Problem Set 08

Your Name

Last modified on August 24, 2024 10:56:13 Eastern Daylight Time

Background

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.

Setup

First load the necessary packages:

R Code

```
# 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)
```

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

R Code

```
if(!dir.exists("./Data")){dir.create("./Data")}
url <- "https://docs.google.com/spreadsheets/d/e/2PACX-1vSypSoDCMH2N76Vo2dZRPkw2q3t1mbvA
if(!file.exists("./Data/gss_sample.csv")){ download.file(url, destfile = "./Data/gss_sam
gss_sample <- read_csv("./Data/gss_sample.csv")
kable(head(gss_sample), caption = "GSS sample data")
```

Table 1: GSS sample data

age	race	tvhours
79	White	1
23	White	1
31	POC	4
53	White	4
39	White	1
59	White	1

Be sure to examine the ***GSS sample data***. The first six rows of the data are displayed above. Each row in the data set is a person that was surveyed (100 rows or cases in total). The variables in the data set include each respondent's `age`, `race`, and number of hours of TV watched a day `tvhours`.

Setting a seed: We will take some random samples and build sampling distributions in this lab. In order to make sure `R` takes the same random sample every time you run your code, you can do what is called "setting a seed". Do this in any code chunk that you take a random sample!

You can set a seed like so. Any number will do. (You do not need to run this right now…just showing you how)

```r
set.seed(45)
```

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; but, leave the seed at 42 for now.

R Code

```r
set.seed(42)
boot_samp_1000 <- gss_sample |>
  rep_sample_n(size = nrow(gss_sample),
               reps = 1000,
               replace = TRUE)
```

Note

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

- `size = nrow(gss_sample)` 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`.

Consider the first six rows of `boot_samp_1000` given below. Note that the `replicate` column labels each bootstrap resample (the first 100 rows are labeled 1, the next 100 rows are labeled 2, etc.)

R Code

```r
kable(head(boot_samp_1000))
```

replicate	age	race	tvhours
1	37	POC	2
1	27	POC	5
1	76	White	0
1	25	White	0
1	67	White	2
1	20	White	2

Problem 1

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

Problem 1 Answers

- Delete this and put your text answer here.

Step 2: Calculate the Bootstrap Statistic

Using `rep_sample_n()` to create the bootstrap distribution

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 1,000 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 three lines of code as follows:

R Code 5

```r
boot_distrib_tv <- boot_samp_1000 |>
  group_by(replicate) |>
  summarize(stat = mean(tvhours))
# Viewing the data
kable(head(boot_distrib_tv))
```

replicate	stat
1	2.72
2	3.45
3	2.96
4	2.80
5	3.16
6	2.77

Note

- 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 `stat`.
- This is the ***bootstrap distribution*** for the mean of `tvhours`.

Using `infer` to create the bootstrap distribution

R Code

```r
# infer pipeline
set.seed(321)
library(infer)
boot_dist_tv_infer <- gss_sample |>
  specify(response = tvhours) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean")
#
kable(head(boot_dist_tv_infer))
```

replicate	stat
1	3.16
2	2.86
3	3.92
4	3.49
5	2.54
6	3.86

Using a `for()` loop to create the bootstrap distribution

R Code

```r
# using a for loop
set.seed(451)
B <- 1000
bs_mean <- numeric(B)
for(i in 1:B){
  bss <- sample(gss_sample$tvhours,
                size = sum(!is.na(gss_sample$tvhours)),
                replace = TRUE)
  bs_mean[i] <- mean(bss)
}
#
head(bs_mean)
```

```
[1] 2.88 3.07 3.42 3.59 3.45 2.56
```

R Code

```r
kable(head(boot_distrib_tv))
```

replicate	stat
1	2.72
2	3.45
3	2.96
4	2.80
5	3.16
6	2.77

Problem 2

How many values of the bootstrap statistic `stat` are there in the object `boot_distrib_tv`? Please explain **why** there are this many values of the bootstrap statistic.

Problem 2 Answers

- Delete this and put your text answer here.

Visualizing the Bootstrap Distribution

1

R Code

```r
ggplot(data = boot_distrib_tv, aes(x = stat)) +
  geom_histogram(color = "white", binwidth = 0.25) +
  labs(title = "Bootstrap distribution",
       x = "boostrap statistic (mean tvhours)")
```

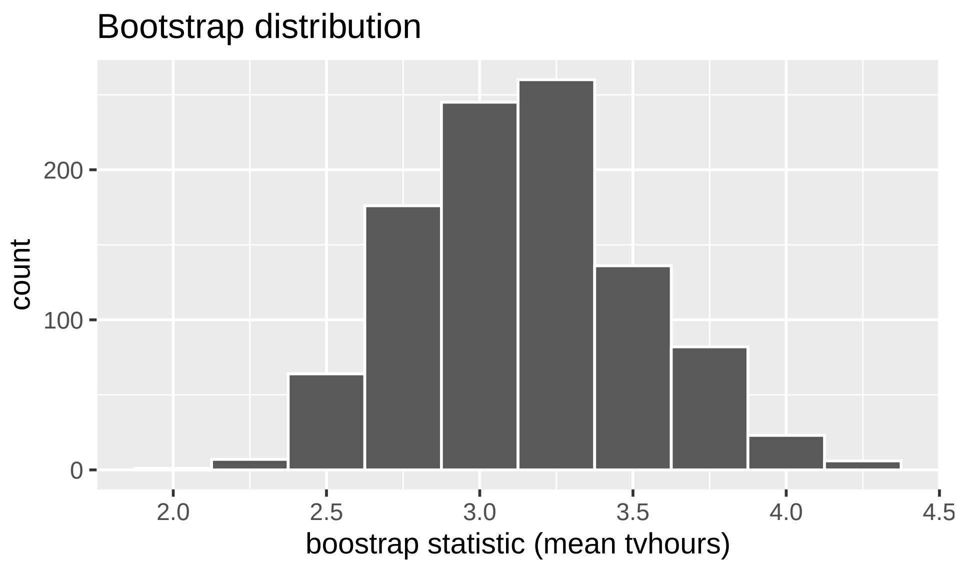


Figure 1: Bootstrap distribution of mean TV hours

R Code

```r
# Or infer pipeline---tibble must come from infer pipeline
visualize(boot_dist_tv_infer, bins = 9)
```

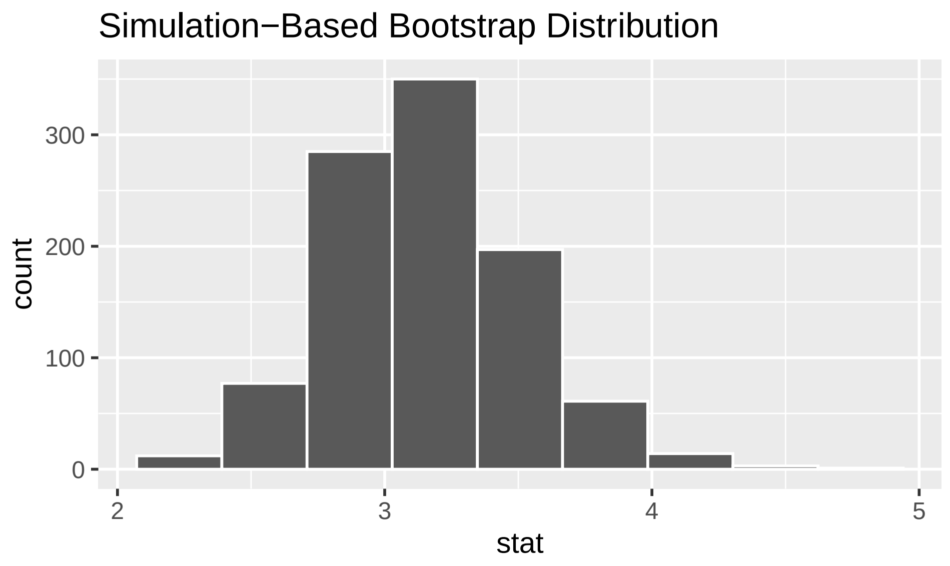


Figure 2: Bootstrap distribution computed with `visualize()`

R Code

```r
hist(bs_mean,
     breaks = "Scott",
     main = "Bootstrap Distribution",
     xlab = expression(paste(bar(x),"*")),
     col = "lightblue")
```

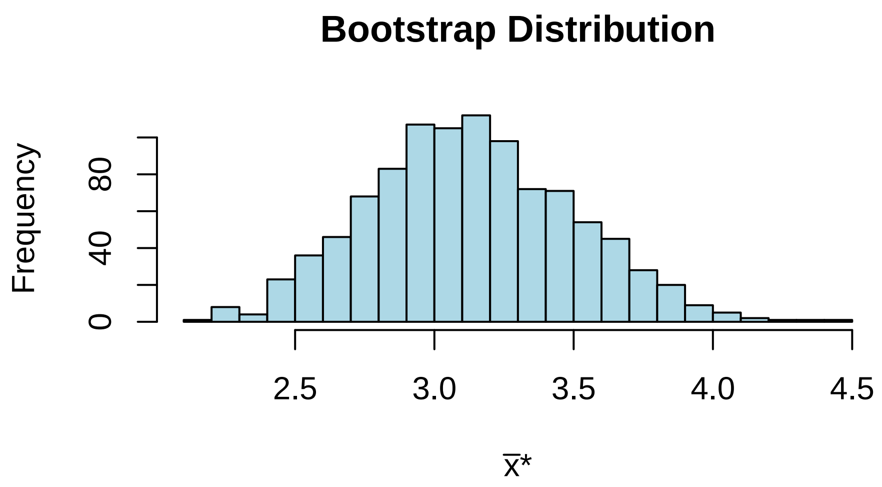


Figure 3: Bootstrap distribution of mean TV hours (base R histogram)

Step 3: CI from a Bootstrap Resample

CI Using the 95% Rule

Note

- the mean here is the mean of the original sample
- the SD here is the standard deviation of the bootstrap distribution, which recall

has a special name: the **standard error**.

R Code

```
# Note that z_{0.975} = 1.96
qnorm(0.975)
```

```
[1] 1.959964
```

```
(xbar <- mean(gss_sample$tvhours)) # mean of the original sample
```

```
[1] 3.14
```

```
boot_dist_tv_infer |>
  summarize(se = sd(stat),
            lower_ci = xbar - (qnorm(0.975) * se),
            upper_ci = xbar + (qnorm(0.975) * se)) -> bnci_tv
kable(bnci_tv)
```

se	lower_ci	upper_ci
0.3630419	2.428451	3.851549

```
#
standard_error_ci <- boot_dist_tv_infer |>
  get_confidence_interval(level = 0.95,
                          type = "se",
                          point_estimate = xbar)
kable(standard_error_ci)
```

lower_ci	upper_ci
2.428451	3.851549

CI Using the Percentile Method

Note

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

- 950 of the `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%.

R Code

```
bpci_tv <- boot_dist_tv_infer |>
  summarize(lower_ci = quantile(stat, 0.025),
            upper_ci = quantile(stat, 0.975))

kable(bpci_tv)
```

lower_ci	upper_ci
2.47975	3.92025

```r
# Or using get_confidence_interval()
boot_dist_tv_infer |>
  get_confidence_interval(level = 0.95, type = "percentile") |>
  kable()
```

lower_ci	upper_ci
2.47975	3.92025

```r
# Which is really just doing the following:
PCI <- boot_dist_tv_infer |>
            summarize(lower_ci = quantile(stat, 0.025),
                      upper_ci = quantile(stat, 0.975))
kable(PCI)
```

lower_ci	upper_ci
2.47975	3.92025

Note

The Percentile 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 `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 `stat`** fall (or 25 cases in this example 975/1000 = 0.975)
- The middle 95% of the values fall between these two quantiles

Visualizing the Confidence Interval

4 4

R Code

```r
ggplot(data = boot_dist_tv_infer, aes(x = stat)) +
  geom_histogram(color = "black", fill = "pink", binwidth = 0.15) +
  labs(title = "Bootstrap distribution with 95% CI",
       x = "boostrap statistic (mean tvhours)") +
  geom_vline(data = bpci_tv, aes(xintercept = lower_ci),
             color = "green", lwd = 1, lty = "dashed") +
  geom_vline(data = bpci_tv, aes(xintercept = upper_ci),
             color = "blue", lwd = 1, lty = "dashed") +
  theme_bw()
```

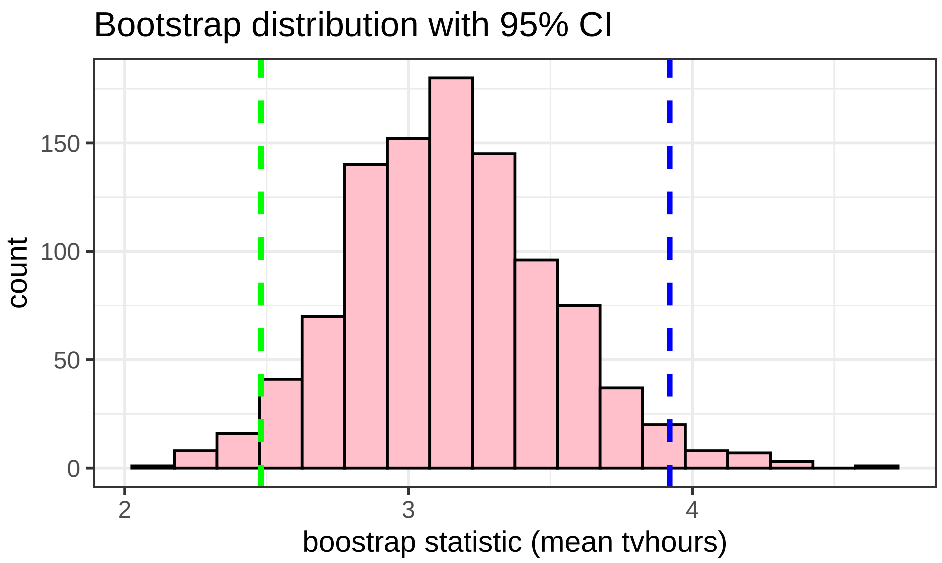
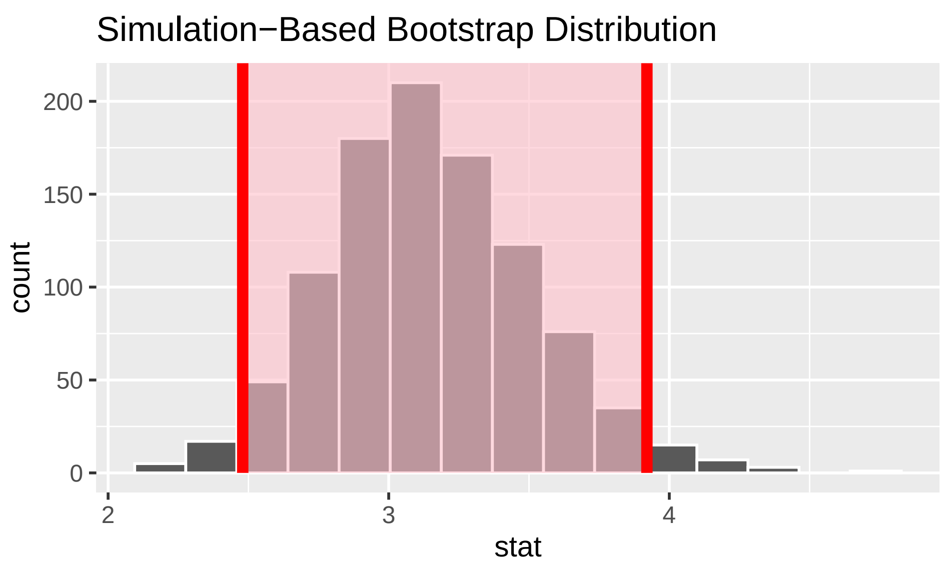


Figure 4: Showing the endpoints of a 95% bootstrap percentile CI

R Code

```r
boot_dist_tv_infer |> visualize() +
  shade_confidence_interval(endpoints = PCI, color = "red", fill = "pink")
```



Problem 3

- **If** we calculated a **90% bootstrap percentile** confidence interval for the mean of `tvhours` using this same bootstrap resample (`boot_dist_tv_infer`) and the percentile method, roughly how many of the 1000 values of `tv_mean` would fall between the `lower_ci` and the `upper_ci`?
- Programatically count the number of values that actually fall between the computed `lower_ci` and the `upper_ci`.

Problem 3 Answers

- Delete this and put your text answer here.

```
# Type your code and comments below
```

Problem 4

Use the bootstrap resampling distribution for `tvhours` generated using the infer pipeline (`boot_dist_tv_infer`) and the **bootstrap percentile** method to calculate a 99% **bootstrap percentile** confidence interval for the mean `tvhours`. Round your answer to two decimal places. Make sure to use inline `R` code to report your answer and include appropriate units with the confidence interval.

Problem 4 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.

Problem 5

Which confidence interval is **WIDER**: the 95% confidence interval or the 99% confidence interval for the population mean `tvhours` μ_{tv}? Why?

Problem 5 Answers

- Delete this and put your text answer here.

Problem 6

- Use the bootstrap resample we generated using the function `rep_sample_n()` (`boot_samp_1000`) to generate a **bootstrap distribution** for the sample mean respondent `age` instead of `tvhours`. Use a seed of 21. Store your resulting bootstrap distribution in `boot_dist_age1`.
- Use the `infer` pipeline to generate a **bootstrap distribution** for the sample mean respondent `age` instead of `tvhours`. Use a seed of 21. Store your resulting bootstrap distribution in `boot_dist_age2`. Note: you will need to start with the original sample (`gss_sample`).

Problem 6 Answers

```
# Type your code and comments inside the code chunk
```

Problem 7

Calculate 95% confidence intervals for the population mean respondent `age` μ_{age} using the **95% rule** method with the data in `boot_dist_age1` and `boot_dist_age2`.

Problem 7 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.
- Delete this and put your text answer here.

Problem 8

Calculate a 95% bootstrap percentile confidence interval for the population mean respondent `age` μ_{age} using the values in `boot_dist_age1`.

Problem 8 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.

Problem 9

How do the 95% confidence intervals you calculated in 7 and 8 compare? i.e. are the 95% CI values similar or are they pretty different?

Problem 9 Answers

- Delete this and put your text answer here.

Problem 10

Use the **bootstrap resampling distribution** for the sample mean respondent `age` (`boot_dist_age1`) and the percentile method to calculate an 80% confidence interval for the population mean respondent age μ_{age}.

Problem 10 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.

Bootstrap Sampling Distribution & Confidence Intervals with Categorical Variables

Step 1: Take 1000 Bootstrap Resamples

Step2: Calculate the Bootstrap Statistic $\hat{p}$

R Code

```r
boot_distrib_POC <- boot_samp_1000 |>
  group_by(replicate) |>
  summarize(n = n(),
            POC_count = sum(race == "POC"),
            boot_stat = POC_count/n,
            phat_boot = mean(race == "POC"))
kable(head(boot_distrib_POC))
```

replicate	n	POC_count	boot_stat	phat_boot
1	100	26	0.26	0.26
2	100	24	0.24	0.24
3	100	25	0.25	0.25
4	100	16	0.16	0.16
5	100	28	0.28	0.28
6	100	22	0.22	0.22

Note

Note that with a categorical variable, the code differs in two important respects now:

- the population parameter that we don't know, but are inferring about via sampling, is now the population proportion p that identify as a POC.
- the sample statistic AKA point estimate that we calculate with the summarize

command is now the **sample proportion** $\hat{p}$ rather than a sample mean $\bar{x}$.

To get our proportion $\hat{p}$ of **ONE** of the race categories (POC), we need to **first** calculate the total sample size for each replicate and the count of how many cases are `race == "POC"` in each replicate.

R Code

```
# Or using infer
set.seed(32)
gss_sample |>
  specify(response = race, success = "POC") |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "prop") -> boot_dist_POC_infer
kable(head(boot_dist_POC_infer))
```

replicate	stat
1	0.22
2	0.22
3	0.22
4	0.21
5	0.23
6	0.20

Step 3: Generate the 95% Confidence Interval

CI Using the 95% Rule

R Code

```r
phat <- mean(gss_sample$race=="POC")
boot_distrib_POC |>
  summarize(se = sd(boot_stat),
            lower_ci = phat - (qnorm(0.975) * se),
            upper_ci = phat + (qnorm(0.975) * se)) |>
  kable()
```

se	lower_ci	upper_ci
0.0421354	0.1574161	0.3225839

```r
### Using infer
boot_dist_POC_infer |>
  get_confidence_interval(level = 0.95,
                          type = "se",
                          point_estimate = phat) |>
  kable()
```

lower_ci	upper_ci
0.1571938	0.3228062

CI with the Percentile Method

R Code

```r
boot_distrib_POC |>
  summarize(lower_ci = quantile(boot_stat, 0.025),
            upper_ci = quantile(boot_stat, 0.975)) |>
  kable()
```

lower_ci	upper_ci
0.16	0.33

```r
### Using infer
boot_dist_POC_infer |>
  get_confidence_interval(level = 0.95,
                          type = "percentile",
                          point_estimate = phat) |>
  kable()
```

lower_ci	upper_ci
0.16	0.33

Problem 11

Calculate a 95% CI for the **population proportion** of respondents p who identified as **White** using BOTH the percentile and the 95% rule method. Note that you will first need to generate the bootstrap distribution for the proportion of respondents who identified as `White`. Compute the requested 95% CI using `boot_samp_1000` and by generating the bootstrap distribution using the infer pipeline. Use a seed of 43 for the infer pipeline.

Problem 11 Answers

```r
# Type your code and comments inside the code chunk
```

```r
# Type your code and comments inside the code chunk
# percentile method

# 95% rule
```

- Delete this and put your text answer here.
- Delete this and put your text answer here.

```
# Type your code and comments inside the code chunk

# percentile method

# 95% rule
```

- Delete this and put your text answer here.
- Delete this and put your text answer here.

Confidence Intervals Based on the Theoretical Normal Distribution

ModernDive Section 8.7.2

-
-

R Code

```
x_bar = mean(gss_sample$tvhours)
gss_sample |>
  summarize(sd = sd(tvhours),
            n = n(),
            se = sd/sqrt(n),
            lower_ci = x_bar - qnorm(.975) * se,
            upper_ci = x_bar + qnorm(.975) * se) -> tci_tv
kable(tci_tv)
```

sd	n	se	lower_ci	upper_ci
3.592979	100	0.3592979	2.435789	3.844211

Problem 12

Write down the three 95% confidence intervals for the population mean of `tvhours` μ_{tv} you've computed in this problem set. Do this by replacing X, Y, A, B, P, and Q with the appropriate values you've computed.
When you are done, make sure all the | in the table still line up so your results print out in a table!

Problem 12 Answers

CI construction method	lower value	upper value
Using boostrap: 95% rule	X	Y
Using boostrap: percentile rule	A	B
Using mathematical formula	P	Q

Problem 13

In your opinion: would you say these three confidence intervals are similar?

Problem 13 Answers

- Delete this and put your text answer here.

Turning in Your Work

Tip

- Make sure you **render a final copy with all your changes** and work.
- Look at your final html file to make sure it contains the work you expect and is formatted properly.

Logging out of the Server

Tip

- Save all your work.
- Click on the orange button in the far right corner of the screen to quit `R`
- Choose **don't save** for the **Workspace image**
- When the browser refreshes, you can click on the sign out next to your name in the top right.
- You are signed out.

```
sessionInfo()
```