Problem Set 11

Daniel Shapiro

11/30/2022

Question 1 Background:

In this problem you will be replicating Jens Hainmueller, Jonathan Mummolo, and Yiqing Xu (2019), "How Much Should We Trust Estimates from Multiplicative Interaction Models? Simple Tools to Improve Empirical Practice," Political Analysis (HMX). I will ask you to recreate the plots and analyses using your own original code.

Your results will not be exactly the same, due to choices of bins, your random seed, and other modeling choices. You do not have to include controls, for example. However, you should show the same broad results, and explain the significance of those results.

1a) Recreate the simulated datasets introduced by HMX on page 168. Run a regression of your Y variables on X, D, and an interaction of X and D. Report the results in a nicely formatted table, and briefly describe what we can learn from the tables.

HMX produce three datasets on page 168. The first two follow this formula:

$$Y_i = 5 - 4X_i - 9D_i + 3D_iX_i + \epsilon_i$$

The last one follows this formula:

$$Y_i = 2.5 - X_i^2 - 5D_i + 2D_iX_i^2 + \zeta_i$$

Now, let's define the parameters for the first two equations, as reported by the article.

```r
# There are 200 rows.

X1 <- rnorm(200, mean = 3, sd = 1)
Error1 <- rnorm(200, mean = 0, sd = 4)
D1 <- rbinom(200, 1, 0.5)

first <- as.data.frame(cbind(X1, D1, Error1)) %>%

# Add an extra empty vector.

  mutate(Y1 = 1)

# Run a for loop.

for(i in 1:200){
  first$Y1[i] <- 5 - 4*first$X1[i] - 9*first$D1[i] + 3*first$D1[i]*first$X1[i] + first$Error1[i]
}

# Setting up next equation

X2 <- rnorm(200, mean = 3, sd = 1)
Error2 <- rnorm(200, mean = 0, sd = 4)
D2 <- rnorm(200, mean = 3, sd = 1)

second <- as.data.frame(cbind(X2, D2, Error2)) %>%
  mutate(Y2 = 1)

for(i in 1:200){
  second$Y2[i] <- 5 - 4*second$X2[i] - 9*second$D2[i] + 3*second$D2[i]*second$X2[i] + second$Error2[i]
}
```

Now, we'll set up the next formula.

```r
X3 <- runif(200, min = -3, max = 3)
Error3 <- rnorm(200, mean = 0, sd = 4)
D3 <- rbinom(200, 1, 0.5)

third <- as.data.frame(cbind(X3, D3, Error3)) %>%
  mutate(Y3 = 1) %>%
  mutate(xsquared = X3^2)

for(i in 1:200){
  third$Y3[i] <- 2.5 - third$xsquared[i] - 5*third$D3[i] + 2*third$D3[i]*third$xsquared[i] + third$Erro
}
```

Now we will run regressions.

```r
# I tried to run robust because it's good practice, but stargazer doesn't
# like lm_robust().

reg1 <- lm(Y1 ~ X1 + D1 + X1*D1, data = first)
reg2 <- lm(Y2 ~ X2 + D2 + X2*D2, data = second)
reg3 <- lm(Y3 ~ X3 + D3 + X3*D3, data = third)

sum1 <- summary(reg1)
sum2 <- summary(reg2)
sum3 <- summary(reg3)

stargazer(reg1, reg2, reg3, type = "text")
```

```
##
## ============================================================
##                                      Dependent variable:
##                                 ------------------------------
##                                     Y1          Y2        Y3
##                                    (1)         (2)       (3)
## ------------------------------------------------------------
## X1                              -3.881***
##                                 (0.439)
##
## D1                            -11.738***
##                                 (2.068)
##
## X1:D1                          3.761***
##                                 (0.651)
##
## X2                                        -4.524***
##                                            (0.852)
##
## D2                                        -9.338***
##                                            (0.919)
##
## X2:D2                                      3.218***
##                                            (0.273)
##
## X3                                                     0.087
##                                                       (0.282)
##
## D3                                                     0.397
##                                                       (0.654)
##
## X3:D3                                                 -0.007
##                                                       (0.404)
##
## Constant                       4.910***    5.643*   -0.119
##                                (1.369)    (2.916)   (0.445)
##
## -----------------------------------------------------------
## Observations                     200        200       200
## R2                              0.287      0.704     0.003
## Adjusted R2                     0.276      0.699    -0.012
## Residual Std. Error (df = 196)  4.237      4.154     4.591
## F Statistic (df = 3; 196)     26.301***  155.217***  0.198
## ===========================================================
## Note:                          *p<0.1; **p<0.05; ***p<0.01
```

I put all of the regressions together into one table. I also renamed the variables to depend on the regression, because to be honest, I never trust R when you re-define one variable name over and over again. So I made sure that they were all distinctive, hence a table that looks a bit longer than perhaps it has to.

Anyway, let's look at the variables. At least for the first two, we can see that the coefficients at least are somewhat close to the coefficients laid out in the equation. There's a fair bit of error; this likely depends on the sampling process and the ways in which we set everything up. For the third one, the coefficients are much harder to compare to the original equation. It's mainly because we're really evaluate a Y ~ X + D + X*D relationship when the equation itself is polynomial, meaning that the coefficients are quite a bit different than what we see on page 168.

I think it would be different if we set up the third regression as Y ~ X^2 + D + X^2*D, but that's not what the question asked.

1b) Run the same regression as in part a), but this time do not include the lower-order X variable. How do your results change? Why? Which model do you prefer?

```r
reg4 <- lm(Y1 ~ D1 + X1:D1, data = first)
reg5 <- lm(Y2 ~ D2 + X2:D2, data = second)
reg6 <- lm(Y3 ~ D3 + X3:D3, data = third)

stargazer(reg4, reg5, reg6, type = "text")
```

```
##
## ============================================================
##                                   Dependent variable:
##                               -----------------------------
##                                  Y1         Y2        Y3
##                                  (1)        (2)       (3)
## ------------------------------------------------------------
## D1                             -0.143
##                               (1.886)
##
## D1:X1                          -0.120
##                               (0.566)
##
## D2                                      -4.908***
##                                          (0.413)
##
## D2:X2                                    1.843***
##                                          (0.094)
##
## D3                                                  0.408
##                                                    (0.651)
##
## D3:X3                                               0.080
##                                                    (0.289)
##
## Constant                      -6.685***  -8.999***  -0.131
##                               (0.462)    (1.014)   (0.443)
##
## ------------------------------------------------------------
## Observations                     200        200       200
## R2                              0.003      0.661     0.003
## Adjusted R2                    -0.007      0.658    -0.008
## Residual Std. Error (df = 197)  4.998      4.432     4.581
## F Statistic (df = 2; 197)       0.278    192.180***  0.250
## ============================================================
## Note:                          *p<0.1; **p<0.05; ***p<0.01
```

I changed the * to a :, meaning that R doesn't include the lower-order X automatically. When you take this out, the coefficients change. This is natural; the variables used are different so the coefficients will be different. I much prefer the first model; it provides a more comprehensive look at the effects of the variables involved, in my opinion.

1c) Write a function that takes a dataframe and a model with a dichotomous treatment variable D as an input and returns a dataframe with columns for the marginal effect of D on Y at different levels of X, X, and the confidence interval for the marginal effect at each level of X.

This took me forever and one misplaced comma could screw the whole thing up. So buckle your seatbelts; I hope this is right.

```
cfunc <- function(dataframe){

modeldata <- lm(data = dataframe, dataframe[,4] ~ dataframe[,1] + dataframe[,2] + dataframe[,1]*datafran

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

# Essentially, this first for loop marks the derivative function of the model
# in R (modeldata) and applies it to each value of X.

for(i in 1:nrow(return)){
  return$V1[i] <- modeldata$coefficients[3] + modeldata$coefficients[4]*dataframe$X[i]
}

# This is just X

for(i in 1:nrow(return)){
  return$V2[i] <- dataframe$X[i]
}

# Now, I find the lower bound of the confidence interval for the marginal effect.
# So I take Y, then subtract 1.96 * the standard error of the marginal effect,
# which I create by taking Var(Beta3) + Var(Beta4) * X^2 + 2 * X * cov(Beta3, Beta4).
# The vcov() function helps here.

for(i in 1:nrow(return)){
  return$V3[i] <- return$V1[i] - 1.96 * sqrt(as.data.frame(vcov(modeldata)[3,3]) + as.data.frame(vcov(mo
}

# Here, I do the same thing, but I add instead.

for(i in 1:nrow(return)){
  return$V4[i] <- return$V1[i] + 1.96 * sqrt(as.data.frame(vcov(modeldata)[3,3]) + as.data.frame(vcov(mo
}

return <- rename(return, marg.effect = V1, X = V2, conf.low = V3, conf.high = V4)

return
}
```

1d) Recreate the plots in HMX Figure 2, using the function you defined above. Explain, in your own words, the significance of the plots. Note that this will require (1) a linear marginal effects line with confidence intervals; (2) the marginal effects using a binned X variable; and (3) demonstration of the "support" of your X variable (note that it is difficult to add a transparent histogram to ggplot, but scatterplots/boxplots can provide the same information).

This also took me almost a whole day to do. I feel like there's got to be an easier way to do this; if there is, though I just don't know it.

```
testlinear <- cfunc(first)
```

```
# Pivoted to put all other values in one column. Then I can put all
# marginal effects values and confidence interval values in the plot.

testlinear$conf.low <- as.numeric(testlinear$conf.low)
testlinear$conf.high <- as.numeric(testlinear$conf.high)

testlinear2 = pivot_longer(testlinear, -X, names_to = "All", values_to = "value")

# I can't figure out how to boxplot() overlaid over the graph. So I'm going to represent
# binned marginal effects in a different way -- by splitting up the graphs and
# finding means of the marginal effects for each mean. Then I'll plot them.

lowdata <- testlinear2 %>%
  filter(All == "marg.effect") %>%
  arrange(X) %>%
  head(67)

lowmeanY <- sum(lowdata$value)/nrow(lowdata)
lowmeanX <- sum(lowdata$X)/nrow(lowdata)

middata <- testlinear2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

middata <- middata[68:134,]

midmeanY <- sum(middata$value)/nrow(middata)
midmeanX <- sum(middata$X)/nrow(middata)

highdata <- testlinear2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

highdata <- highdata[135:200,]

highmeanY <- sum(highdata$value)/nrow(highdata)
highmeanX <- sum(highdata$X)/nrow(highdata)
```

```
# Next I need to make another function that works for our non-linear model.
# The other one really only works for linear models. So I have to use the I()
# function to make the variables squared.

cfuncnonlin <- function(dataframe){
```

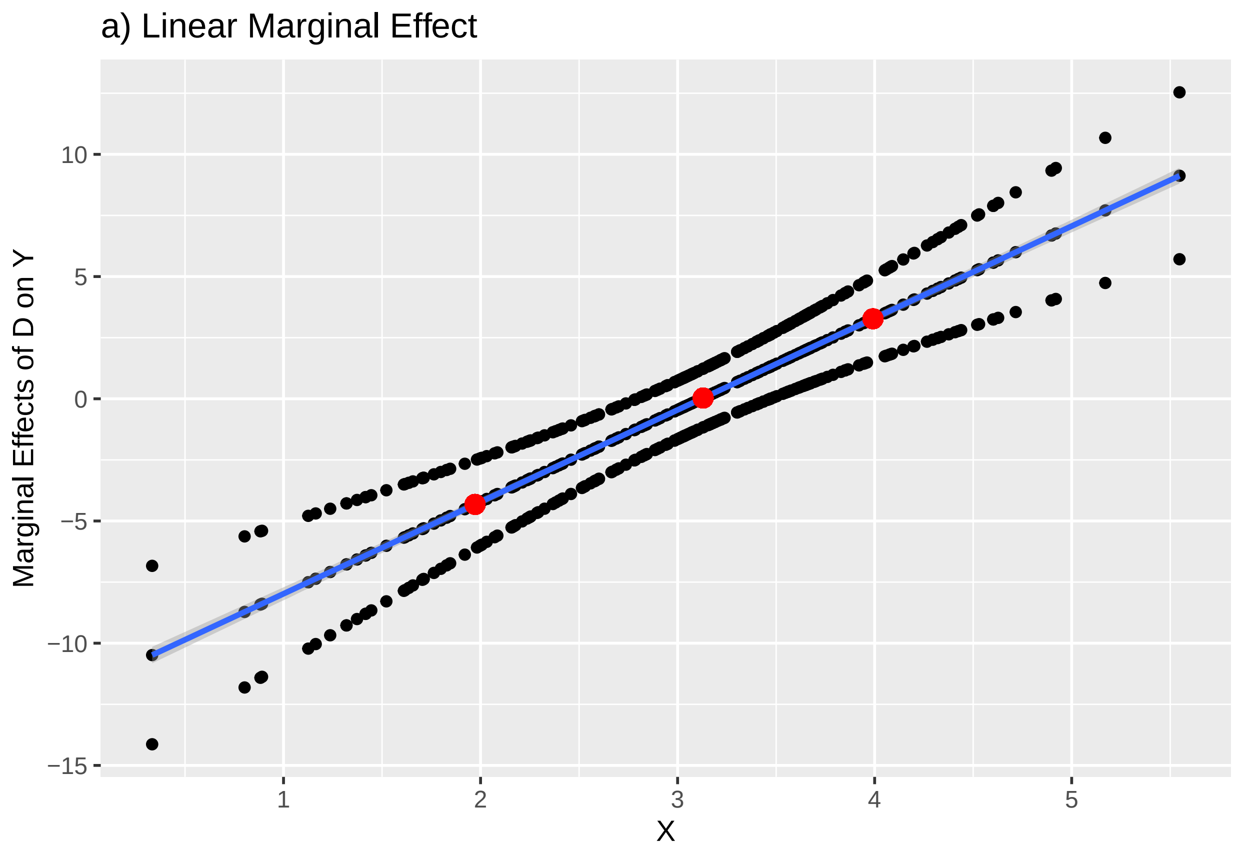
```
modeldata <- lm(data = dataframe, dataframe[,4] ~ I(dataframe[,1]^2) + dataframe[,2] + I(dataframe[,1]^

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

for(i in 1:nrow(return)){
  return$V1[i] <- modeldata$coefficients[3] + modeldata$coefficients[4]*I(dataframe$X[i]^2)
}

for(i in 1:nrow(return)){
  return$V2[i] <- dataframe$X[i]
}

for(i in 1:nrow(return)){
  return$V3[i] <- return$V1[i] - 1.96 * sqrt(as.data.frame(vcov(modeldata)[3,3]) + as.data.frame(vcov(m
}

for(i in 1:nrow(return)){
  return$V4[i] <- return$V1[i] + 1.96 * sqrt(as.data.frame(vcov(modeldata)[3,3]) + as.data.frame(vcov(m
}

return <- rename(return, marg.effect = V1, X = V2, conf.low = V3, conf.high = V4)

return
}
```

```
# Run the nonlinear function.

testnonlinear <- cfuncnonlin(third)
```

```
testnonlinear$conf.low <- as.numeric(testnonlinear$conf.low)
testnonlinear$conf.high <- as.numeric(testnonlinear$conf.high)

testnonlinear2 = pivot_longer(testnonlinear, -X, names_to = "All", values_to = "value")

lownldata <- testnonlinear2 %>%
  filter(All == "marg.effect") %>%
  arrange(X) %>%
  head(67)

lownlmeanY <- sum(lownldata$value)/nrow(lownldata)
lownlmeanX <- sum(lownldata$X)/nrow(lownldata)

midnldata <- testnonlinear2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

midnldata <- midnldata[68:134,]

midnlmeanY <- sum(midnldata$value)/nrow(midnldata)
midnlmeanX <- sum(midnldata$X)/nrow(midnldata)

highnldata <- testnonlinear2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

highnldata <- highnldata[135:200,]

highnlmeanY <- sum(highnldata$value)/nrow(highnldata)
highnlmeanX <- sum(highnldata$X)/nrow(highnldata)
```

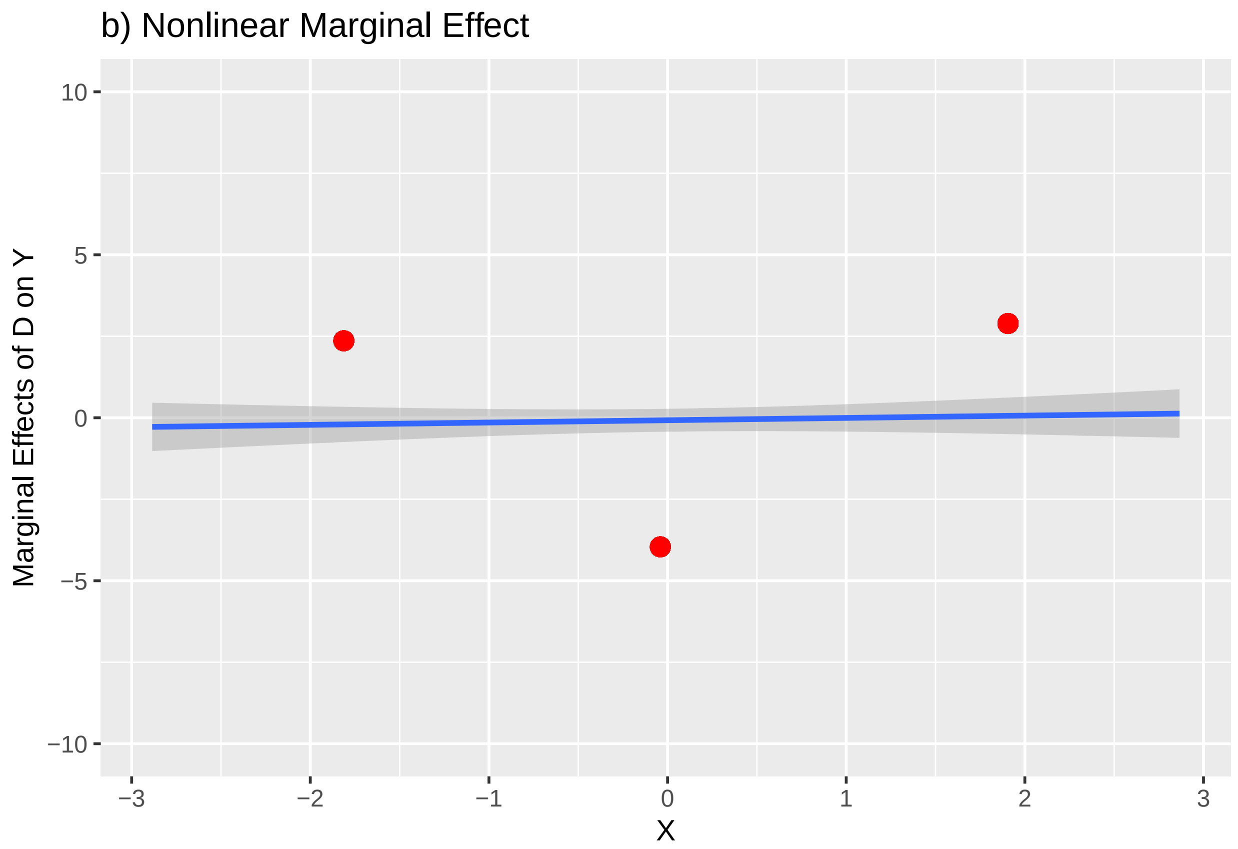
```
ggplot(testlinear2, aes(x = X, y = as.numeric(value))) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(y = "Marginal Effects of D on Y",
       title = "a) Linear Marginal Effect") +
  geom_point(aes(x = lowmeanX, y = lowmeanY), color = "red", size = 3) +
  geom_point(aes(x = midmeanX, y = midmeanY), color = "red", size = 3) +
  geom_point(aes(x = highmeanX, y = highmeanY), color = "red", size = 3)
```

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



```
ggplot(testnonlinear2, aes(x = X, y = as.numeric(value))) +
  geom_smooth(method = "lm") +
  labs(y = "Marginal Effects of D on Y",
       title = "b) Nonlinear Marginal Effect") +
  geom_point(aes(x = lownlmeanX, y = lownlmeanY), color = "red", size = 3) +
  geom_point(aes(x = midnlmeanX, y = midnlmeanY), color = "red", size = 3) +
  geom_point(aes(x = highnlmeanX, y = highnlmeanY), color = "red", size = 3) +
  ylim(-10, 10)
```

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

## Warning: Removed 26 rows containing non-finite values (stat_smooth).
```

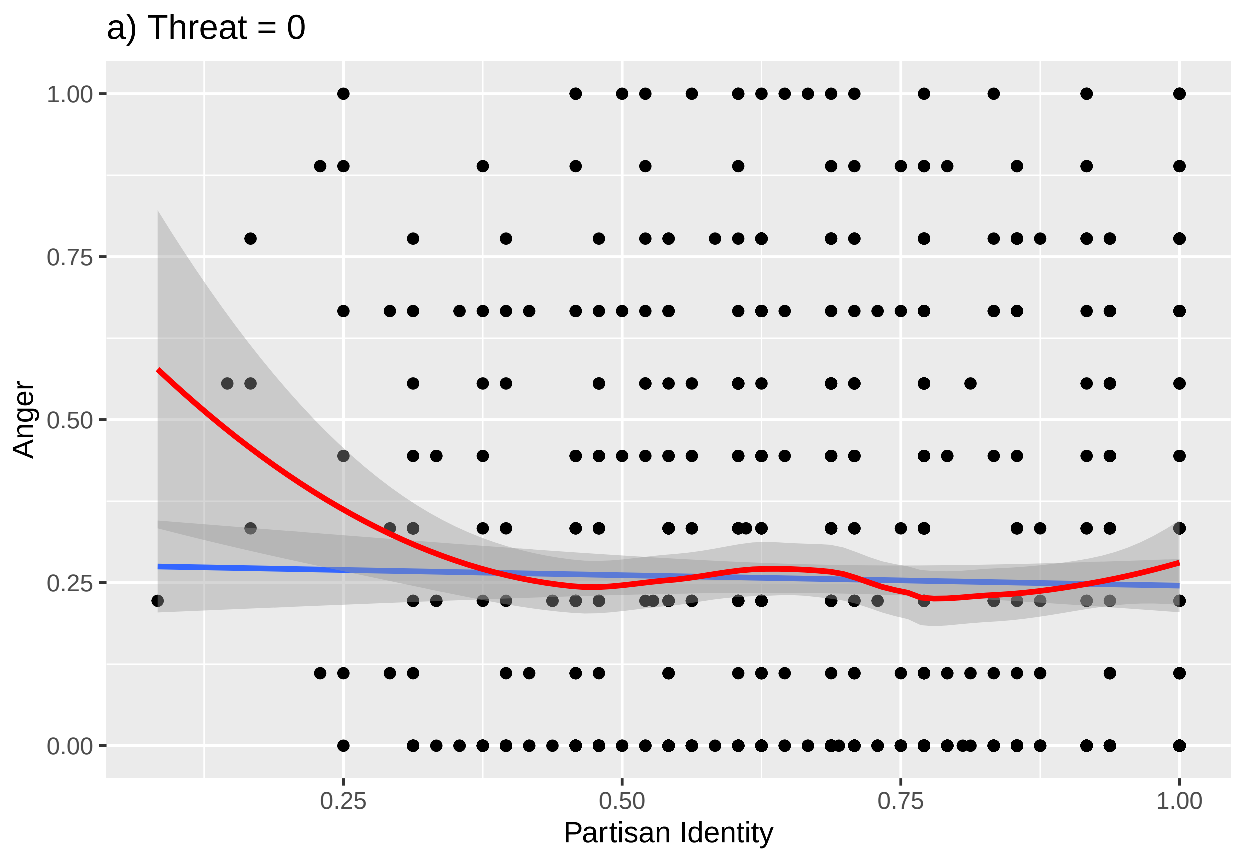


First of all, I'm honestly just happy that I could get my plots to look a fair amount like the ones in the paper. Some of their graphics look different than mine, but the basics are pretty similar. Regarding the marginal effects, we can see that using the lm line doesn't really do all that much for nonlinear models. The binned marginal effects (the red dots) follow the line for the linear model, but they don't at all for the nonlinear model. We need to use other models for this sort of effect.

1e) Using rep_huddy_2015a.dta, replicate Figures 4a and 4b of HMX ($Y = totangry, D = threat, X = pidentity$). Use robust standard errors. Explain, in your own words, the significance of this plot.

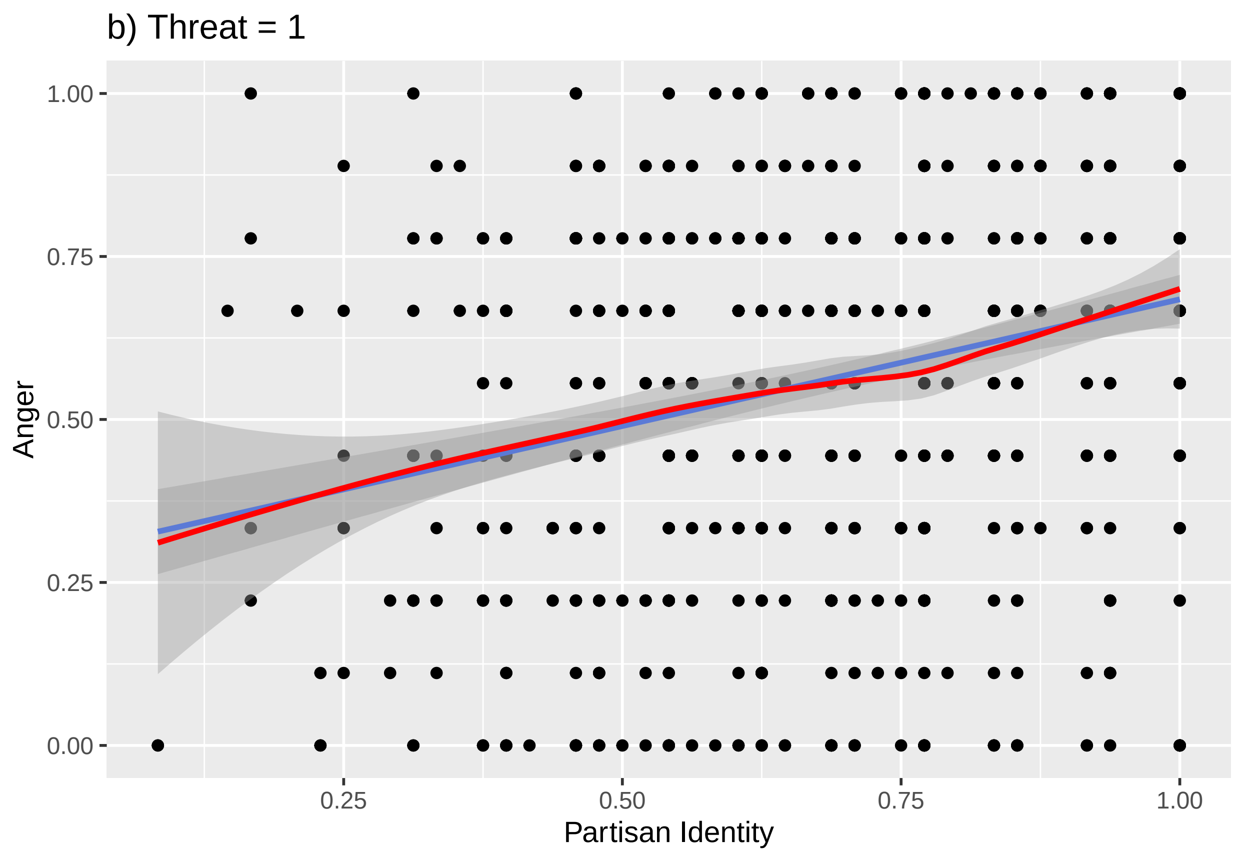
```
huddy <- read.dta("rep_huddy_2015a.dta")

huddy_model <- lm_robust(totangry ~ pidentity + threat + pidentity*threat, data = huddy)

plot1data <- huddy %>%
  select(c(pidentity, threat, totangry)) %>%
  filter(threat == 0)

plot2data <- huddy %>%
  select(c(pidentity, threat, totangry)) %>%
  filter(threat == 1)

ggplot(plot1data, aes(x = pidentity, y = totangry)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(method = "loess", color = "red") +
  labs(x = "Partisan Identity",
       y = "Anger",
       title = "a) Threat = 0")
```

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



```
ggplot(plot2data, aes(x = pidentity, y = totangry)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(method = "loess", color = "red") +
  labs(x = "Partisan Identity",
       y = "Anger",
       title = "b) Threat = 1")
```

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



Now, I'm going to do graph b. I'm going to do basically exactly what I did in the last question.

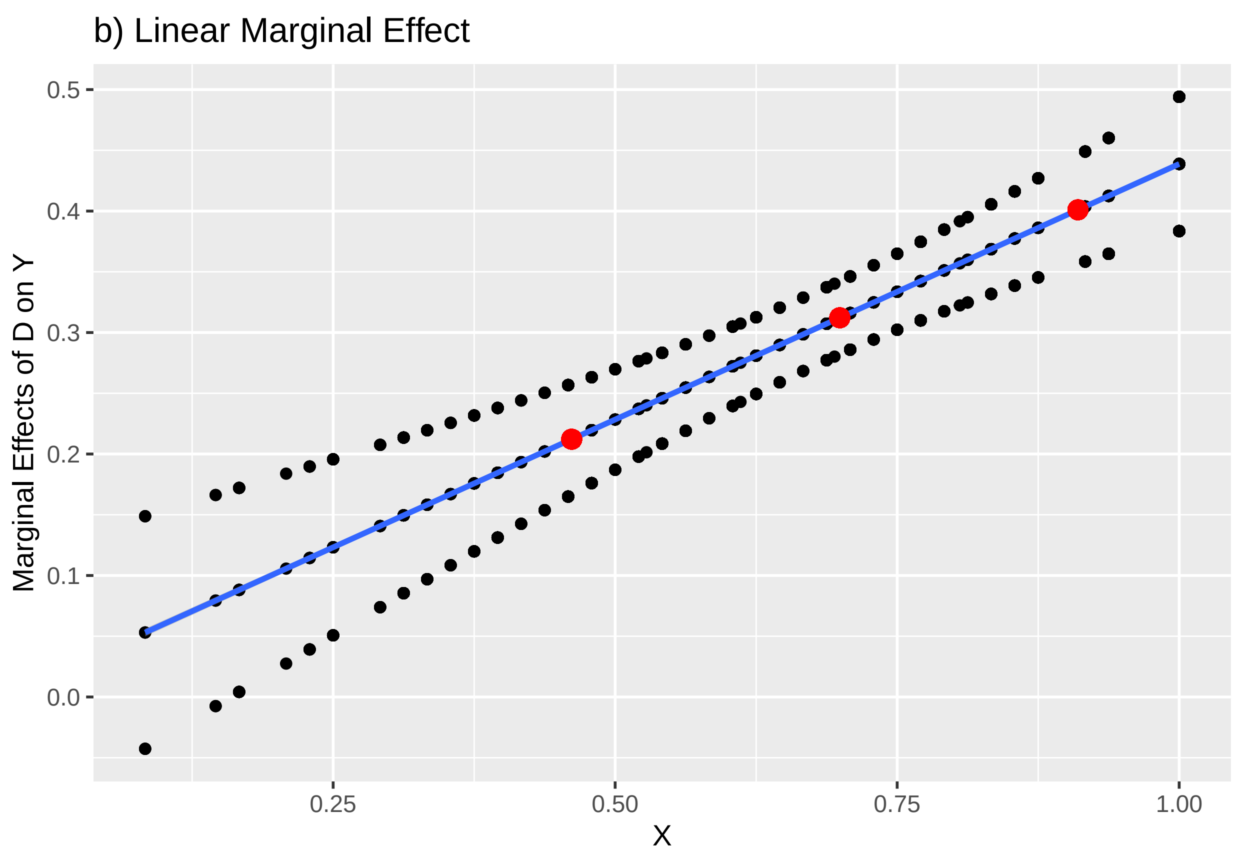
```
# First, I need to format it to fit the function. Had to put a random 3rd variable in.

func_huddy <- huddy %>%
  select(c(pidentity, threat, id_time, totangry))

func_huddy <- rename(func_huddy, X = pidentity, D = threat, Y = totangry)

test_huddy <- cfunc(func_huddy)

test_huddy$conf.low <- as.numeric(test_huddy$conf.low)
test_huddy$conf.high <- as.numeric(test_huddy$conf.high)
```

```
test_huddy2 <- pivot_longer(test_huddy, -X, names_to = "All", values_to = "value")

lowdata2 <- test_huddy2 %>%
  filter(All == "marg.effect") %>%
  arrange(X) %>%
  head(494)

lowmeanY2 <- sum(lowdata2$value)/nrow(lowdata2)
lowmeanX2 <- sum(lowdata2$X)/nrow(lowdata2)
```

```r
middata2 <- test_huddy2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

middata2 <- middata2[495:988,]

midmeanY2 <- sum(middata2$value)/nrow(middata2)
midmeanX2 <- sum(middata2$X)/nrow(middata2)

highdata2 <- test_huddy2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

highdata2 <- highdata2[989:1482,]

highmeanY2 <- sum(highdata2$value)/nrow(highdata2)
highmeanX2 <- sum(highdata2$X)/nrow(highdata2)
```

```r
ggplot(test_huddy2, aes(x = X, y = as.numeric(value))) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(y = "Marginal Effects of D on Y",
       title = "b) Linear Marginal Effect") +
  geom_point(aes(x = lowmeanX2, y = lowmeanY2), color = "red", size = 3) +
  geom_point(aes(x = midmeanX2, y = midmeanY2), color = "red", size = 3) +
  geom_point(aes(x = highmeanX2, y = highmeanY2), color = "red", size = 3)
```

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



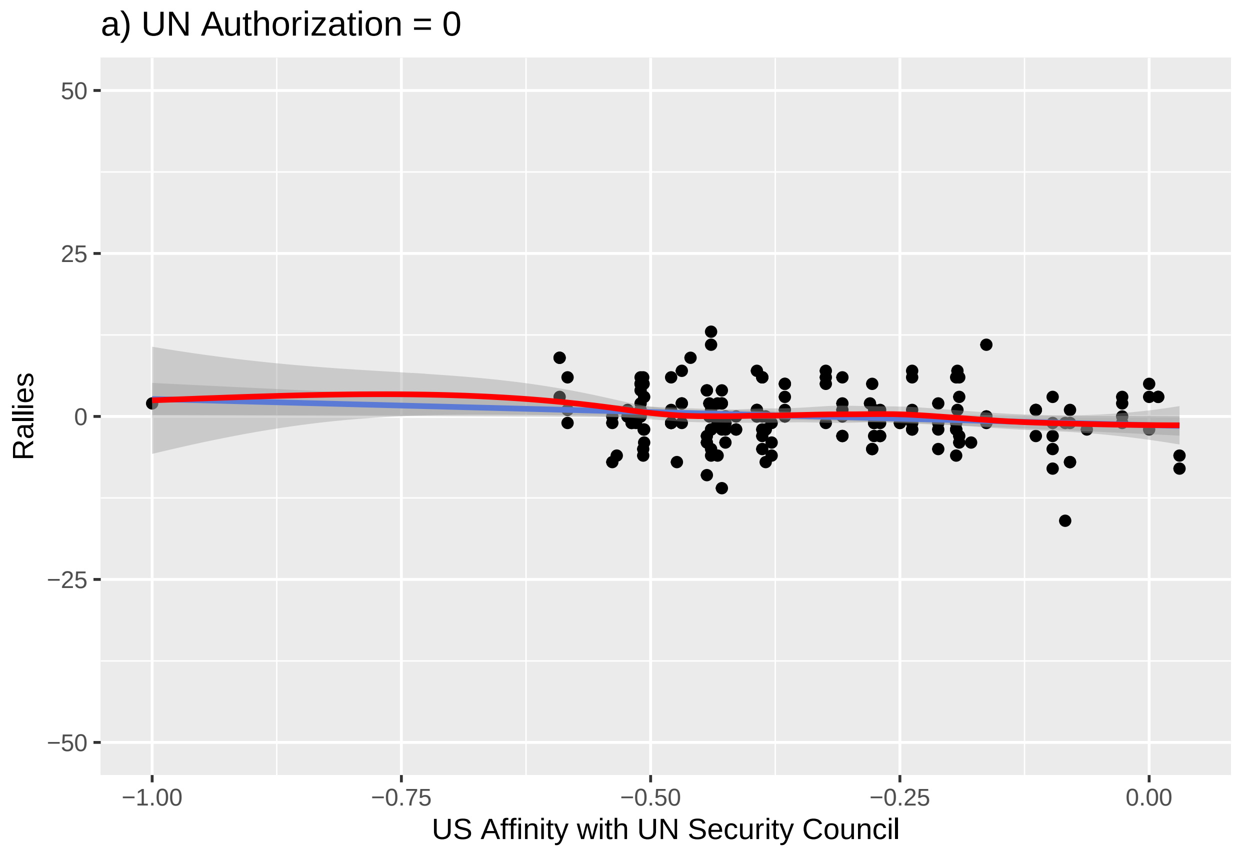
It looks like the marginal effects graph works pretty well linearly. Also, as the article points out, the middle point (from the middle bin) is basically right in between the high and low points. They're all skewed toward the right, but that's just because the data is generally skewed a bit more to the right.

1f) Using rep_chapman_2009.dta, replicate Figure 5a and 5b of HMX ($Y = rally, D = unauth, X = S$). Explain, in your own words, the significance of these plots.

```
chapman <- read.dta("rep_chapman_2009.dta")
```

```
plot3data <- chapman %>%
  select(c(rally, unauth, S)) %>%
  filter(unauth == 0)

plot4data <- chapman %>%
  select(c(rally, unauth, S)) %>%
  filter(unauth == 1)

ggplot(plot3data, aes(x = S, y = rally)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(method = "loess", color = "red") +
  labs(x = "US Affinity with UN Security Council",
       y = "Rallies",
       title = "a) UN Authorization = 0") +
  ylim(-50, 50)
```

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

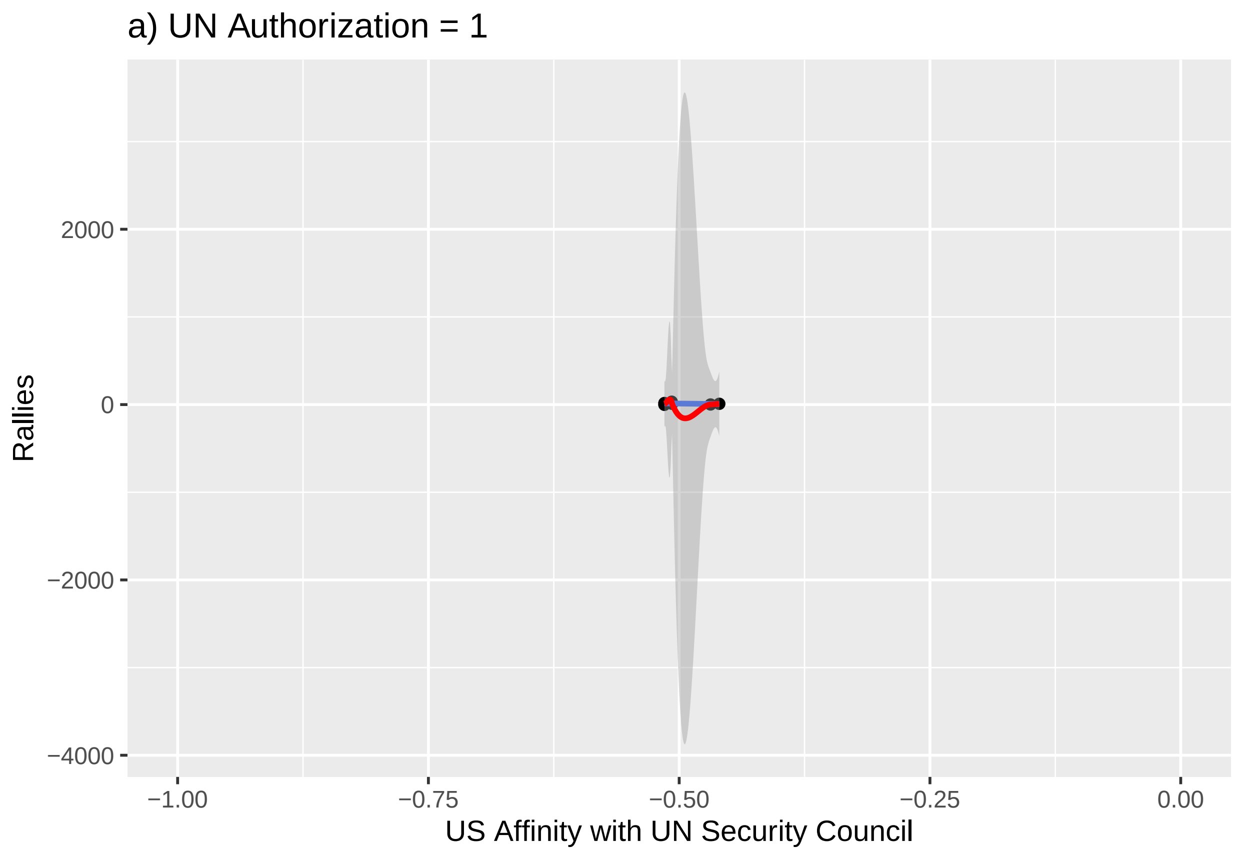


```
ggplot(plot4data, aes(x = S, y = rally)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(method = "loess", color = "red") +
  labs(x = "US Affinity with UN Security Council",
       y = "Rallies",
       title = "a) UN Authorization = 1") +
  xlim(-1, 0)
```

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

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.51498
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.0079372
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 5.2464e-005

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at
## -0.51498

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius
## 0.0079372

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition
## number 0

## Warning in predLoess(object$y, object$x, newx = if
## (is.null(newdata)) object$x else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : There are other near
## singularities as well. 5.2464e-005
```



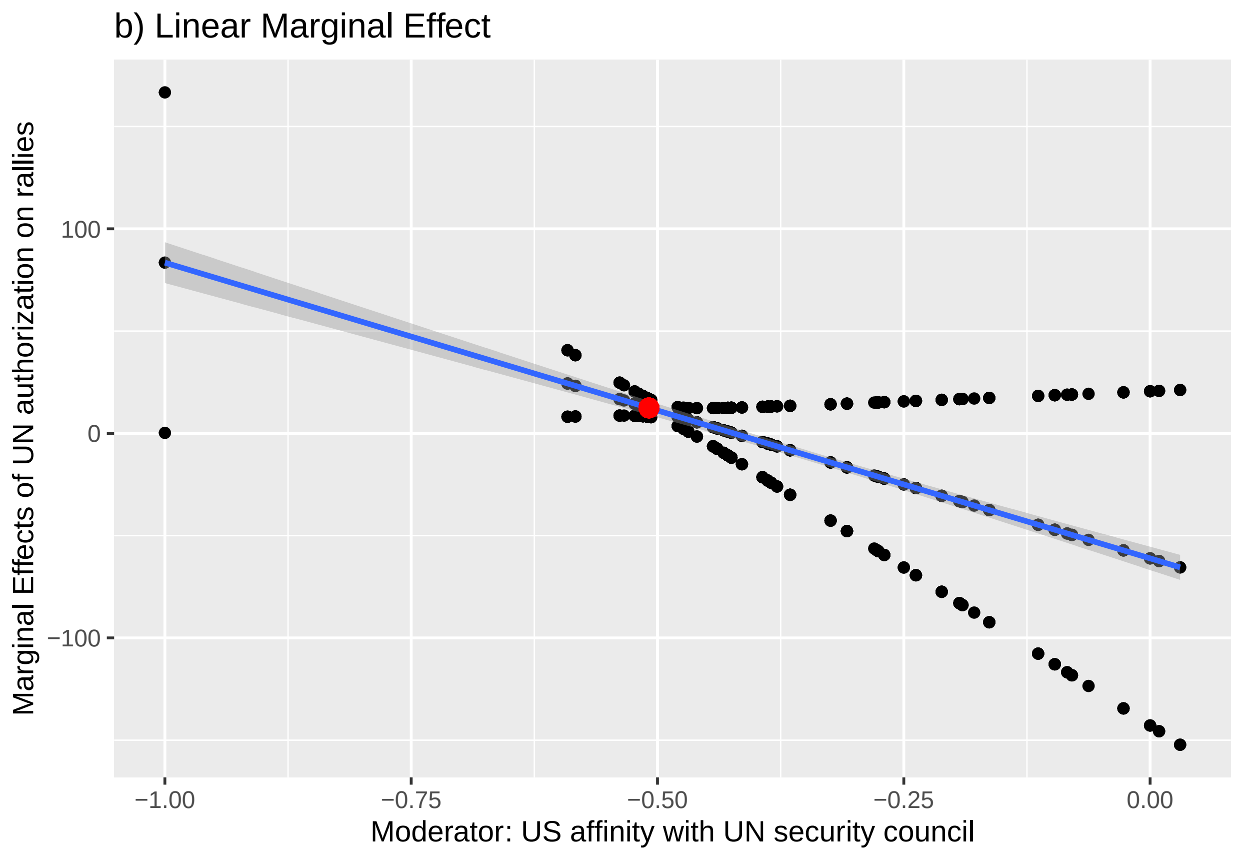
Now, I'm going to do graph b. I'm going to do basically exactly what I did in the last question.

```r
# First, I need to format it to fit the function.

func_chapman <- chapman %>%
  select(c(S, unauth, priorpop, rally))

func_chapman <- rename(func_chapman, X = S, D = unauth, Y = rally)

test_chapman <- cfunc(func_chapman)

test_chapman$conf.low <- as.numeric(test_chapman$conf.low)
test_chapman$conf.high <- as.numeric(test_chapman$conf.high)
```

```r
test_chapman2 <- pivot_longer(test_chapman, -X, names_to = "All", values_to = "value")

lowdata3 <- test_chapman2 %>%
  filter(All == "marg.effect") %>%
  arrange(X) %>%
  head(65)

lowmeanY3 <- sum(lowdata3$value)/nrow(lowdata3)
lowmeanX3 <- sum(lowdata3$X)/nrow(lowdata3)

middata3 <- test_chapman2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

middata3 <- middata3[66:130,]

midmeanY3 <- sum(middata3$value)/nrow(middata3)
midmeanX3 <- sum(middata3$X)/nrow(middata3)

highdata3 <- test_chapman2 %>%
  filter(All == "marg.effect") %>%
  arrange(X)

highdata3 <- highdata3[131:194,]

highmeanY3 <- sum(highdata3$value)/nrow(highdata3)
highmeanX3 <- sum(highdata3$X)/nrow(highdata3)
```

```r
ggplot(test_chapman2, aes(x = X, y = as.numeric(value))) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "Moderator: US affinity with UN security council",
       y = "Marginal Effects of UN authorization on rallies",
       title = "b) Linear Marginal Effect") +
  geom_point(aes(x = lowmeanX3, y = lowmeanY3), color = "red", size = 3)
```

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



2) Find and describe an example (in the news or in an academic article) of measurement bias. Explain whether and why measurement bias impacts the conclusions that the authors draw. Is there something we might be able to do about this measurement bias? Could you improve the study using different data/research design?

Measurement bias or measurement error is a common aspect of both news and academic articles. The first area that my mind goes to when I think about this concept is Russian polling data. There are three main public opinion polling organizations in Russia – VTsIOM, FOM, and Levada. Levada is an independent organization, but the other two are state-run. In today's political environment in Russia, VTsIOM and FOM polls are the main ones that get picked up by mainstream Russian news outlets. Here's an example:

https://lenta.ru/news/2022/10/07/vciom_again/

According to this article from Lenta.ru (a relatively reliable Russian news source), VTsIOM recently reported that 76.9% of Russians trust Russian president Vladimir Putin. But there are a number of issues at work that almost certainly lead to measurement bias/error in this study.

For one, VTsIOM is a state-run organization, so they have a special incentive to over-report support for state institutions and have routinely done so over the last 20 years or so. Most people who do work on Russia tend to cite Levada data over VTsIOM data, as it is generally considered to be more independent and less susceptible to government pressure. So the fact that VTsIOM sociologists are already employed by the state is a key factor leading to measurement bias. This is not exactly something that can be fixed; it's simply a fact.

Secondly, on a related note, after the start of the war in Ukraine in February 2022, the Russian government became significantly more restrictive and repressive towards dissent. This biases VTsIOM's reporting in a number of ways. For one, VTsIOM has even *less* of an incentive now to report accurate statistics of support for the Russian president; if they did, the likelihood that they would get in trouble is much higher than it was before. Also, respondents have less of an incentive to answer the question honestly. Given the fact that Russians routinely get arrested for any sort of anti-Putin rhetoric, there is a sufficiently high likelihood that even in a supposedly "anonymous" survey, a person voicing an anti-government opinion could be traced and persecuted by the state. So the likelihood that the statistics, as reported by Lenta, are strictly accurate, are not particularly high. There is almost certainly a fair amount of measurement bias and error in this data.

So what can be done? Unfortunately, there are a number of confounding factors for improvement. For one, there is a fair amount of *willing* bias and error on the part of the polling organization that is difficult to solve. The solution comes from a) detaching VTsIOM from the state, and b) loosening the state's autocratic grip. But these aren't exactly issues that are easily solvable in "the next poll." These are systematic changes to the society of a country that aren't easy to address.

There are likely some ways, however, in which I could change the research design. I think that if I were VTsIOM, I would hire contractors to run the surveys who are skilled in the methodology, so that it would look more independent and respondents wouldn't get as nervous. It would also give VTsIOM some more deniability if the statistics that they gather are not to the government's liking. The question is, however – does VTsIOM care? Do they actually *want* to be more objective? Or do they specifically want there to be some bias and error? The answer to this question is, unfortunately, a whole other issue.