# Math 300 Lesson 26 Notes

Interpreting Confidence Intervals

YOUR NAME HERE

July, 2022

## Contents



### Objectives

1. Correctly interpret a confidence interval to include identifying incorrect statements.
2. Explain the factors that impact the width of a confidence interval.

### Reading

Chapter 8.5

### Lesson

There are no learning checks for this lesson.

- The interpretation of the confidence interval is subtle because the 95% coverage relates to process. If we were to repeat the data collection process and confidence interval construction, the coverage relates to how often these intervals contain the true population parameter. In practice we don't know the true population parameter and we don't repeat the entire process.
- We will use a simulation to help us understand the confidence interval. We will use a proportion instead of a mean. Pay attention to the code used with the `infer` package.

---

#### Libraries

```
library(tidyverse)
library(moderndive)
library(infer)
```

### Understanding confidence intervals

The interpretation of a confidence interval relates to the entire process of collecting data and building a confidence interval. In practice we do this process only once. However, in a simulation we can repeat the process. We will do this in this lesson.

Let's use the bin of red and white balls. The population is a set of 2400 balls where 900 are red and 1500 are white. We already know more than we would in practice since we have a census.

```
# The population data
head(bowl)
```

```
## # A tibble: 6 x 2
##   ball_ID color
##     <int> <chr>
## 1       1 white
## 2       2 white
## 3       3 white
## 4       4 red
## 5       5 white
## 6       6 white
```

Let's find the population summary numbers.

```
bowl %>%
  summarize(red=mean(color=="red"),total=n(),num_red=sum(color=="red"))
```

```
## # A tibble: 1 x 3
##     red total num_red
##   <dbl> <int>   <int>
## 1 0.375  2400     900
```

This information will give us knowledge about the population that is not known in practice.

Let's take a sample of size 100 from the population and summarize that sample.

```
set.seed(911)
sample1<-bowl %>%
    rep_sample_n(size = 100, reps = 1, replace = FALSE)
```

```
sample1 %>%
  summarize(red=mean(color=="red"),total=n(),num_red=sum(color=="red"))
```

```
## # A tibble: 1 x 4
##   replicate   red total num_red
##       <int> <dbl> <int>   <int>
## 1         1  0.42   100      42
```

In this sample we see that we have 42 red balls out of 100, for a proportion of 0.42. Why is this not .375? Let's construct a 90% confidence interval. Pay attention to the options used in each function call.

```
(bootstrap_dist1<-sample1 %>%
      specify(formula = color ~ NULL, success = "red") %>%
      generate(reps = 1000, type = "bootstrap") %>%
      calculate(stat = "prop"))
```

```
## Response: color (factor)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  0.49
##  2         2  0.54
##  3         3  0.35
##  4         4  0.35
##  5         5  0.43
##  6         6  0.45
##  7         7  0.42
##  8         8  0.39
##  9         9  0.46
## 10        10  0.37
## # ... with 990 more rows
```

And now the confidence interval.

```
percentile_ci_1 <- bootstrap_dist1 %>%
  get_confidence_interval(level = 0.90, type = "percentile")
```

```
percentile_ci_1
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     0.34      0.5
```

- Interpret this confidence interval.

**Answer:**

- What is wrong with the following statement?

*There is a 90% probability that the true population proportion of red balls is between 0.34 and 0.5*

**Answer:**

---

Notice that the above confidence interval captured the true mean. Repeat the process again.

```
# Complete the code
# bowl %>%
#     rep_sample_n(size = _______, reps = 1, replace = __________) %>%
#     specify(formula = color ~ NULL, success = "________") %>%
#     generate(reps = 1000, type = "__________") %>%
#     calculate(stat = "________") %>%
#     get_confidence_interval(level = _________, type = "____________")
```

Notice that this interval also included the true value but it was a different interval. Again, let's repeat the process.

```
# Add code here
```

We could just keep doing a cut and paste, but let's write a function that will automate this process. You are not expected to generate this code but it is presented if you are interested in learning about more sophisticated coding in R.

First we write a function that builds a confidence interval using the bootstrap method for a given random sample.

```
# Function to find a confidence interval
  ci_pipeline <- function(sample_data) {
    sample_data %>%
      specify(formula = color ~ NULL, success = "red") %>%
      generate(reps = 1000, type = "bootstrap") %>%
      calculate(stat = "prop") %>%
      get_confidence_interval(level = 0.90, type = "percentile") %>%
      mutate(mid_point=(upper_ci+lower_ci)/2)
  }
```

Now we just need to repeat the sampling process and apply our function to each sample.

```
set.seed(9073)
bowl %>%
  rep_sample_n(size = 100, reps = 100, replace = FALSE) %>%
  group_by(replicate) %>%
  nest() %>%
  mutate(bootstraps = map(data, ci_pipeline)) %>%
  unnest(bootstraps) %>%
  mutate(captured = lower_ci <= 0.375 & 0.375 <= upper_ci) -> sim_cis
```

```
head(sim_cis)
```

```
## # A tibble: 6 x 6
## # Groups:   replicate [6]
##   replicate data               lower_ci upper_ci mid_point captured
##       <int> <list>                <dbl>    <dbl>     <dbl> <lgl>
## 1         1 <tibble [100 x 2]>     0.29     0.45     0.37  TRUE
## 2         2 <tibble [100 x 2]>     0.27     0.42     0.345 TRUE
## 3         3 <tibble [100 x 2]>     0.31     0.47     0.39  TRUE
## 4         4 <tibble [100 x 2]>     0.29     0.45     0.37  TRUE
## 5         5 <tibble [100 x 2]>     0.23     0.39     0.31  TRUE
## 6         6 <tibble [100 x 2]>     0.32     0.48     0.4   TRUE
```

We can find out how many of the confidence intervals included the true proportion of red balls. It should be around 90.

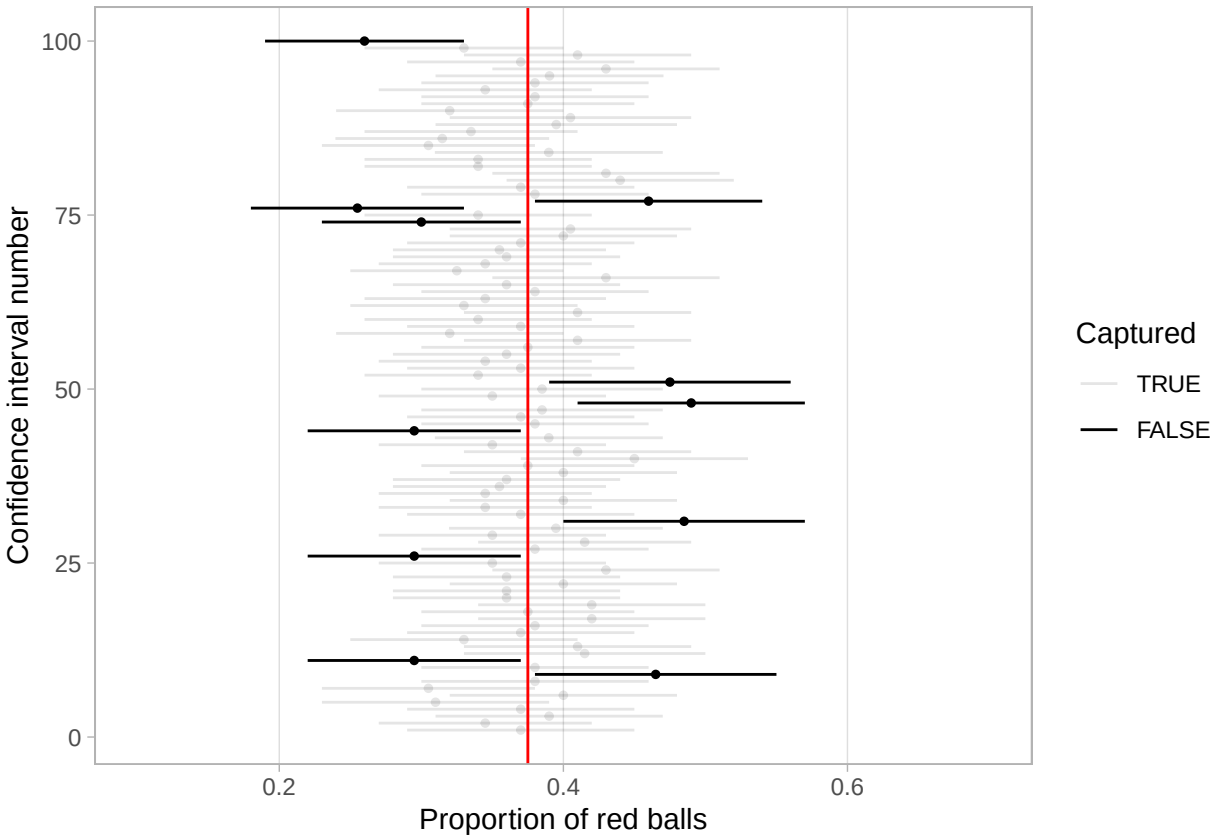
```
sim_cis %>%
  ungroup() %>%
  summarize(coverage=sum(captured))
```

```
## # A tibble: 1 x 1
##   coverage
##      <int>
## 1       89
```

This means that 89 out of the 100 simulated confidence intervals included the true population proportion. Why is it not 90?

Let's plot all the intervals.

```
ggplot(sim_cis) +
  geom_segment(aes(
    y = replicate, yend = replicate, x = lower_ci, xend = upper_ci,
    alpha = factor(captured, levels = c("TRUE", "FALSE"))
  )) +
  geom_point(
    aes(
      x = mid_point, y = replicate,
      alpha = factor(captured, levels = c("TRUE", "FALSE"))
    ),
    show.legend = FALSE, size = 1
  ) +
  labs(
    x = expression("Proportion of red balls"), y = "Confidence interval number",
    alpha = "Captured"
  ) +
  geom_vline(xintercept = 0.375, color = "red") +
  coord_cartesian(xlim = c(0.1, 0.7)) +
  theme_light() +
  theme(
    panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank(),
    panel.grid.minor.x = element_blank()
  )
```




- Interpret this plot in terms of a confidence interval.

---

What happens to width of a confidence interval if

- the sample size is larger?

**Answer:**

- the confidence level is larger?

**Answer:**

---

## Documenting software

- File creation date: 2022-07-06
- R version 4.1.3 (2022-03-10)
- `tidyverse` package version: 1.3.1
- `moderndive` package version: 0.5.4
- `infer` package version: 1.0.2