Problem Set 10

Daniel Shapiro

11/17/2022

Question 1 Background:

We'll use some data on crime to evaluate and correct for issues with our standard errors, using the ***prison.Rdata*** *dataset available on the course website. Note that there is not one right way to model this question - your answers can legitimately be different from one anothers', though the steps taken and the types of arguments made should be similar.*

```
load('prison.Rdata')
head(prison)
```

```
##    state year     criv     pris    polpc     incpc    unem  black  metro
## 1      1   80 4.447868 141.3347 235.0026  7673.176 0.08775 0.2560 0.6320
## 2      1   81 4.700944 163.5336 227.2263  8442.723 0.10667 0.2557 0.6362
## 3      1   82 4.497580 183.6948 219.4395  8855.808 0.14367 0.2554 0.6404
## 4      1   83 4.186833 218.6242 222.5979  9411.027 0.13667 0.2551 0.6446
## 5      1   84 4.353239 245.0686 214.3978 10267.680 0.11167 0.2548 0.6488
## 15     2   80 4.773632 132.0099 315.9204 13784.794 0.09592 0.0340 0.4340
##       ag0_14    ag15_17   ag18_24   ag25_34
## 1  0.2408834 0.05752439 0.1322547 0.1538264
## 2  0.2367951 0.05460577 0.1324317 0.1584588
## 3  0.2336306 0.05197452 0.1317197 0.1584713
## 4  0.2313167 0.04956787 0.1304016 0.1596340
## 5  0.2279858 0.04858300 0.1277834 0.1609312
## 15 0.2686567 0.05472637 0.1492537 0.2263682
```

1a) Test the hypothesis that the violent crime rate is linked to unemployment (`unem`), controlling for per capita income (`incpc`), prison population (`pris`), police presence (`polpc`), percent living in an urban area (`metro`), and the 4 age categories. Briefly interpret the coefficients from this regression, and describe whether the model meets the OLS assumptions. Evaluate the residuals using the plots produced by `autoplot`.

```
model <- lm(criv ~ unem + incpc + pris + polpc + metro + `ag0_14` + `ag15_17` + `ag18_24` + `ag25_34`, d
summary(model)
```

```
##
## Call:
## lm(formula = criv ~ unem + incpc + pris + polpc + metro + ag0_14 +
##     ag15_17 + ag18_24 + ag25_34, data = prison)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0832 -0.7387 -0.0022  0.5626  4.6461
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.788e+00  2.250e+00  -3.906 0.000121 ***
## unem         9.412e+00  3.664e+00   2.569 0.010802 *
## incpc       -1.890e-04  7.709e-05  -2.452 0.014913 *
## pris         1.275e-02  1.338e-03   9.531  < 2e-16 ***
## polpc        2.365e-02  1.542e-03  15.339  < 2e-16 ***
## metro        3.465e+00  4.533e-01   7.644 4.77e-13 ***
## ag0_14      -5.565e+00  4.598e+00  -1.210 0.227326
## ag15_17      7.469e+01  2.641e+01   2.828 0.005067 **
## ag18_24     -7.067e+00  1.274e+01  -0.555 0.579619
## ag25_34      1.844e+01  8.619e+00   2.139 0.033400 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.251 on 245 degrees of freedom
## Multiple R-squared:  0.848,  Adjusted R-squared:  0.8424
## F-statistic: 151.9 on 9 and 245 DF,  p-value: < 2.2e-16
```

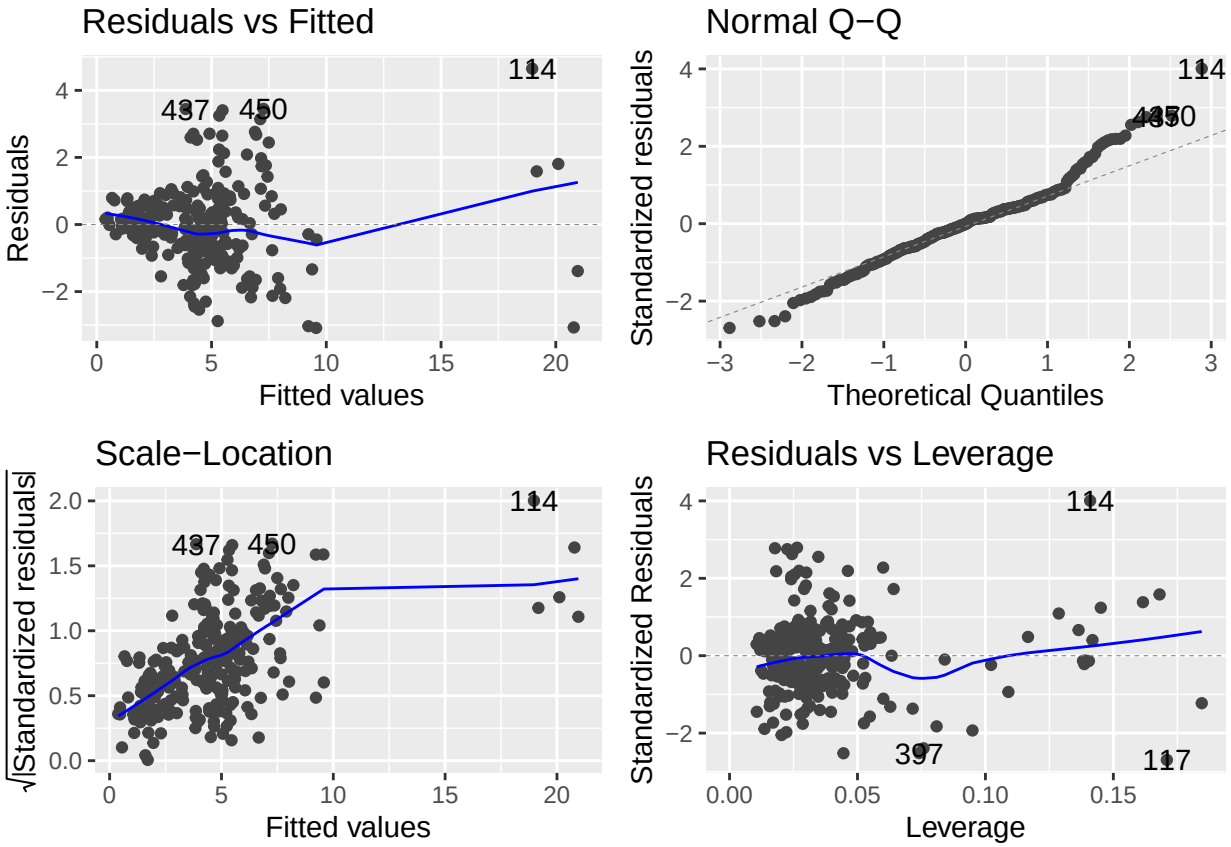
It looks from this regression that unemployment has a really strong relationship with violent crime – the coefficient is 9.4 or so. But that's a bit misleading; they're saying that for each "1" increase in unemployment, violent crime will go up by a bit over 9. But a "1" increase means a 100% increase in unemployment, because of the way that the data is structured. So really the coefficient is much lower. Nonetheless though, we can see that higher unemployment does tend to mean more violent crime. Honestly, many of these coefficients are a bit irritating to interpret; the coefficient for "incpc" measures what an increase in a single dollar does to violent crime. This stat is too minute; it would be better to look at it by thousands or by tens of thousands or something.

The main thing we should look at is the signs and the significance. There's a negative relationship between income and violent crime, and pretty strong positive correlations between prisons, police presence, and living in a city. There are mixed results for age.

The model doesn't really fit a lot of the OLS assumptions. One major one that we can see right off the bat without running data analysis is the endogeneity/exogeneity question. There are a lot of variables in here that are potentially endogenous. Even unemployment and violent crime rate could be the other way around – it could be that a high violent crime rate reduces the likelihood that businesses will move to the area, lowering the number of jobs available and thus increasing unemployment. High police presence could be *because* of high crime rates, not the other way around. A high prison population could be *because* of high violent crime rates, not the other way around. There are a lot of variables that should really be explored further.

Let's look at some of the autoplots.

```
autoplot(model)
```



Looking at the first graph, it looks like the linearity assumption largely holds, but I'm a bit worried about the skew toward the right side of the graph. I can't tell about the second one (random sampling) – I don't know the methodology of the paper. Variation in X seems fine; there are a ton of different values.

Homoskedasticity – The data definitely does **not** look homoskedastic. The scale-location plot shows this. The line does go horizontal eventually, but only for a few points, and there is a non-horizontal line for the most part.

Normality of errors – there are a few outliers, but in the grand scheme of things, it could be worse. I'm looking at Normal Q-Q and the residuals vs leverage graphs. The one thing that worries me is the jog upwards in the Q-Q graph towards the end. But honestly all things considered, it could be worse, in my opinion.

1b) Evaluate your data to see whether any of your variables should be log transformed. Re-run the model with any transformations that you see as useful/necessary. Reevaluate the model for normality of errors after making these changes. Briefly interpret the coefficients.

I looked at histograms of all the variables to see which ones were heavily skewed one way or another with a ton of outliers. In my opinion, there were three variables that should probably be log transformed: criv, pris, and polpc. All have fairly significant outliers.

Below, I show what log transforming these does to the data.

```
prison_log <- prison %>%
  mutate(logcriv = log(criv)) %>%
  mutate(logpris = log(pris)) %>%
  mutate(logpolpc = log(polpc)) %>%
```

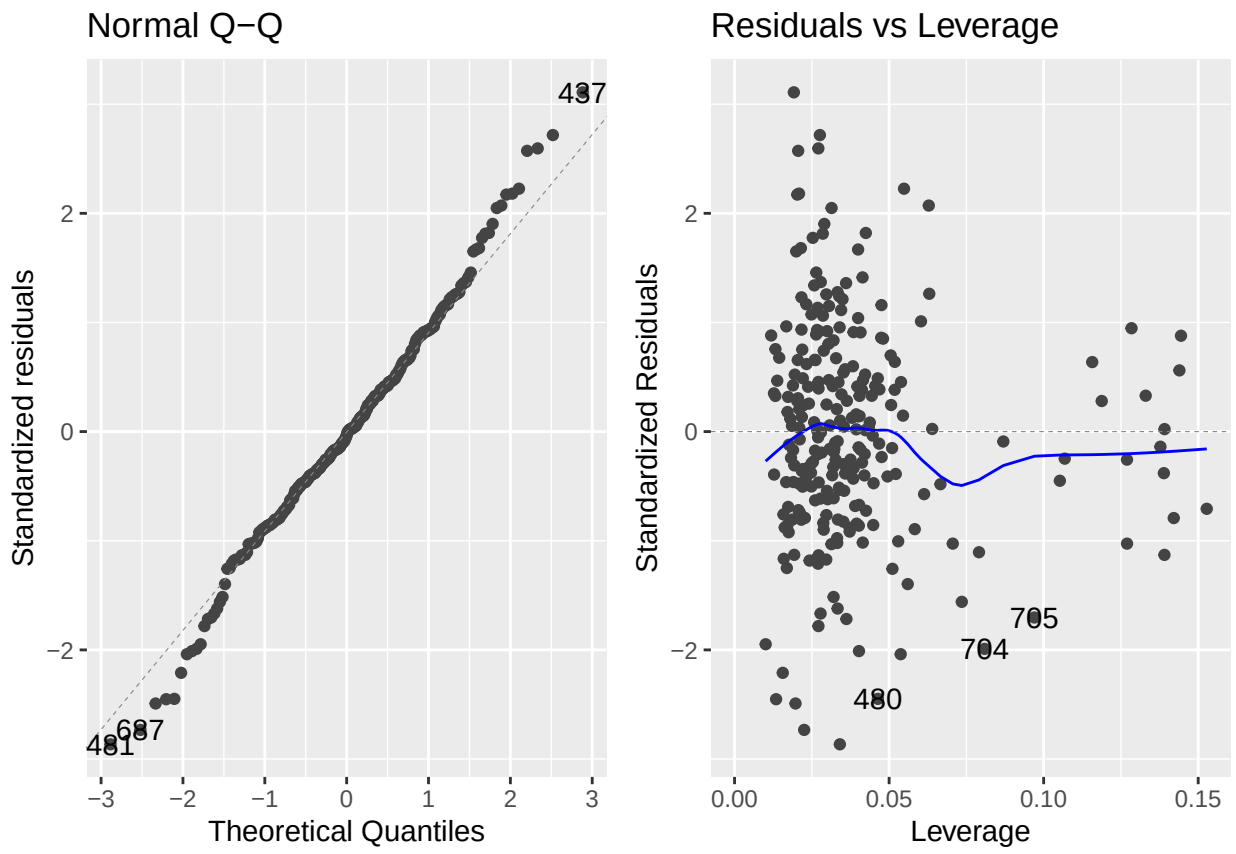
```
  select(-c(criv, pris, polpc))

log_model <- lm(logcriv ~ unem + incpc + logpris + logpolpc + metro + `ag0_14` + `ag15_17` + `ag18_24` +
summary(log_model)
```

```
##
## Call:
## lm(formula = logcriv ~ unem + incpc + logpris + logpolpc + metro +
##     ag0_14 + ag15_17 + ag18_24 + ag25_34, data = prison_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.73679 -0.15905 -0.00941  0.15520  0.80556
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.497e+00  6.315e-01 -13.454  < 2e-16 ***
## unem         1.626e+00  7.723e-01   2.105 0.036303 *
## incpc       -3.142e-05  1.613e-05  -1.948 0.052501 .
## logpris      5.530e-01  3.698e-02  14.955  < 2e-16 ***
## logpolpc     9.347e-01  1.036e-01   9.020  < 2e-16 ***
## metro        1.151e+00  9.626e-02  11.955  < 2e-16 ***
## ag0_14       6.372e-01  9.444e-01   0.675 0.500510
## ag15_17      3.206e+01  5.454e+00   5.878 1.35e-08 ***
## ag18_24     -8.956e+00  2.618e+00  -3.421 0.000731 ***
## ag25_34      5.432e+00  1.790e+00   3.034 0.002673 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2616 on 245 degrees of freedom
## Multiple R-squared:  0.8308, Adjusted R-squared:  0.8246
## F-statistic: 133.7 on 9 and 245 DF,  p-value: < 2.2e-16
```

Looking at the coefficients, it looks like not too much changed. The ones that flipped signs were already non-significant in the intial model, so I'm not entirely worried about that. The model in general ended up with many more "three star" significant variables than the initial one. The coefficients are all different than in the initial one, but that's because I log transformed the dependent variable as well, so the exact numbers will naturally changed. I'm mostly concerned about a) significance and b) if too many of the signs completely shifted.

Let's run the error tests.

```
autoplot(log_model, c(2, 5))
```



Honestly, this looks a fair amount better in my opinion. It's still not perfect, but it definitely looks better than the initial model did.

1c) Implement White's heteroskedasticity consistent errors, and compare your results to those obtained through your chosen model in part b). Do any of our conclusions change? If so, how? Are heteroskedasticity-robust errors appropriate here?

```
hclog_model <- lm_robust(logcriv ~ unem + incpc + logpris + logpolpc + metro + `ag0_14` + `ag15_17` + `a
summary(hclog_model)
```

```
##
## Call:
## lm_robust(formula = logcriv ~ unem + incpc + logpris + logpolpc +
##     metro + ag0_14 + ag15_17 + ag18_24 + ag25_34, data = prison_log)
##
## Standard error type:  HC2
##
## Coefficients:
##               Estimate Std. Error  t value  Pr(>|t|)   CI Lower   CI Upper  DF
## (Intercept) -8.497e+00  0.6038682 -14.0706 2.345e-33 -9.686e+00 -7.307e+00 245
## unem         1.626e+00  0.6751727   2.4080 1.678e-02  2.960e-01  2.956e+00 245
## incpc       -3.142e-05  0.0000129  -2.4362 1.556e-02 -5.682e-05 -6.016e-06 245
## logpris      5.530e-01  0.0374070  14.7834 8.800e-36  4.793e-01  6.267e-01 245
## logpolpc     9.347e-01  0.1099048   8.5048 1.819e-15  7.182e-01  1.151e+00 245
```

```
## metro        1.151e+00  0.0932564  12.3407 1.563e-27  9.672e-01  1.335e+00 245
## ag0_14       6.372e-01  0.7668250   0.8309 4.068e-01 -8.732e-01  2.148e+00 245
## ag15_17      3.206e+01  4.9898135   6.4259 6.779e-10  2.224e+01  4.189e+01 245
## ag18_24     -8.956e+00  2.6306205  -3.4045 7.744e-04 -1.414e+01 -3.774e+00 245
## ag25_34      5.432e+00  1.4692506   3.6972 2.691e-04  2.538e+00  8.326e+00 245
##
## Multiple R-squared:  0.8308 ,    Adjusted R-squared:  0.8246
## F-statistic: 191.4 on 9 and 245 DF,  p-value: < 2.2e-16
```

The error terms definitely change; there are six that are lower than in the initial model and four that are higher. As we noted in lecture, however, White's estimator does not change our estimates $\hat{\beta}$. Here, the lm_robust() function uses HC2 errors, which according to the slides, should produce pretty similar results as the others.

My sense is that heteroskedasticity-robust errors are appropriate here. Jane's slides said: "There is very little reason *not* to use robust standard errors - so as a general practice, you should always use heteroskedasticity robust SEs." I'm taking this to heart. Plus, we objectively can see that there is at least some heteroskedasticity in the data due to the last autoplot() tests in the previous question, even though we minimized it at least a bit.

1d) Note that these data include multiple observations from each state. What are we assuming when not accounting for this fact? Adjust your regression results for this issue. What, if anything, has changed?

This is an interesting point. When we don't account for this fact, we are essentially assuming that the state doesn't really matter for the purposes of this experiment. So if I control for state, let's see what happens.

```
hclog_model_state <- lm_robust(logcriv ~ unem + incpc + logpris + logpolpc + metro + `ag0_14` + `ag15_1
summary(hclog_model_state)
```

```
##
## Call:
## lm_robust(formula = logcriv ~ unem + incpc + logpris + logpolpc +
##     metro + ag0_14 + ag15_17 + ag18_24 + ag25_34 + state, data = prison_log)
##
## Standard error type:  HC2
##
## Coefficients:
##               Estimate Std. Error t value  Pr(>|t|)   CI Lower   CI Upper  DF
## (Intercept) -8.026e+00  6.825e-01 -11.760 1.359e-25 -9.370e+00 -6.682e+00 244
## unem         1.524e+00  6.576e-01   2.318 2.129e-02  2.288e-01  2.819e+00 244
## incpc       -3.754e-05  1.339e-05  -2.803 5.465e-03 -6.393e-05 -1.116e-05 244
## logpris      5.370e-01  3.778e-02  14.215 8.199e-34  4.626e-01  6.114e-01 244
## logpolpc     9.109e-01  1.088e-01   8.372 4.452e-15  6.966e-01  1.125e+00 244
## metro        1.159e+00  9.286e-02  12.479 5.687e-28  9.759e-01  1.342e+00 244
## ag0_14       7.321e-01  7.180e-01   1.020 3.089e-01 -6.821e-01  2.146e+00 244
## ag15_17      2.989e+01  5.133e+00   5.822 1.820e-08  1.978e+01  4.000e+01 244
## ag18_24     -9.029e+00  2.556e+00  -3.532 4.922e-04 -1.406e+01 -3.994e+00 244
## ag25_34      5.440e+00  1.416e+00   3.842 1.557e-04  2.651e+00  8.230e+00 244
## state       -3.651e-03  1.225e-03  -2.980 3.170e-03 -6.063e-03 -1.238e-03 244
##
## Multiple R-squared:  0.8374 ,    Adjusted R-squared:  0.8308
## F-statistic: 176.4 on 10 and 244 DF,  p-value: < 2.2e-16
```

This really doesn't do all that much of anything. Also, the coefficient for state is essentially useless; the state variable is coded as numbers, so this just shows that apparently states that get coded with higher numbers on average tend to have higher crime rates than states coded with lower numbers. Since we don't know what all goes into the coding, it doesn't really tell us anything decipherable at the moment.

1e) Find your clustered standard errors using a block bootstrap with 1,000 simulations. Briefly discuss whether, how, and why your standard errors change from part d).

I'm going to base this off of the example we did in class.

```
set.seed(6800)

coefv <- c()

for (i in 1:1000){

  s_cluster <- sample(unique(prison_log$state),
                      length(unique(prison_log$state)), replace = T)
  bb_data <- matrix(nrow = 0, ncol = length(colnames(prison_log)))
  colnames(bb_data) <- colnames(prison_log)

for (cluster in s_cluster){
  new_data <- subset(prison_log, state == cluster)
  bb_data <- rbind(bb_data, new_data)
}
  mod <- lm(logcriv ~ unem + incpc + logpris + logpolpc + metro + `ag0_14` + `ag15_17` + `ag18_24` + `ag
  coefv <- c(coefv, mod$coefficients[2])
}
```

```
sd(coefv)
```

```
## [1] 1.058911
```

This is the SE value for unemployment (unem). It's a bit bigger than the corresponding value for 1d, meaning that the estimate is a bit more conservative. There's a pretty small number of clusters and a relatively high level of within-cluster correlation, so that could explain this outcome.

Question 2 Background:

We will revisit the `x.csv` *data we used previously. In this problem, we are going to explore the case when the error term is not normally distributed and the case of heteroskedasticity. We keep the same population regression model:*

$$y_i = 3 + 5x_i + u_i$$

except this time we generate u from an exponential distribution with rate=0.5. We will also need to subtract 2 from each u to make sure it has the expectation of zero. Treat x as fixed, meaning your random samples should use the same x with different errors.

2a) Simulate the sampling distributions for $\hat{\beta}_0$ and $\hat{\beta}_1$ by doing the following 1,000 times:

1. Generate random errors from u from an exponential distribution with rate =0.5 and subtract 2 from each u
2. Generate values for y using u, the fixed x, and the true population parameters
3. Run a regression of y on x
4. Record your OLS estimates for coefficients and standard errors
5. Repeat

```
data2 <- read.csv("x.csv")
```

This is very similar to question 1 in Problem Set 8, but the "u" is created in a different way. I use similar code from that with the new parameters for u.

```
# Create empty dataframe

dataframe <- data.frame(matrix(ncol = 4, nrow = 1000))

for(i in 1:1000){
maindata <- data2 %>%
  mutate(u = rexp(1000, rate = 0.5) - 2) %>%
  mutate(y = (3 + 5*`x` + `u`))

regression <- lm(y ~ x, data = maindata)

m <- tidy(regression)

dataframe$X1[i] <- m[1,2][[1]]

dataframe$X2[i] <- m[2,2][[1]]

dataframe$X3[i] <- m[1,3][[1]]

dataframe$X4[i] <- m[2,3][[1]]
}
```

Compare your sampling distribution and the standard errors you get from your regression. How do the two compare?

```
sd(dataframe$X1); mean(dataframe$X3)
```

```
## [1] 0.06409334
```

```
## [1] 0.06324416
```

```
sd(dataframe$X2); mean(dataframe$X4)
```

```
## [1] 0.06551653
```

```
## [1] 0.06307289
```

The standard deviations of the sampling distributions are fairly close to those of the mean of our regressions' standard errors.

2b) Repeat a), but with $n = 5$ (using only the first 5 observations for x). Compare your results with part a), and why you see any changes.

Again, this is pretty similar to what we did on Problem Set 8, but the errors are different.

```
dataframe2 <- data.frame(matrix(ncol = 4, nrow = 1000))

for(i in 1:1000){
maindata <- data2 %>%
  mutate(u = rexp(1000, rate = 0.5) - 2) %>%
  mutate(y = (3 + 5*`x` + `u`))

newbdata <- maindata[1:5,]

regression <- lm(y ~ x, data = newbdata)

m <- tidy(regression)

dataframe2$X1[i] <- m[1,2][[1]]

dataframe2$X2[i] <- m[2,2][[1]]

dataframe2$X3[i] <- m[1,3][[1]]

dataframe2$X4[i] <- m[2,3][[1]]
}
```

Now, we see how the errors match up.

```
sd(dataframe2$X1); mean(dataframe2$X3)
```

```
## [1] 0.8502147
```

```
## [1] 0.741634
```

```
sd(dataframe2$X2); mean(dataframe2$X4)
```

```
## [1] 1.311021
```

```
## [1] 1.107638
```

These errors are much farther away from one another. This makes sense, given that our sample size is so much smaller.

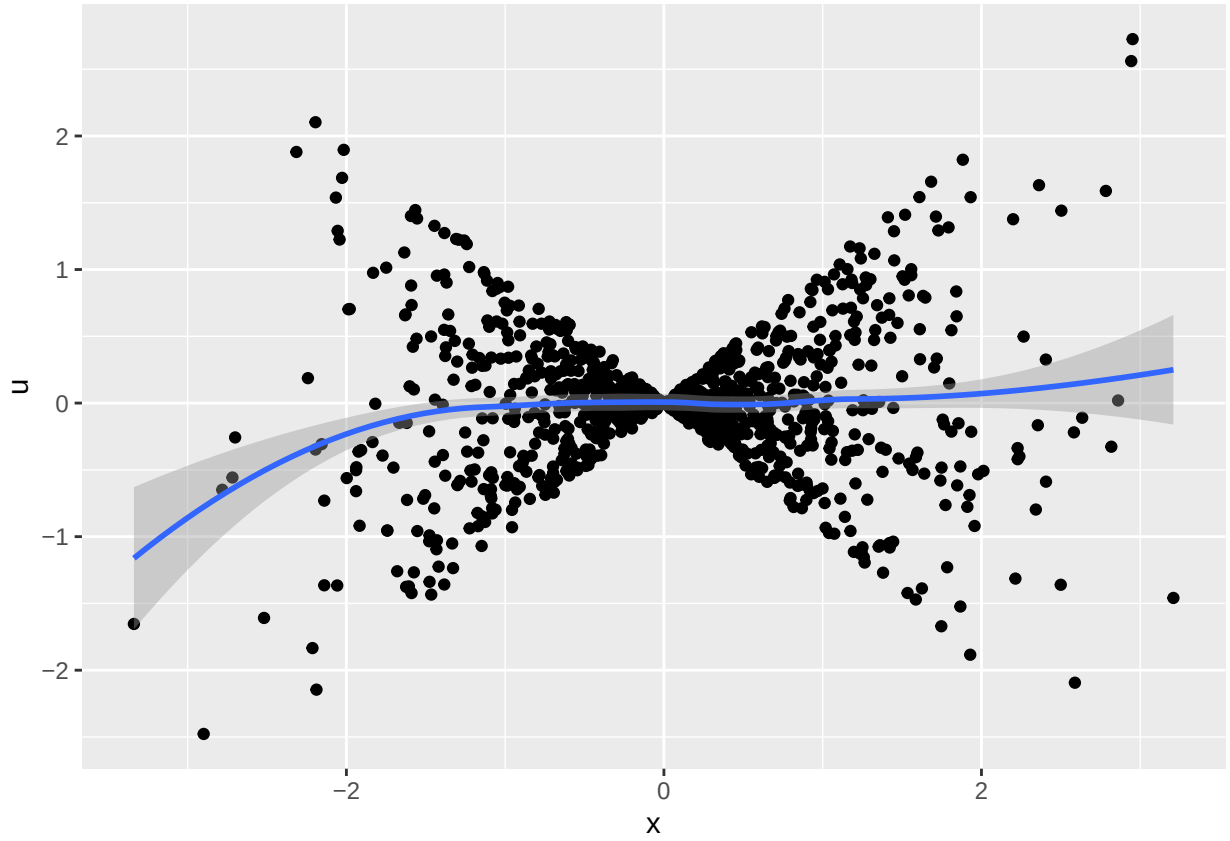
2c) Now we create a set of values u such that your errors will be heteroskedastic - the dispersion of the error term is a function of x while the mean is zero (so as to satisfy the zero mean assumption):

```
x <- read.csv('x.csv')$x
u <- runif(length(x), -abs(x), abs(x))

cdata <- cbind(x, u)
cdata <- as.data.frame(cdata)
```

Plot your errors against the x values to demonstrate that your errors are indeed heteroskedastic. Repeat part (a) and (b) using this new u. Describe your results. Why are you getting results like this?

```
ggplot(cdata, aes(x = x, y = u)) +
  geom_point() +
  geom_smooth(method = "loess")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



The errors are indeed heteroskedastic.

Now, we're again going to do similar stuff as in Problem Set 8.

```
dataframe3 <- data.frame(matrix(ncol = 4, nrow = 1000))

for(i in 1:1000){
maindata <- data2 %>%
  mutate(u = runif(length(x), -abs(x), abs(x))) %>%
  mutate(y = (3 + 5*`x` + `u`))

regression <- lm(y ~ x, data = maindata)

m <- tidy(regression)

dataframe3$X1[i] <- m[1,2][[1]]

dataframe3$X2[i] <- m[2,2][[1]]

dataframe3$X3[i] <- m[1,3][[1]]

dataframe3$X4[i] <- m[2,3][[1]]
}
```

```
sd(dataframe3$X1); mean(dataframe3$X3)
```

```
## [1] 0.01810246

## [1] 0.0183147
```

```
sd(dataframe3$X2); mean(dataframe3$X4)
```

```
## [1] 0.03239961

## [1] 0.0182651
```

```
dataframe4 <- data.frame(matrix(ncol = 4, nrow = 1000))

for(i in 1:1000){
maindata <- data2 %>%
  mutate(u = runif(length(x), -abs(x), abs(x))) %>%
  mutate(y = (3 + 5*`x` + `u`))

newcdata <- maindata[1:5,]

regression <- lm(y ~ x, data = newcdata)

m <- tidy(regression)

dataframe4$X1[i] <- m[1,2][[1]]

dataframe4$X2[i] <- m[2,2][[1]]

dataframe4$X3[i] <- m[1,3][[1]]

dataframe4$X4[i] <- m[2,3][[1]]
}
```

```
sd(dataframe4$X1); mean(dataframe4$X3)
```

```
## [1] 0.1690043
```

```
## [1] 0.1375847
```

```
sd(dataframe4$X2); mean(dataframe4$X4)
```

```
## [1] 0.3614931
```

```
## [1] 0.2054842
```

The errors overall get smaller compared to their counterparts in part a) and b), because we've changed our parameters for how the error terms are constructed. But what's mainly interesting is that heteroskedastic errors affect $\hat{\beta}_1$ much more than for $\hat{\beta}_0$.

2d) Repeat part c) using heteroskedasticity-robust standard errors for your regression of y on x. Compare the standard errors from your regressions to those derived from the bootstrapped sampling distribution. How do your results differ from part c)? Why?

```
dataframe5 <- data.frame(matrix(ncol = 4, nrow = 1000))

for(i in 1:1000){
maindata <- data2 %>%
  mutate(u = runif(length(x), -abs(x), abs(x))) %>%
  mutate(y = (3 + 5*`x` + `u`))

regression <- lm_robust(y ~ x, data = maindata)

m <- tidy(regression)

dataframe5$X1[i] <- m[1,2][[1]]

dataframe5$X2[i] <- m[2,2][[1]]

dataframe5$X3[i] <- m[1,3][[1]]

dataframe5$X4[i] <- m[2,3][[1]]
}
```

```
sd(dataframe5$X1); mean(dataframe5$X3)
```

```
## [1] 0.01900123
```

```
## [1] 0.01825068
```

```
sd(dataframe5$X2); mean(dataframe5$X4)
```

```
## [1] 0.03108495
```

```
## [1] 0.0317489
```

```
dataframe6 <- data.frame(matrix(ncol = 4, nrow = 1000))

for(i in 1:1000){
maindata <- data2 %>%
  mutate(u = runif(length(x), -abs(x), abs(x))) %>%
  mutate(y = (3 + 5*`x` + `u`))

newddata <- maindata[1:5,]

regression <- lm_robust(y ~ x, data = newddata)

m <- tidy(regression)

dataframe6$X1[i] <- m[1,2][[1]]

dataframe6$X2[i] <- m[2,2][[1]]

dataframe6$X3[i] <- m[1,3][[1]]

dataframe6$X4[i] <- m[2,3][[1]]
}
```

```
sd(dataframe6$X1); mean(dataframe6$X3)
```

```
## [1] 0.1692862
```

```
## [1] 0.1384096
```

```
sd(dataframe6$X2); mean(dataframe6$X4)
```

```
## [1] 0.3566093
```

```
## [1] 0.2357734
```

The main difference here is that the errors for the first model (with the high n) for $\hat{\beta}_1$ end up evening out. This is because we use the robust modeling. It doesn't work as well for the second model, because we use a much smaller value of n.