Problem Set 06: Confidence Intervals

Professor Bradley Warner

June, 2022

Documentation: None

Introduction

In this problem set we will use a small **sample** of data from the General Social Survey. The survey is designed to monitor changes in both social characteristics and attitudes. You will work with a **sample** from one neighborhood. The full neighborhood of **ALL individuals** is the population. For this problem set we do **not** know the **true population parameters** for any of the variables, because we do not have data on every person in the neighborhood. This is different from problem set 5 as we are using a sample of data.

Setup

First load the necessary packages

```
# Recall that loading the tidyverse "umbrella" package loads ggplot2, dplyr, and
# readr all at once. Feel free to load these packages any way you choose.
library(tidyverse)
library(moderndive)
library(infer)
```

Next load the data set, from where it is stored on the web:

```
gss_sample <- read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vSypSoDCMH2N76Vo2dZRPkw2q3t1mbv
```

Confidence intervals from a bootstrap resample

Step 1: Take 1000 bootstrap resamples

The following code tells R to take 1000 bootstrap resamples from the `gss_sample` data. You can set the seed to whatever value you like!

```
set.seed(42)

boot_samp_1000 <- gss_sample %>%
  rep_sample_n(size = 100, reps = 1000, replace = TRUE)
```

Note a few important details about the `rep_sample_n` function, and bootstrap sampling in general:

- `size = 100` tells R that each bootstrap resample we take has 100 cases… the size of the original sample
- `reps = 1000` tells R to take 1000 bootstrap resamples (each of size 100).
- The `replace = TRUE` argument tells R that in each bootstrap resample, we can include a row from `gss_sample` multiple times. So if for instance, respondent # 12 is the first random resample taken here, respondent 12 is still available to be resampled **again** at random. Thus, some people may appear **multiple times** in our bootstrap resample, and some people from the original data set may not appear at all.
- We save the results in a data frame `boot_samp_1000`.

Exercise 1

How many rows does `boot_samp_1000` have? **Why?**

Answer: 100,000. Because we took 1000 samples of 100 cases each, which is 100,000 = One hundred thousand.

Step 2: Calculate the bootstrap statistic

Let's say we want to use the bootstrap resample that we just generated to calculate a confidence interval for the population mean μ_{tv} of `tvhours`. To do so, we need to know the sample mean $\bar{x}$ of `tvhours` **for each of the 1000 bootstrap resamples**. In this case, the sample mean $\bar{x}$ of `tvhours` for **each bootstrap resample** is our **BOOTSTRAP STATISTIC**. We can calculate that with two lines of code, like so:

```
boot_distrib_tv <- boot_samp_1000 %>%
  group_by(replicate) %>%
  summarize(boot_stat = mean(tvhours))
```

Note that:

- The `group_by()` argument tells R to take the sample mean of `tvhours` **separately** for each different `replicate` in the bootstrap resample.
- We put the sample mean for each bootstrap resample in a column called `boot_stat`

  This is the bootstrap distribution for the mean of `tvhours`!

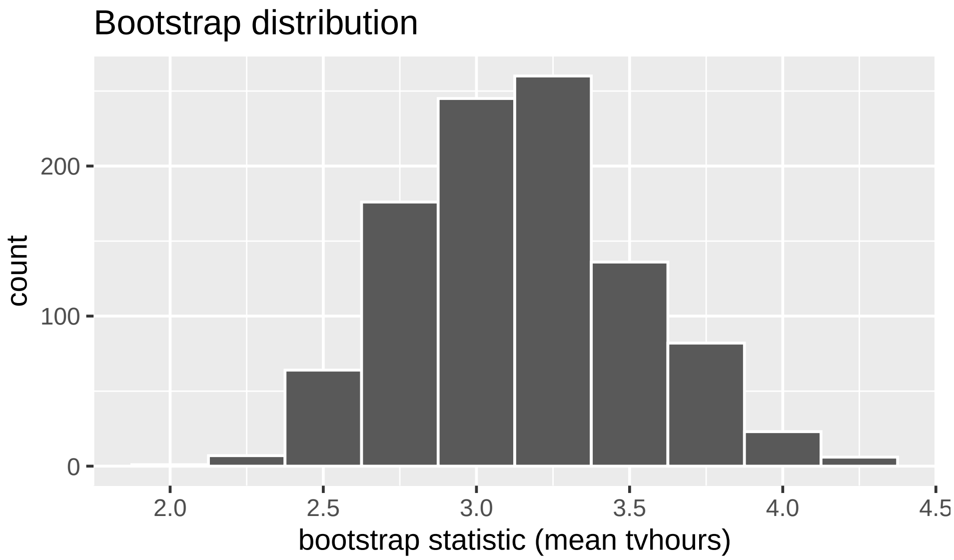
Take a look at the `boot_distrib_tv` we just created in RStudio's data viewer.

Exercise 2

How many values of the bootstrap statistic `boot_stat` are there? Please explain **why** there are this many values of the bootstrap statistic.

Answer: there are 1000 boot_stat values, because we took 1000 bootstrap resamples, and calculated the sample mean for each sample. This equates to 1000 bootstrap statistics.

Visualizing the bootstrap distribution The bootstrap distribution is shown in the figure below. This is a histogram of the `boot_stat` values from `boot_distrib_tv`.



Step 3: CI from a bootstrap resample

a) CI using the standard error We can now use the bootstrap distribution for the sample mean `tvhours` $\bar{x}$ to calculate a 95% confidence interval for the population mean `tvhours` μ_{tv}, using the standard error method. This method assumes the bootstrap distribution is approximately normal. This interval is calculated as

$$\text{mean} \pm 1.96 \cdot SD$$

The 1.96 comes from the standard normal distribution.

```
qnorm(.975)
```

```
## [1] 1.959964
```

- the mean here would be the mean of the bootstrap distribution
- the SD here is the standard deviation of the bootstrap distribution, which recall has a special name: the **standard error**.

We can thus apply the standard error method, like so:

```
boot_distrib_tv %>%
  summarize(mean = mean(boot_stat),
            se = sd(boot_stat),
            lower_ci = mean - (qnorm(.975) * se),
            upper_ci = mean + (qnorm(.975) * se))
```

```
## # A tibble: 1 x 4
##    mean    se lower_ci upper_ci
##   <dbl> <dbl>    <dbl>    <dbl>
## 1  3.14 0.365     2.43     3.86
```

b) CI using percentile method You can also calculate a 95% confidence interval using the percentile method. The logic goes like this:

Since our bootstrap resample had 1000 values of `boot_stat`:

- 950 of the `boot_stat` values fall **inside** this 95% confidence interval, i.e. 95%
- 25 values fall **below** it. i.e. the lower 2.5%
- 25 values fall **above** it. i.e. the higher 2.5%

totaling 100%. We can use **the quantiles** of the bootstrap distribution to find these values like so:

```
prct_ci_tv <- boot_distrib_tv %>%
  summarize(lower_ci = quantile(boot_stat, 0.025),
            upper_ci = quantile(boot_stat, 0.975))

prct_ci_tv
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     2.51     3.89
```

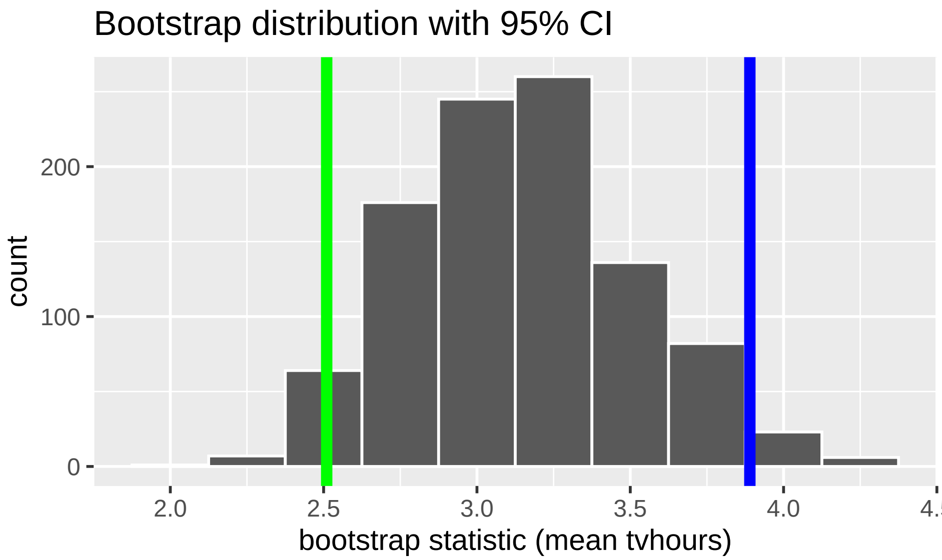
This method

- Asks `R` to identify the 0.025 quantile of the bootstrap sample means... this is the value **below** which **2.5% of the values of `boot_stat`** fall (or 25 cases in this example... 25/1000 = 0.025)
- Asks `R` to identify the 0.975 quantile for the bootstrap sample means... this is the value **above** which the other **2.5% of the values of `boot_stat`** fall (or 25 cases in this example 975/1000 = 0.975)
- The middle 95% of the values fall between these two quantiles

Based on these results, we are 95% confident that the **true mean hours of TV watched** μ_{tv} **in the population** is between the upper and lower CI we just calculated.

Visualizing the Confidence interval

The bootstrap distribution and the 95% confidence intervals we just calculated are shown in the figure below. This is a histogram of the `boot_stat` values from `boot_distrib_tv`. The green line is the lower bound of the 95% CI, and the blue line is the upper bound. 950 of the 1000 bootstrap resamples had a mean for `tvhours` that fell **between** the green and blue lines...25 of the samples had a mean above the blue line, and 25 of the samples had a mean below the green line.



Exercise 3

If we calculated a **90% confidence interval** for the mean of `tvhours` using this same bootstrap resample and the percentile method, roughly how many of the 1000 values of `tv_mean` would fall between the green and blue lines?

Answer: 900

Exercise 4

Use the bootstrap resampling distribution for `tvhours` generated above (`boot_distrib_tv`) and the **percentile** method to calculate a 99% confidence interval for the mean `tvhours`.

```
boot_distrib_tv %>%
  summarize(lower_ci = quantile(boot_stat, 0.005),
            upper_ci = quantile(boot_stat, 0.995))
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     2.32     4.13
```

Exercise 5

Which confidence interval is **WIDER**: the 95% confidence interval or the 99% confidence interval for the population mean `tvhours` μ_{tv}? Why is this the case? Answer in terms of the trade-off between confidence level and confidence interval width.

Answer: Conceptually, the 99% CI is wider, because we are casting a wider net. In other words, to have a higher degree of certainty we captured the population mean μ_{tv}, we need a wider range of values contained in our confidence interval. Also, more of the boot_stat values are included in a 99% CI (990) as compared to a 95% CI (950)

Finding Confidence Intervals Using `Infer`

As we learning in the reading, the `infer` package provides a way to find confidence intervals. We used this package in many of our in-class notes.

Exercise 6

Use the `infer` package and the the functions `specify()`, `generate()`, and `calculate()` to generate a bootstrap distribution for the variable `race`. Set the seed to 504. Success is `POC`. Use 1000 bootstrap replications.

Answer:

```
set.seed(504)
infer_boot<-gss_sample %>%
  specify(formula=race~NULL,succes="POC") %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat="prop")
```

```
head(infer_boot)
```

```
## Response: race (factor)
## # A tibble: 6 x 2
##   replicate  stat
##       <int> <dbl>
## 1         1  0.19
## 2         2  0.33
## 3         3  0.26
## 4         4  0.21
## 5         5  0.21
## 6         6  0.16
```

Exercise 7

Calculate a 90% confidence interval for the proportion of respondents that identify as `POC` using the **standard error** method.

Answer:

```
gss_sample %>%
  summarize(prob=mean(race=="POC"))
```

```
## # A tibble: 1 x 1
##    prob
##   <dbl>
## 1  0.24
```

```
get_ci(infer_boot,level=.90,type="se",point_estimate = 0.24)
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    0.168    0.312
```

Exercise 8

Calculate a 90% confidence interval for the proportion of respondents that identify as `POC` using the **percentile** method.

Answer:

```r
get_ci(infer_boot,level=.90,type="percentile")
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     0.17     0.31
```

Exercise 9

Interpret the 90% **percentile** confidence interval.

Answer:

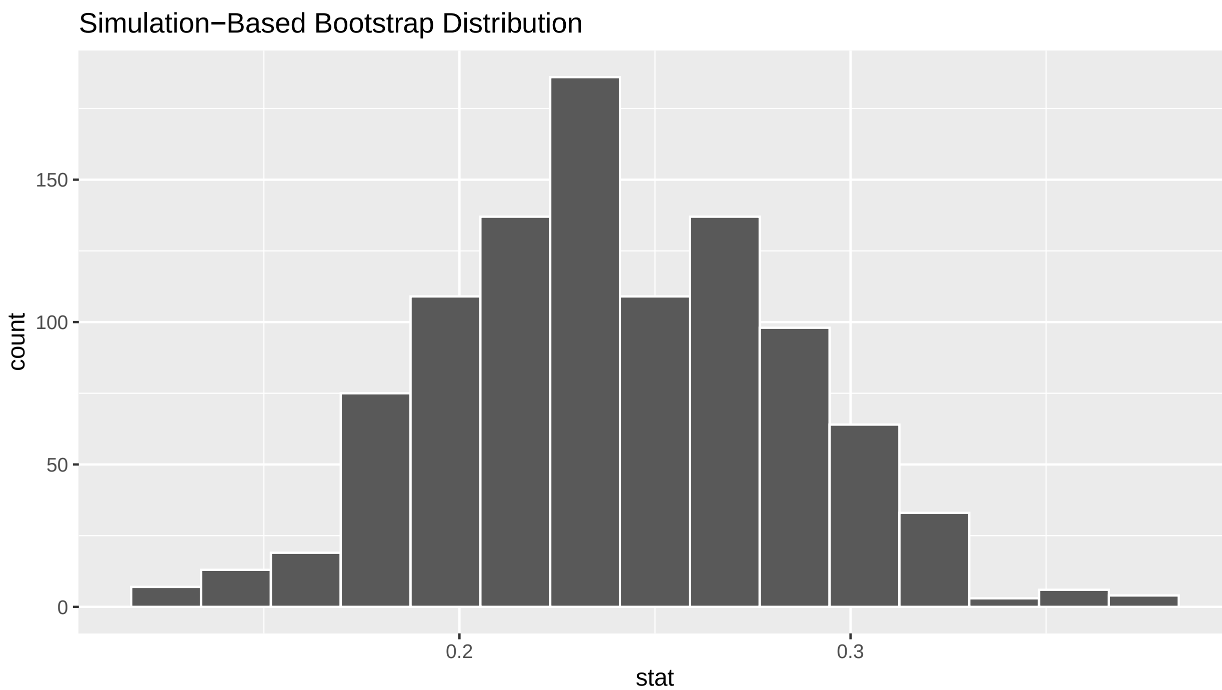
We are 90% confident that the true proportion of respondents that identify as a person of color is between 0.168311 and 0.311689.

Exercise 10

Create a visualization, using the `infer` package, of the bootstrap sampling distribution. Discuss if it is appropriate to use the standard error method.

Answer:

```r
visualize(infer_boot)
```



The distribution appears to be approximately normal so the use of the standard error method seems appropriate.

Documenting software

- File creation date: 2022-06-13
- R version 4.1.3 (2022-03-10)
- `tidyverse` package version: 1.3.1
- `moderndive` package version: 0.5.4
- `infer` package version: 1.0.0