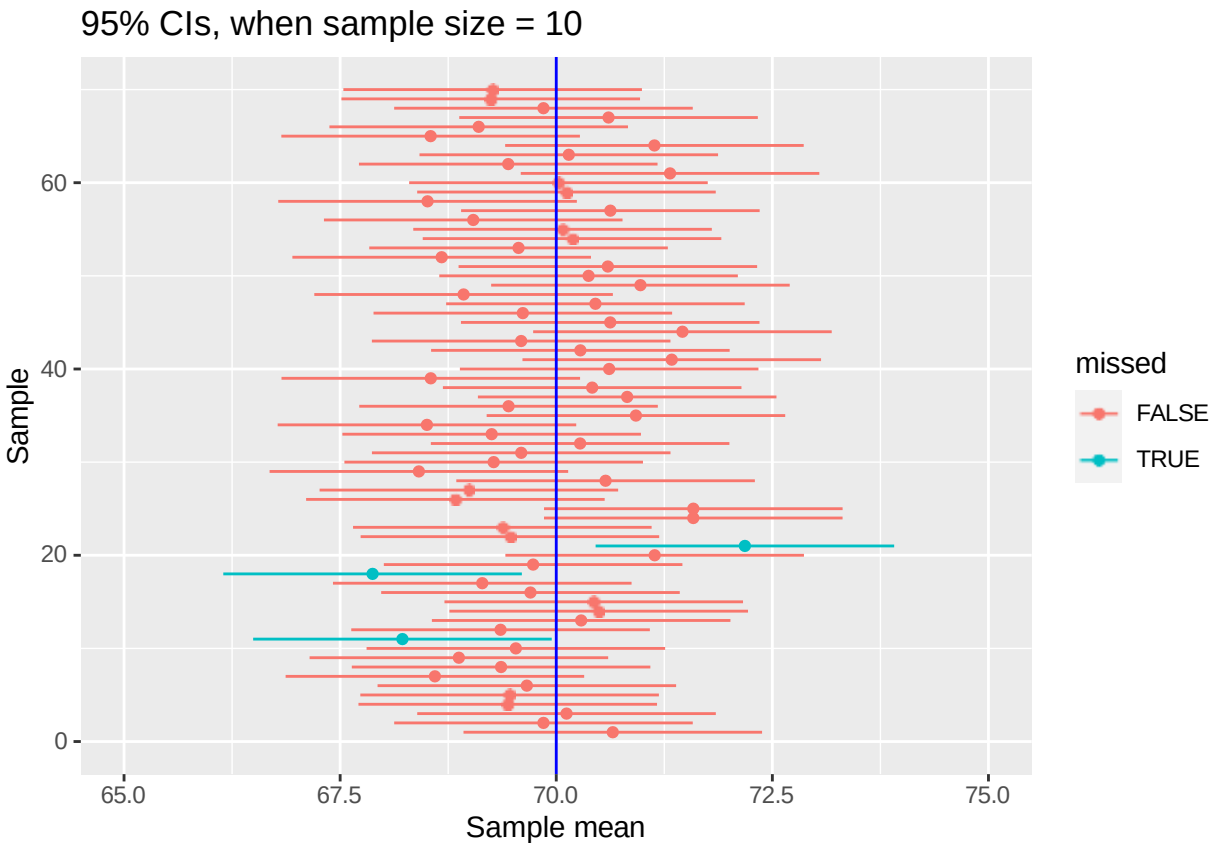
Now, let's make a plot as if we sampled for 100 CIs and see how many contain the true value for the mean. Try to understand what is going on within the code and experiment by changing the number of samples!

Plot sample size 10

```
#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:

```
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
```

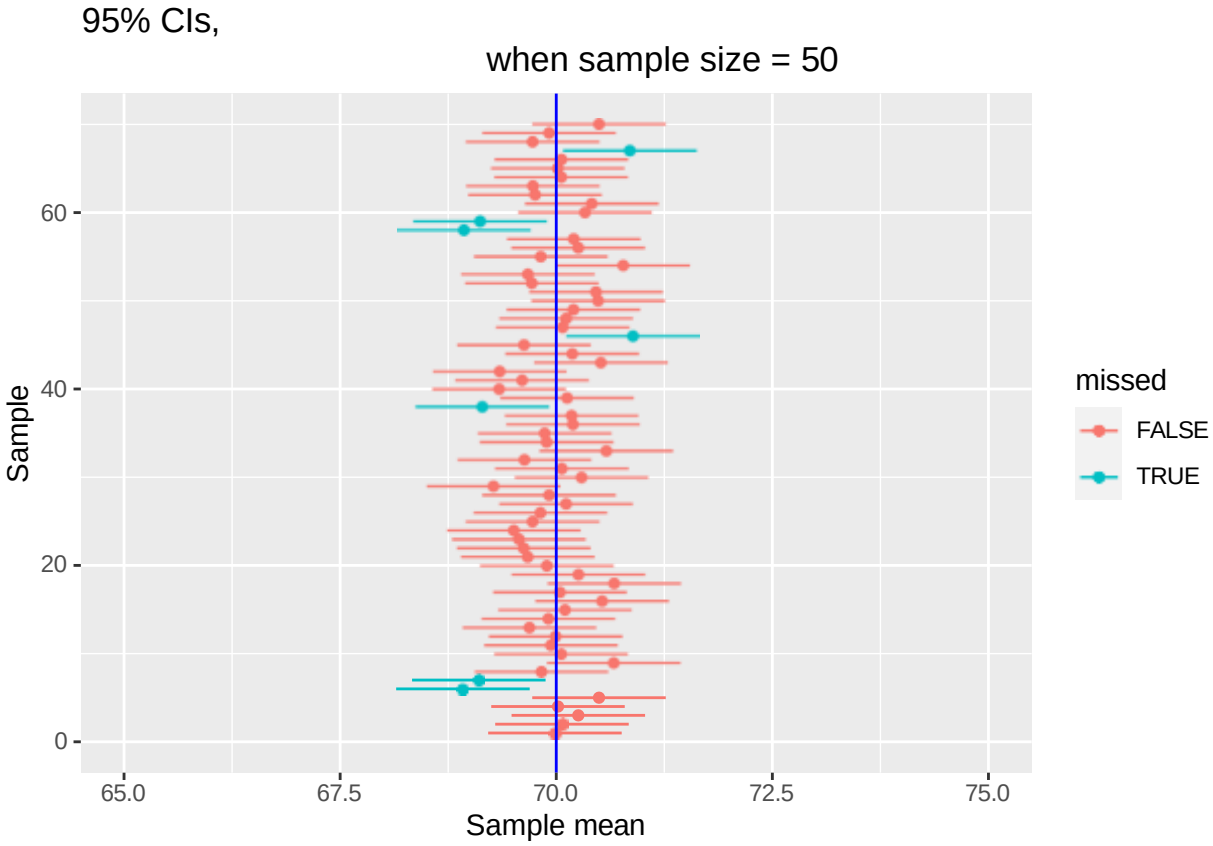
Once this is done, plot a sample for 100 CIs. Again, try to understand what is going on within the code and experiment by changing the number of samples!

Plot sample size 50

```
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()').