Problem Set 1

Author Name

2023-09-07

Note: *In all of these r codeblocks, I set eval=FALSE. You will need to undo that.* I also ask you to save all code to the output folder. You can do this by adding the following after any graph you want to save. Alternatively, you can save the graph as an `object` and then save it to the output folder. If you save the plot as an object, you will need to add `plot` to the ggsave function.

```r
ggplot(CODEHERE)
ggsave(filename=paste0(output,'figure.pdf'))
#Save the plot as an object
plot<-ggplot(CODEHERE)
#Just calling the object `plot` will show the graph
plot

othercode
#Saves the plot object
ggsave(plot, filename=paste0(output,'figure.pdf'))
```

I will be grading this problem set based on the following criteria: - Quality of code (33%): Is it well-commented? Is it easy to follow? Can I run it? - Quality of graphs (33%): Are they well-labeled? Do they have titles? Do they have legends? Are they formatted well? - Quality of answers (33%): Are they clear? Do they answer the question?

1. Open the .gitignore and add *.csv,* .dta, *.Rdata, etc.

This will prevent you from pushing any large data files to GitHub. GitHub cannot receive data files that are larger than 100MB. You can use Git Large File Storage to submit larger files, but even that has limits.

2. Create folders for data, code, documentation, output, and any necessary subfolders within this repository.

Reorganize the data files and code within these folders.

3. Skim the [paper]https://opportunityinsights.org/wp-content/uploads/2023/07/CollegeAdmissions_Paper.pdf and also reference the non-technical summary and a New York Times report. Briefly explain how the data were generated.

4. Correct the download_data.R to download the .csv to your preferred data folder.

```r
school_data <- read.csv(paste0(data,'CollegeAdmissions.csv'))
```

5. First, look at the data. Any quirks about it?

```
school_data
```

6. Look at the variables `rel_apply` and `rel_attend`. What do these mean?

ANSWER HERE

7. Check the documentation for an explanation of what each row is – use that information to calculate the number of rows that there should be in the data. Does this match the number of rows in the dataset?

Answer how you calculated the number of rows and whether it matches the number of rows in the dataset.

```
some_function_to_count_rows(school_data)
```

8. What about the variable `rel_attend_cond_app`. What does this mean? Can you verify that it is calculated correctly?

9. The codebook mentions that most of these data are test-score-reweighted. What does that mean?

ANSWER HERE.

10. Replicate Figure 2b from the paper. I recommend using the package `ggplot2`.

You'll need to check the Appendix to get the exact group of schools. Interpret the graph. Save it to your output folder.

```
ggplot(data=school_data)
```

11. Replicate Figures 3a and 3b from the paper with 95% CI bars. Save them to your output folder. Interpret these graphs.

```
ggplot(data=school_data)
```

11. Replicate Figures 4a and 4b from the paper with 95% CI bars. Save them to your output folder. Interpret these graphs.

```
ggplot(data=school_data)
```

12a. Plot the relative application, attendance, and matriculation rates for the schools in the NESCAC. Save it to your output folder. Interpret this graph.

```
filter(school_data, somecondition) %>%
    ggplot()
```

12b. Add 95% confidence intervals to the plot you just made.

```
school_data <- mutate(school_data,lower=estimate-1.96*se,upper=estimate+1.96*se)
plot + geom_errorbar()
```

13. Come up with your own cool visualization of the data. Save it to your output folder. Interpret this graph.

```
ggplot(data=school_data)
```

14. Based on what you have seen so far, do you think that the admissions process is meritocratic? What about at Bates and other NESCAC schools? Why or why not? What else would you want to know?

Hints:

Below I provide a list of handy functions/tips for R.

These `tidyverse` functions: - `filter()` – subset the data - `mutate()` – create new variables - `group_by()` – group the data - `summarise()` – calculate summary statistics - `%>%` – pipe operator to chain together functions applied to the same data frame

And this is some useful base R: - `c(1,2,3)` – create a vector - `seq(1,10,by=2)` – create a sequence - `rep(1,10)` – repeat a value - `%in%` – check if a value is in a vector - `signif` – round to a certain number of significant figures - `stopifnot` – stop the code if a condition is not met. If a condition must be true, you want to stop the code before advancing.

And some useful ggplot2: - `geom_point()` – scatterplot - `geom_line()` – line plot - `geom_errorbar()` – error bars - `aes()` – the aesthetic function, which can be used to set the x and y axes, as well as the color, shape, etc. of the points - You can save any ggplot object as a variable and then add layers to it. For example, you can save a scatterplot as `plot` and then add a line to it with `plot + geom_line()`. - `ggsave()` – save a plot to a file. - You can specify the plot object and the filename.

```
plot<-ggplot(data=school_data, aes(x=rel_apply, y=rel_attend)) + geom_point()
ggsave(filename=paste0(output,'figure.pdf'),plot=plot)
```

```
- If you don't specify a plot object, it will save the last plot you made:
```

```
ggplot(data=school_data, aes(x=rel_apply, y=rel_attend)) + geom_point()
ggplot(data=school_data, aes(x,y)) + geom_point()
#This saves the second ggplot
ggsave(filename=paste0(output,'figure.pdf'))
```

Basic statistics:

Use the rough formula that the upper and lower bounds of the confidence interval are:

$$\hat{\beta} \pm 1.96 \times SE(\hat{\beta})$$

where $\hat{\beta}$ is the estimated statistic and $SE(\hat{\beta})$ is the standard error of the statistic.