Submission Problem Set 4

Nicholas R. Gonzalez

Worked on with Tanner Bentley.

Problem 1

(a)

```
pset4_function_noseed <- function(n, b1, b2, b3, usigma) {
  z <- rnorm(n, 5, 2)
  u <- rnorm(n, 0, usigma)
  x <- rbinom(n, 1, .5)
  y <- 50 + b1*x + b2*z + b3*x*z + u

  model <- lm(y ~ x * z)

  return(model)
}
```

(b)

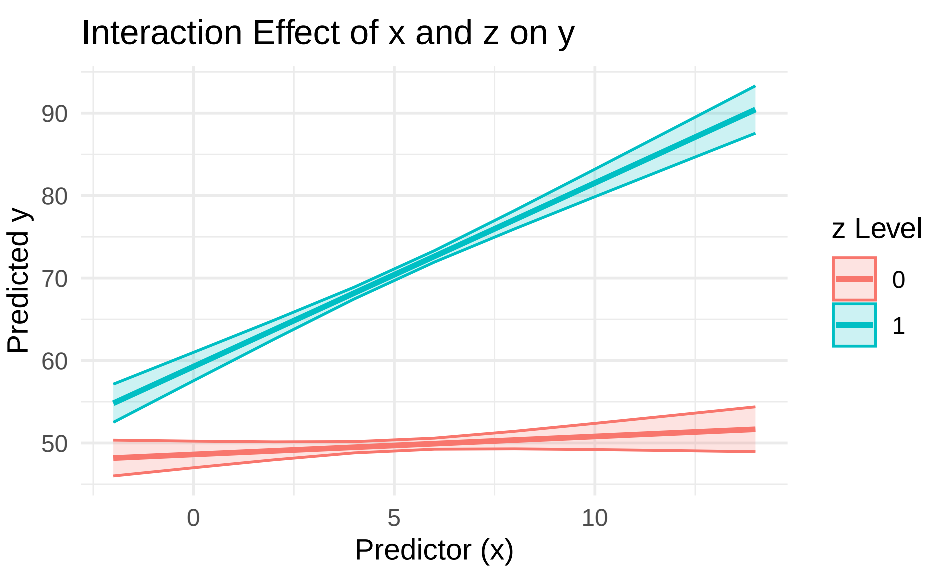
The results of our simulation suggest that there is a significant interaction between x and z, on y. The slope coefficient for $x * z$, 2.01, tells us that the effect of x on y relates to the value of z. In short, as z increases, the impact of x on y becomes stronger. meaning that as z increases, the impact of x on y becomes stronger. We also see this within the plot created using *ggeffects*.

```
set.seed(4052025)

pset4_function <- function(n, b1, b2, b3, usigma) {
  z <- rnorm(n, 5, 2)
  u <- rnorm(n, 0, usigma)
  x <- rbinom(n, 1, .5)
  y <- 50 + b1*x + b2*z + b3*x*z + u

  model <- lm(y ~ x * z)
```

```r
  return(model)
}

model <- pset4_function(n=500, b1= 10, b2= 0, b3= 2, usigma= 5)

tidy(model)
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   48.6       0.821     59.2  5.48e-227
2 x             10.7       1.20       8.88 1.20e- 17
3 z              0.218     0.151      1.45 1.49e-   1
4 x:z            2.01      0.220      9.13 1.74e- 18
```

```r
interaction_plot <- ggpredict(model, terms = c("z", "x"))


ggplot(interaction_plot, aes(x = x, y = predicted, color = factor(group))) +
  geom_line(size = 1) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high, fill = factor(group)), alpha = 0.2) +
  labs(title = "Interaction Effect of x and z on y",
       x = "Predictor (x)",
       y = "Predicted y",
       color = "z Level",
       fill = "z Level") +
  theme_minimal()
```



(c)

```
set.seed(4052025)

beta_1_out <- c()
beta_2_out <- c()
beta_3_out <- c()
b1_lower <- c()
b1_upper <- c()
b2_lower <- c()
b2_upper <- c()
b3_lower <- c()
b3_upper <- c()


for(i in 1:5000){
  out <- pset4_function(n=500, b1= 10, b2= 0, b3= 2, usigma= 5)
  beta_1_out[i] <- out$coefficients["x"] #save the estimated coefficients
  beta_2_out[i] <- out$coefficients["z"]
  beta_3_out[i] <- out$coefficients["x:z"]

  tidy_out <- tidy(out, conf.int = TRUE)
  b1_lower[i] <- tidy_out$conf.low[2]
  b1_upper[i] <- tidy_out$conf.high[2]
  b2_lower[i] <- tidy_out$conf.low[3]
  b2_upper[i] <- tidy_out$conf.high[3]
  b3_lower[i] <- tidy_out$conf.low[4]
  b3_upper[i] <- tidy_out$conf.high[4]

}

pset4_loop <- as.data.frame(cbind(beta_1_out, beta_2_out, beta_3_out, b1_lower, b1_upper, b2_lo
```

(d)

(i)

The standard errors are almost exactly the same. Since they are the same, this suggests that the estimate is pretty accurate.

```
sd_beta_1 <- sd(pset4_loop$beta_1_out)
sd_beta_2 <- sd(pset4_loop$beta_2_out)
sd_beta_3 <- sd(pset4_loop$beta_3_out)

sd_df <- data.frame(
```

```
  Coefficient = c("Beta_1", "Beta_2", "Beta_3"),
  Standard_Deviation = c(sd_beta_1, sd_beta_2, sd_beta_3)
)

print(sd_df)
```

```
  Coefficient Standard_Deviation
1      Beta_1          1.2158039
2      Beta_2          0.1593848
3      Beta_3          0.2265468
```

(ii)

I am not fully sure what to expect, or what to have expected. A proportion of 95% is higher than what I assume it is suppose to be, which is 5%, since for a 95% CI, only 5% of the data should cross zero.

```
cross_zero_b2 <- mean(pset4_loop$b2_lower <= 0 & pset4_loop$b2_upper >= 0)

print(cross_zero_b2)
```

```
[1] 0.9516
```

(iii)

Our research design has a false negative, or a Type II error. This suggest that our simulation has insufficient error. In sum, this simulation is likely not *sensitive* enough to detect real effects generated by the DGS.

```
interaction_ci <- mean(pset4_loop$b3_lower <= 0 & pset4_loop$b3_upper >= 0)

print(interaction_ci)
```

```
[1] 0
```

(iv)

For β_3 the proportion of p-values less than 0.005 is .048. For β_2, it is 1.

```
b2_pvalue <- 1 - mean(pset4_loop$b2_lower <= 0 & pset4_loop$b2_upper >= 0)
b3_pvalue <- 1 - mean(pset4_loop$b3_lower <= 0 & pset4_loop$b3_upper >= 0)

print(b3_pvalue)
```

```
[1] 1
```

```r
print(b2_pvalue)
```

```
[1] 0.0484
```

(e)

```r
set.seed(4052025)

beta_1_out_1200 <- c()
beta_2_out_1200 <- c()
beta_3_out_1200 <- c()
b1_lower_1200 <- c()
b1_upper_1200 <- c()
b2_lower_1200 <- c()
b2_upper_1200 <- c()
b3_lower_1200 <- c()
b3_upper_1200 <- c()


for(i in 1:5000){
  increase <- pset4_function(n=1200, b1= 10, b2= 0, b3= 2, usigma= 5)
  beta_1_out[i] <- increase$coefficients["x"]
  beta_2_out[i] <- increase$coefficients["z"]
  beta_3_out[i] <- increase$coefficients["x:z"]

  tidy_increase <- tidy(increase, conf.int = TRUE)
  b1_lower[i] <- tidy_increase$conf.low[2]
  b1_upper[i] <- tidy_increase$conf.high[2]
  b2_lower[i] <- tidy_increase$conf.low[3]
  b2_upper[i] <- tidy_increase$conf.high[3]
  b3_lower[i] <- tidy_increase$conf.low[4]
  b3_upper[i] <- tidy_increase$conf.high[4]

}

pset4_increase <- as.data.frame(cbind(beta_1_out, beta_2_out, beta_3_out, b1_lower, b1_upper, b
```

(f)

(i)

They're lower, which makes sense because as we collect, or generate more data, the data becomes more consistent and has more explaining power.m

```
b1_twelve_sd <- sd(pset4_increase$beta_1_out)
b2_twelve_sd <- sd(pset4_increase$beta_2_out)
b3_twelve_sd <- sd(pset4_increase$beta_3_out)

sd_df_1200 <- data.frame(
  Coefficient = c("Beta_1", "Beta_2", "Beta_3"),
  Standard_Deviation = c(b1_twelve_sd, b2_twelve_sd, b3_twelve_sd)
)

print(sd_df_1200)
```

```
  Coefficient Standard_Deviation
1      Beta_1          0.7886520
2      Beta_2          0.1032036
3      Beta_3          0.1463645
```

(ii)

They're the same. This is likely because the variance is low, and the same size was already large enough. It also because of underlying theories, or realities of the data generating process. Which again speaks to low variance.

```
b2_twelve_ci <- mean(pset4_increase$b2_lower <= 0 & pset4_increase$b2_upper >= 0)
b3_twelve_ci <- mean(pset4_increase$b3_lower <= 0 & pset4_increase$b3_upper >= 0)

ci_df_1200 <- data.frame(
  Coefficient = c("Beta_2", "Beta_3"),
  Standard_Deviation = c(b2_twelve_ci, b3_twelve_ci)
)

print(ci_df_1200)
```

```
  Coefficient Standard_Deviation
1      Beta_2             0.9506
2      Beta_3             0.0000
```

(g)

```
set.seed(4052025)

beta_1_out <- c()
beta_2_out <- c()
beta_3_out <- c()
```

```
b1_lower <- c()
b1_upper <- c()
b2_lower <- c()
b2_upper <- c()
b3_lower <- c()
b3_upper <- c()


for(i in 1:5000){
  error_increase <- pset4_function(n=1200, b1= 10, b2= 0, b3= 2, usigma= 20)
  beta_1_out[i] <- out$coefficients["x"]
  beta_2_out[i] <- out$coefficients["z"]
  beta_3_out[i] <- out$coefficients["x:z"]

  tidy_error_increase <- tidy(error_increase, conf.int = TRUE)
  b1_lower[i] <- tidy_error_increase$conf.low[2]
  b1_upper[i] <- tidy_error_increase$conf.high[2]
  b2_lower[i] <- tidy_error_increase$conf.low[3]
  b2_upper[i] <- tidy_error_increase$conf.high[3]
  b3_lower[i] <- tidy_error_increase$conf.low[4]
  b3_upper[i] <- tidy_error_increase$conf.high[4]

}

pset4_error_increase <- as.data.frame(cbind(beta_1_out, beta_2_out, beta_3_out, b1_lower, b1_up
```

(i)

The standard errors actually increased, from $n = 1200$, but are now similar to when $n = 500$. This is because while we now have more data, the increase in error variance increases the standard error. The higher variance is setting off having more data. The reason why these things cancel out is because in OLS, the *se* of an estimator uses both sample size and variance.

```
b1_higherror_sd <- sd(pset4_loop$beta_1_out)
b2_higherror_sd <- sd(pset4_loop$beta_2_out)
b3_higherror_sd <- sd(pset4_loop$beta_3_out)


print(b1_higherror_sd)
```

```
[1] 1.215804
```

```
print(b2_higherror_sd)
```

```
[1] 0.1593848
```

```r
print(b3_higherror_sd)
```

```
[1] 0.2265468
```

(ii)

No, there is no difference. The increase σ^2 had no effect.

```r
b2_ci_error <- mean(pset4_loop$b2_lower <= 0 & pset4_loop$b2_upper >= 0)
b3_ci_error <- mean(pset4_loop$b3_lower <= 0 & pset4_loop$b3_upper >= 0)

print(b2_ci_error)
```

```
[1] 0.9516
```

```r
print(b3_ci_error)
```

```
[1] 0
```

(iii)

The variance of our residuals mirrors the variance of the population error. Therefore, as the error variance changes, so in our case we upped it to 20, the residual variance would also change proportionally.