PS405: Linear Models

Problem Set 4

Professor: Nicole Wilson TA: Artur Baranov

Due: Thursday, February 13, 2025

Please submit your write-up, including *any* code that you ran, via Canvas. We suggest using Quarto or `RMarkdown`, but any software that renders R code into a PDF document will work. I have provided an `RMarkdown` template if you prefer to use it (note that the process/format is very similar to Quarto, if you are familiar with that). This should be completed *before* class begins at 9:30 AM. Please save your write-up with easy-to-recognize file names (e.g., `ps1_wilson.pdf` or `ps1_baranov.pdf`). Late submissions will not be accepted without special permission from the course instructors. Where instructed to follow a specific coding procedure in `R`, such as the creation of a new function or a data generating process, please include the code in a code chunk in your write up. For example, you can use `echo = TRUE` in the chunk header in Quarto or `RMarkdown` to ensure your code is visible in the output. You do not need to submit the raw file you use to produce your PDF, but the course instructors may request it on a case-by-case basis.

Please post all questions to the Canvas anonymous discussion forum.

Problem 1

In this problem, we are going to run a simulation to help us think about the uncertainty of estimates. This problem will take you through step by step.

(a) First, we need to think about the data generating progress (DGP) in our simulated population, and then we will write a function to estimate our model.

- We have an X that is randomly assigned in an experiment to those who are sampled. An equal number of people are assigned to treatment and control (i.e., there is an equal probability of being in the treatment or the control group). *Note: A distribution with only two possible outcomes is called a binomial distribution.*
- We have a covariate Z that is normally distributed in the population with mean of 5 and a standard deviation of 2: $Z \sim N(5, 2)$ (this is wrong in the template so please update accordingly).

- We have an outcome Y that has a mean of 50 for those who are not treated. Y is affected by our treatment, X, but the effect depends on the value of the covariate Z. But there is no direct effect of Z on Y. *Note: This is typically referred to as "heterogeneous treatment effects" – the effect of a treatment on an outcome depends on some other characteristic. For instance, the treatment may have different effects based on gender or age.*
- We also have some random variation in Y that is unrelated to X and Z that we call u. u is normally distributed with a mean of zero and a standard deviation of 40: $u \sim N(0, 40)$

Together we have:

$$Y = \beta_0 + \beta_1 X + \beta_2 Z + \beta_3 X * Z + u$$

$$Y = 50 + 10X + 0Z + 2X * Z + u$$

We are going to write a function that simulates one draw of a sample for which we will run the experiment. To do this you will use the `function()` command. It will have five arguments: the sample size (`n`), the relationship between X and Y (`b1`), the relationship between Z and Y (`b2`), the size of the interaction effect (`b3`), and the error standard deviation (`usigma`). An argument is just an input into a function. For instance, when we run `ggplot` we put in information about what data we are using, which varibles, etc. I'm giving you a code template but you need to:

- Come up with a descriptive but short name for your function (replace `fcnname`, which is not, in fact, a good function name)
- When we set up a function, we can put in default values for are arguments. We should set these to match the DGP described above. I already set the default sample size to be 500 and you can do the rest.
- Set up the distribution for u, following the format of Z. Remember that the standard deviation is not fixed at a certain value, but an argument in our function, so you will put in *usigma* instead of a single number. That way, if we put in a different value for usigma when we run our function, that will become the new standard deviation of the error.
- Finish coding the regression that we would want to run on our data to detect our treatment effect and how it varies based on Z. Keep in mind what we would be able to observe/measure and what we wouldn't.
- Make sure to change the chunk to `eval=TRUE` so your code actually runs

```
fcnname <- function(n=500, b1="put default here and remove quotes",
                    b2="put default here and remove quotes",
                    b3="put default here and remove quotes",
                    usigma="put default here and remove quotes") {
  z <- rnorm(n, 5, 3) #draw from a normal distribution
  u <- rnorm( , , ) #follow the same template as for Z
  x <- rbinom(n, 1, .5) # treatment assignment
  y <- 50 + b1*x + b2*z + b3*x*z + u
```

```
  mod <- lm()

  return(mod)
}
```

Now you have written your own function that works just like any other function in `R` – pretty impressive! Test it out and see what happens! Since we are not changing any of the default arguments, you can just call `fcnname()` (with whatever your function name is). It should spit out a model, because that is what we told it to "return."

(b) Set your seed to 4052025 using `set.seed(4052025)` and run your function again.

Interpret your output (all coefficients), put them in a table, and plot the interaction effect. It might help to save the function output as an object.

Note: We set the seed because our function is a random process. If we run it one time, and then run it again, we will get different results. Setting the seed before running the function will ensure that you always get the same results for this particular draw (and will mean that you don't have to keep re-writing your interpretion).

(c) Now let's pretend that we can redo this experiment with different samples of the same size from the population, to generate a sampling distribution.

Here, a `for` loop will probably be helpful. You can also use an `apply()` function to complete this part, if you are more familiar with that, but the steps provided are going to be for loops. Let's simulate our experiment and model estimation 5000 times. Within the code we want to say what will happen in each of the 5000 rounds.

Before we start our loop, we want to set up some empty vectors that will hold the results of our loop. We want to save:

- A vector of our estimated $\hat{\beta}_1$'s, the estimated treatment effect (I did this one for you; fill the rest into the code)
- A vector of our estimated $\hat{\beta}_2$'s, the estimated treatment effect
- A vector of our estimated $\hat{\beta}_3$'s, the estimated interaction effect
- A vector of the lower bound of our estimated confidence intervals for $\hat{\beta}_1$'s
- A vector of the upper bound of our estimated confidence intervals for $\hat{\beta}_1$'s
- A vector of the lower bound of our estimated confidence intervals for $\hat{\beta}_2$'s
- A vector of the upper bound of our estimated confidence intervals for $\hat{\beta}_2$'s
- A vector of the lower bound of our estimated confidence intervals for $\hat{\beta}_3$'s
- A vector of the upper bound of our estimated confidence intervals for $\hat{\beta}_3$'s

For all, we want to estimate 95% confidence intervals.

Note: There are more efficient, but more complicated ways to do this, which you are welcome to try.

First, we will run our function that generates out data and estimates our regression. Then we will want to pull out our estimated coefficients and confidence intervals. We will assign them to the vector using the index `i` which indicates with run of the loop we are on. This will ensure that we are not saving over our previous result each time. In short, the result from the first run of the loop will save in the first position in the vector, the second in the second, and so on. To get confidence intervals, you may want to use the `tidy()` function from the `broom` package.

You will want to set a seed before running your loop so that your results stay stable.

```
beta_1_out <- c()
# do the same thing for the rest, giving them descriptive names

for(i in 1:5000){
  #run fcn, saving output to object

  beta_1_out[i] <- out$coefficients["x"] #save the estimated coefficients

  # get and save the confidence intervals
}
```

(d) Now we want to explore the properties of these sampling distributions.

(i) Standard errors

First, what are the standard errors (the standard deviations of the sampling distributions) of our estimators $\hat{\beta}_1$, $\hat{\beta}_2$, and $\hat{\beta}_3$? Now, look at your model output from (b) and see what the estimated standard errors were having just done one trial (like how things are in the real world where we only collect data once). How do they compare? Do you think this is a good estimate?

(ii) Confidence interval for Z

Now, we want to know what proportion of our confidence intervals crossed zero for $\hat{\beta}_2$.

Note: You may want to combine your vectors into a dataframe to make it easier to manipulate, but this is not necessary.

It may be useful to look into the `ifelse()`, `sign()` and/or `abs()` commands.

How does this compare to what you would expect? Why?

(iii) Confidence interval for interaction

Now, we want to know what proportion of our confidence intervals crossed zero for $\hat{\beta}_3$.

Knowing what you know about the true DGP, what type of error is this? What might this indicate about our research design?

(iv) P-values

We didn't capture the p-values in our loop, but knowing what you know about the relationship between confidence intervals and p-values, what proportion of the p-values for $\hat{\beta}_2$ and $\hat{\beta}_3$ would have been less than 0.05?

(e) Now we are going to run a similar loop, but the only thing we will change is our sample size, which we will increase to 1200. We still want to run 5000 trials. Remember to set your seed before running your loop (it doesn't matter what you set it to).

(f) Now we want to explore the properties of these sampling distributions again.

(i) Standard errors.

Look at the standard errors of our estimators at $n = 1200$ (using the simulated sampling distribution). How do they compare to the standard errors when $n = 500$?

(ii) Confidence intervals

Again, we want to know what proportion of our confidence intervals crossed zero for $\hat{\beta}_2$ and $\hat{\beta}_3$. Is this an improvement over the $n = 500$ case? Why or why not?

(g) Now we are going to run a similar loop, keeping our sample size at 1200, but decreasing our error variance. Specifically, we are going to set our error standard deviation to 20. Use the arguments you set up in your function: the function itself should not have to change.

(i) Standard errors

Compare the standard errors of our estimators when we hold constant $n = 1200$ but changing the error variance. Is this surprising or not? Why? Hint: Think about how OLS estimates standard errors.

(ii) Confidence intervals

One more time, we want to know what proportion of our confidence intervals crossed zero for $\hat{\beta}_2$ and $\hat{\beta}_3$. Is this an improvement over the results in part (f)?

(iii) Finally, how does the variance of the residuals change as the variance of the population error changes?