Instructions

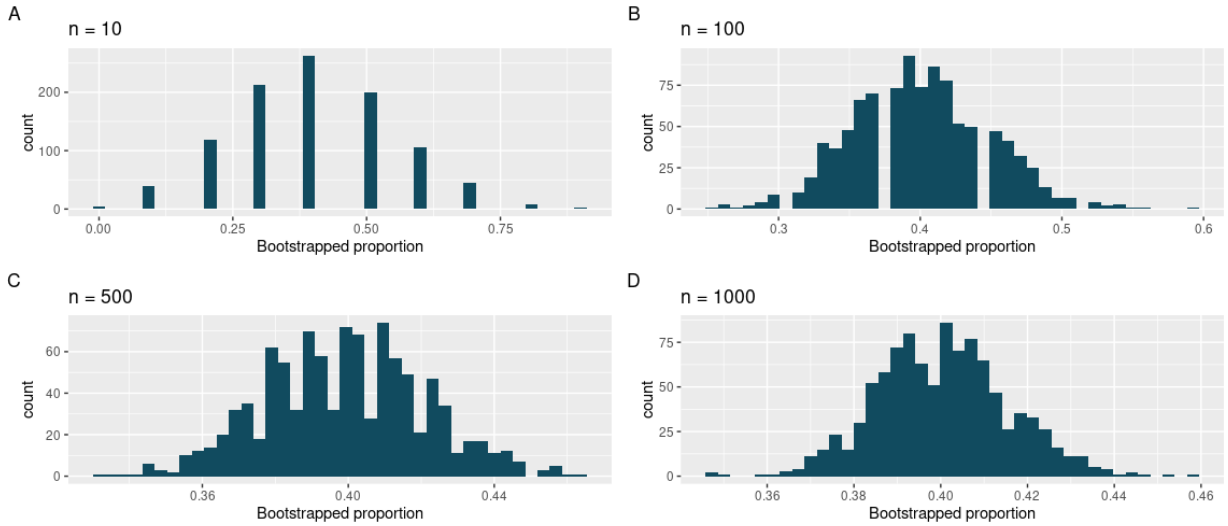
Upload a PDF file, named with your UC Davis email ID and homework number (e.g., sfrei_hw4.pdf), to Gradescope (accessible through Canvas). You will give the commands to answer each question in its own code block, which will also produce output that will be automatically embedded in the output file. All code used to answer the question must be supplied, as well as written statements where appropriate.

All code used to produce your results must be shown in your PDF file (e.g., do not use `echo = FALSE` or `include = FALSE` as options anywhere). Rmd files do not need to be submitted, but may be requested by the TA and must be available when the assignment is submitted.

Students may choose to collaborate with each other on the homework, but must clearly indicate with whom they collaborated.

Problem 1: [IMS] 12.6

1. **Bootstrap distributions of $\hat{p}$, III.** Each of the following four distributions was created using a different dataset. Each dataset had the same proportion of successes ($\hat{p} = 0.4$) but a different sample size. The four datasets were given by $n = 10, 100, 500$, and 1000.



Consider each of the following values for the true popluation p (proportion of success). Datasets A, B, C, D were bootstrapped 1000 times, with bootstrap proportions as given in the histograms provided. For each parameter value, list the datasets which could plausibly have come from that population. (Hint: there may be more than one dataset for each parameter value.)

a. p = 0.05

b. p = 0.25

c. p = 0.45

d. p = 0.55

e. p = 0.75

Problem 2

In this problem we work through how to do a randomization test in R. We will work with the following example dataset.

```
df <- tibble(
  group = rep(c("control", "treatment"), each=50), # 50 control and 50 treatment observations
  response = factor(c(rep("success", 20), rep("failure", 30),
                      rep("success", 29), rep("failure", 21)),
                    levels = c("success", "failure"))
)
```

Part (a) - computing sample proportion

Write a function `compute_proportions` which takes in the argument `tib`, a tibble, which we assume takes the form of the above tibble (i.e., it has two columns, "group" and "response", where "group" is a factor which takes values "control" and "treatment" while "response" is a factor which takes values "success" and "failure".)

The function `compute_proportions` returns a tibble with two variables, "group" and "proportion". Each row of the returned table corresponds to the proportion of "success" within each group of "control" and "treatment".

```
# code here
```

Part (b) - generating random treatment/control vectors

Write a function `randomized_treatment` which takes as its argument `tib`, a tibble in the same format as the previous part of the problem, and returns a tibble where the treatments and controls are assigned randomly. *Hint: the function `sample()` in R should be helpful. Examine what this function does to a vector of categorical variables. What happens when you repeatedly apply this function to the same tibble (e.g., the tibble `df` in the introduction to the problem?)

```
# code here
```

Part (c) - randomization test

Write a function `randomization_tests` which takes two arguments: `tib`, a tibble, and `n`, an integer with default value of 500. Assume `tib` has the same form as in the previous problem. It returns a named list, with one component, `actual_diff`, which returns the difference in proportion of successes in treatment vs. proportion of successes in control (this difference is positive if there are more successes in treatment). The second component of the named list, `randomized_differences`, is a vector of length `n` which computes the difference in proportions between treatment and control when the treatment and control groups are assigned *randomly*, using the `randomized_treatment()` function from the previous part of the problem.

You may wish to first initialize a vector `differences`, and to use a for loop - see below. Recall that if you want to set the i-th component of the vector differences to be e.g. 3, you can write `differences[i] <- 3`.

```
randomization_tests <- function(tib, n = 500) {
  set.seed(123) # For reproducibility
  differences <- numeric(n)

  # code here
  for(i in 1:n) {
    # code here
  }

  # code here
}
```

Part (d) - plotting the results

Write a function, `plot_randomization_results`, which takes as its argument a named list `randomization_results` which has the same format as the output of the previous part of the problem (i.e., a component "actual_diff" and a component "differences"), and returns a histogram with binwidth 0.005 of the values of the differences that come from using $n = 1000$ randomization tests. In addition, plot a dashed red line which denotes the actual difference observed in the original data. Make sure the x-axis, y-axis, and title of the plot are descriptive, and that the x-axis and y-axis labels are visible.

```
# code here
```

Part (e) - analysis of how extreme events are

Write a function `percent_more_extreme()`, which takes as its input `randomization_results` (a named list which is the output of the function `randomization_tests` from part (b)) and returns the proportion of the simulated differences-in-proportions in randomizations which have as extreme as a result as the observed difference in the data. For example:

- If the original data had difference in proportions of 0.03, and randomizations had difference in proportions of -0.02, 0.01, 0.04, 0.05, and 0.07, then the function would return $0.6 = 3/5$ since 0.04, 0.05, and 0.07 were more extreme (positive) than the observed difference of 0.03.
- If the original data had difference in proportions of -0.01, and randomizations had difference in proportions of -0.02, 0.01, 0.04, 0.05, and 0.07, then the function would return $0.2 = 1/5$ since only -0.02 was more extreme (negative) than the observed difference of -0.01.

For this problem you can assume that the observed differences in the data is not zero. Be sure that the reader can read every line of your code in the PDF!

```
# code here
```

Part (f) - running the analysis and interpreting

Now run the code that you've built up as follows: (remember to remove the `eval=FALSE` when you are knitting your own Rmd)

```
randomization_results <- randomization_tests(df)
plot_randomization_results(randomization_results)
percent_more_extreme(randomization_results)
```

Interpret the graph and the results of the function `percent_more_extreme`. How likely was it that the treatment is independent of success/failure?