Econ 675 Assignment 1

Nathan Mather

September 28, 2018

Contents

List of Figures

List of Tables

1 Question 1: Simple Linear Regression with Measurement Error

1.1 OLS estimator

$\hat{\beta}_{ls} = (\tilde{x}'\tilde{x})^{-1}\tilde{x}'y$ and we want to show that $\hat{\beta}_{ls} \to_p \lambda\beta$

First note that

$$y = \beta(\tilde{x} - \mu) + \epsilon = \beta\tilde{x} + (\epsilon - \beta\mu)$$

So The measurement error in x becomes part of the error term in the regression. This means OLS will lead to a negative bias in $\hat{\beta}_{ls}$ if the true β is positive and a positive bias in $\hat{\beta}_{ls}$ if the true β is negative (an attenuation bias). In order to determine the magnitude of the bias consider the following.

$$\hat{\beta}_{ls} = \frac{\mathrm{Cov}(\tilde{x}, y)}{\mathrm{Var}(\tilde{x})} = \frac{\mathrm{Cov}(x+\mu, \beta x + \epsilon)}{\mathrm{Var}(x+\mu)} = \frac{\beta\mathrm{Cov}(x,x) + \mathrm{Cov}(x,\epsilon) + \mathrm{Cov}(\mu, \beta x) + \mathrm{Cov}(\mu, \epsilon)}{\mathrm{Var}(x+\mu)}$$

$$= \frac{\beta\mathrm{Var}(x)}{\mathrm{Var}(x+\mu)} \to_p \frac{\beta\sigma_x^2}{\sigma_x^2 + \sigma_\mu^2} = \lambda\beta$$

$$\text{This implies that } \lambda = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_\mu^2}$$

1.2 Standard Errors

Start with $\hat{\epsilon} = y - \hat{\beta}_{ls}(x + \mu)$

Now add and subtract the True error term $\epsilon = y - \beta x$ and collect terms to get $\hat{\epsilon} + \epsilon - \epsilon = \epsilon - (y - \beta x) + y - \hat{\beta}_{ls}x - \hat{\beta}_{ls}\mu = \epsilon + (\beta - \hat{\beta}_{ls})x - \hat{\beta}_{ls}\mu$

recall that $\hat{\beta}_{ls} \to_p \lambda\beta$ and that ϵ, x, μ are all uncorrelated. This implies that $\hat{\sigma_\epsilon^2} \to_p \sigma_\epsilon^2 + (1-\lambda)^2\beta^2\sigma_x^2 + \lambda^2\beta^2\sigma_\mu^2$

so this is biased upwards since we are adding positive terms to the true value

next to compute the probability limit of $\hat{\sigma}_\epsilon^2(\tilde{x}'\tilde{x}/n)^{-1}$

$$\hat{\sigma}_\epsilon^2(\tilde{x}'\tilde{x}/n)^{-1} = \frac{\hat{\sigma}_\epsilon^2}{\hat{\sigma}_{\tilde{x}}^2} \to_p \frac{\sigma_\epsilon^2 + (1-\lambda)^2\beta^2\sigma_x^2 + \lambda^2\beta^2\sigma_\mu^2}{\sigma_x^2 + \sigma_\mu^2}$$

$$= \frac{\sigma_x^2}{\sigma_x^2 + \sigma_\mu^2}\left(\frac{\sigma_\epsilon^2}{\sigma_x^2}\right) + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_\mu^2}(1-\lambda)^2\beta^2 + \frac{\sigma_\mu^2}{\sigma_x^2 + \sigma_\mu^2}\lambda^2\beta^2 = \lambda\left(\frac{\sigma_\epsilon^2}{\sigma_x^2}\right) + \lambda(1-\lambda)^2\beta^2 + (1-\lambda)\lambda^2\beta^2$$

now note that $\lambda(1-\lambda)^2\beta^2 + (1-\lambda)\lambda^2\beta^2 = \beta^2\lambda(1-\lambda)[(1-\lambda) + \lambda] = \beta^2\lambda(1-\lambda)$
Combining these gives us that

$$\frac{\hat{\sigma}_\epsilon^2}{\hat{\sigma}_{\tilde{x}}^2} \to_p \frac{\lambda\sigma_\epsilon^2}{\sigma_x^2} + \lambda(1-\lambda)\beta^2$$

multiplying the first term by λ biases the result downwards but the second term is positive so it biases the result upwards. So the overall result of the bias cannot be signed in general

1.3 t-test

$$\frac{\hat{\beta}_{ls}}{\sqrt{\hat{\sigma}_\epsilon^2(\tilde{x}'\tilde{x}/n)^{-1}}} \to_p \frac{\lambda\beta}{\sqrt{\lambda\frac{\sigma_\epsilon^2}{\sigma_{\tilde{x}}^2} + \lambda(1-\lambda)\beta^2}} = \frac{\sqrt{\lambda}\beta}{\sqrt{\frac{\sigma_\epsilon^2}{\sigma_{\tilde{x}}^2} + (1-\lambda)\beta^2}}$$

which is smaller than

$$\frac{\beta}{\sqrt{\frac{\sigma_\epsilon^2}{\sigma_x^2}}}$$

So the t-test is downward biased

1.4 Second measurement, Consistency

$$y = x\beta + \epsilon$$

by assumption $\mathrm{E}[\breve{x}\epsilon] = 0$

Now multiply y by $\breve{x}'$ and take the expectation to get $\mathrm{E}[\breve{x}'y] = \mathrm{E}[\breve{x}'x]\beta$

Now assuming $\mathrm{E}[\breve{x}'x]$ is full rank we get $\beta = (\mathrm{E}[\breve{x}'x])^{-1}\mathrm{E}[\breve{x}'y]$

So $\hat{\beta}_{IV} = (\breve{x}'x)^{-1}\breve{x}'y$

Now to show it is consistent

$\hat{\beta}_{IV} = (\breve{x}'x)^{-1}\breve{x}'(x\beta + \epsilon) = \beta + (\frac{\breve{x}'x}{n})^{-1}(\frac{\breve{x}'\epsilon}{n}) \to_p \beta$

since $\mathrm{E}[\breve{x}'\epsilon] = 0$ so $\frac{\breve{x}'\epsilon}{n} \to_p 0$ by LLN

1.5 Second measurement, Distribution

$$\sqrt{n}(\hat{\beta}_{IV} - \beta) = (\breve{x}'x)^{-1}\breve{x}'\epsilon = \sqrt{n}\left(\frac{\breve{x}'x}{n}\right)^{-1}\left(\frac{\breve{x}'\epsilon}{n}\right)$$

Now using the CLT we get

$$\sqrt{n}\left(\frac{\breve{x}'\epsilon}{n}\right) \xrightarrow{d} N(0, \mathrm{E}[\breve{x}'\epsilon'\epsilon\breve{x}])$$

Now all together we get

$$\sqrt{n}(\hat{\beta}_{IV} - \beta) \xrightarrow{d} N(0, \mathrm{E}[\breve{x}'x]^{-1}\mathrm{E}[\breve{x}'\epsilon'\epsilon\breve{x}]E[x\breve{x}']^{-1})$$

1.6 Second measurement, Inference

To create a confidence interval robust to Standard errors we want to use the following, unsimplified, version of the asymptotic variance estimator.

$$\hat{V}_{IV} = Avar(\hat{\beta}_{IV}) = (\breve{x}'x)^{-1}\left(\sum_{i=1}^{n}\epsilon_i^2\breve{x}_i'\breve{x}_i\right)(\breve{x}'x)^{-1}$$

We also showed above that

$$\sqrt{n}\left(\frac{\hat{\beta}_{IV}}{\sqrt{\hat{V}_{IV}}}\right) \to_d \mathcal{N}(\beta, 1)$$

Inverting the standard normal distribution and the following confidence interval

$$\left[\hat{\beta}_{IV} - \Phi^{-1}\left(1 - \frac{(1-\alpha)}{2}\right)\left(\sqrt{\frac{\hat{V}_{IV}}{n}}\right), \hat{\beta}_{IV} + \Phi^{-1}\left(1 - \frac{(1-\alpha)}{2}\right)\left(\sqrt{\frac{\hat{V}_{IV}}{n}}\right)\right]$$

where $\alpha = 0.95$ in this case

1.7 Validation sample, Consistency

First note that $(\frac{1}{n}\tilde{x}'\tilde{x}) \rightarrow_p \sigma_x^2 + \sigma_u^2$ and as shown in part 1 $\hat{\beta}_{ls} \rightarrow_p \beta\frac{\sigma_x^2}{\sigma_x^2+\sigma_\mu^2}$

Now we define $\hat{\beta}_{VS} = \hat{\beta}_{ls}\left(\frac{1}{n}\frac{\tilde{x}'\tilde{x}}{\breve{\sigma}_x^2}\right)$

and by Slutsky's theorem we get that $\hat{\beta}_{VS} \rightarrow_p \beta$

1.8 Validation sample, Distribution

We know from section 1.7 that $\hat{\beta}_{VS} = \hat{\beta}_{ls}\left(\frac{1}{n}\frac{\tilde{x}'\tilde{x}}{\breve{\sigma}_x^2}\right)$

We can break this into three pieces and define $\hat{\beta}_{VS}$ in the following way

$$\hat{\beta}_{VS} = g(a,b,c) = \frac{ab}{c}$$
$$a = \hat{\beta}_{ls}$$
$$b = \frac{1}{n}\tilde{x}'\tilde{x}$$
$$c = \breve{\sigma}_x^2$$

g is a continuous function so we can apply the delta method.

$$\sqrt{n}\left(g\left(\hat{\beta}_{ls}, \frac{1}{n}\tilde{x}'\tilde{x}, \breve{\sigma}_x^2\right) - g\left(\lambda\beta, \sigma_x^2 + \sigma_\mu^2, \sigma_x^2\right)\right) \rightarrow_d \mathcal{N}\left(\nabla g\left(\lambda\beta, \sigma_x^2 + \sigma_\mu^2, \sigma_x^2\right)' \mathbf{\Sigma} \nabla g\left(\lambda\beta, \sigma_x^2 + \sigma_\mu^2, \sigma_x^2\right)\right)$$

$$V_{vs} = \nabla g\left(\lambda\beta, \sigma_x^2 + \sigma_\mu^2, \sigma_x^2\right)' \mathbf{\Sigma} \nabla g\left(\lambda\beta, \sigma_x^2 + \sigma_\mu^2, \sigma_x^2\right)$$

1.9 Validation sample, Inference

Similar to problem 1.6 we have that

$$\sqrt{n}\left(\frac{\hat{\beta}_{VS}}{\sqrt{\hat{V}_{VS}}}\right) \rightarrow_d \mathcal{N}(\beta, 1)$$

Inverting the standard normal distribution and the following confidence interval

$$\left[\hat{\beta}_{VS} - \Phi^{-1}\left(1 - \frac{(1-\alpha)}{2}\right)\left(\sqrt{\frac{\hat{V}_{VS}}{n}}\right), \hat{\beta}_{VS} + \Phi^{-1}\left(1 - \frac{(1-\alpha)}{2}\right)\left(\sqrt{\frac{\hat{V}_{VS}}{n}}\right)\right]$$

where $\alpha = 0.95$ in this case

1.10 FE estimator, Consistency

First note that because we have $T = 2$, the FE estimator is equivalent to the first-difference (FD) estimator. That is

$$\hat{\beta}_{FE} = \hat{\beta}_{FD}$$
$$\left(\frac{1}{n}\sum_{i-1}^{n}(\tilde{x}_{i2} - \tilde{x}_{i1})^2\right)^{-1}\left(\frac{1}{n}\sum_{i-1}^{n}(\tilde{x}_{i2} - \tilde{x}_{i1})(y_{i2} - y_{i1})\right)$$

Not by using the WLLN:

$$\frac{1}{n}\sum_{i-1}^{n}(\tilde{x}_{i2} - \tilde{x}_{i1})^2 \to_p \mathrm{E}[(\tilde{x}_{i2} - \tilde{x}_{i1})^2] = \mathrm{E}[(x_{i2} - x_{i1} + u_{i2} - u_{i1})^2]$$
$$= \mathrm{E}[(x_{i2} - x_{i1})^2] + \mathrm{E}[(u_{i2} - u_{i1})^2] + 2\mathrm{E}[(x_{i2} - x_{i1})(u_{i2} - u_{i1})] = \sigma^2_{\Delta x} + \sigma^2_{\Delta u}$$

since $\mathrm{E}[x_{it}u_{it}] = 0 \ \forall \ t, s \in \{1, 2\}$ Next

$$\frac{1}{n}\sum_{i-1}^{n}(\tilde{x}_{i2} - \tilde{x}_{i1})(y_{i2} - y_{i1}) \to_p \mathrm{E}[(\tilde{x}_{i2} - \tilde{x}_{i1})(y_{i2} - y_{i1})]$$
$$= \mathrm{E}[(x_{i2} - x_{i1} + u_{i2} - u_{i1})(x_{i2}\beta - x_{i1}\beta + e_{i2} - e_{i1})]$$

$$= \mathrm{E}[(x_{i2} - x_{i1})^2]\beta + \mathrm{E}[(x_{i2} - x_{i1})(e_{i2} - e_{i1})] + \mathrm{E}[(x_{i2} - x_{i1})(u_{i2} - u_{i1})]\beta + \mathrm{E}[(u_{i2} - u_{i1}(e_{i2} - e_{i1}))]$$
$$= \mathrm{E}[(x_{i2} - x_{i1})^2]\beta = \sigma_{\Delta x}\beta$$

since $\mathrm{E}[x_{it}u_{it}] = \mathrm{E}[x_{it}e_{it}] = \mathrm{E}[u_{it}e_{it}] \ \forall t, s \in \{1, 2\}$. Finally we can put these together by the CMT to get.

$$\hat{\beta}_{FE} \to_p \frac{\sigma^2_{\Delta x}}{\sigma^2_{\Delta x} + \sigma^2_{\Delta u}}\beta$$

Thus we can see that $\hat{\beta}_{FE}$ is biased downwards.

1.11 FE estimator, time dependence

Covariance stationarity implies that $\sigma^2_{xt} = \sigma^2_x$ and $\sigma^2_{ut} = \sigma^2_u$ for $t \in \{1, 2\}$
this means that

$$\sigma^2_{\Delta x} = \mathrm{V}[x_{i2} - x_{i1}] = \mathrm{V}[x_{i2}] + \mathrm{V}[x_i 1] - 2Cov[x_{i2}, x_{i1}] = 2\sigma^2_x - 2Cov[x_{i2}, x_{i1}]$$

Thus we get

$$\gamma = \frac{2(\sigma^2_x - \mathrm{Cov}[x_{i2}, x_{i1}])}{2(\sigma^2_x - \mathrm{Cov}[x_{i2}, x_{i1}] + \sigma^2_u - \mathrm{Cov}[u_{i2}, u_{i1}])}$$
$$= \frac{\sigma^2_x - \rho_x\sigma^2_x}{\sigma^2_x - \rho_x\sigma^2_x + \sigma^2_u - \rho_u\sigma^2_u} = \frac{\sigma^2_x(1 - \rho)x}{\sigma^2_x(1 - \rho_x) + \sigma^2_u(1 - \rho_u)} = \frac{\sigma^2_x}{\sigma^2_x + \sigma^2_u\frac{1-\rho_u}{1-\rho_x}}$$

1.12 FE estimator, time dependence

let $\rho_u = 0$ given this we can calculate

$$lim_{n \to \infty} \gamma = 0$$

This implies that under the given conditions $\hat{\beta}_{FE}$ will be biased to zero. Thus the FE estimator will tend to give you zero for coefficients regardless of the true β. The idea is that if x is almost perfectly correlated over time, but the measurement error is completely random, then the only variation in our observations over time is because of random measurement error. So, our ability to observe actual variation in the variable of interest x is going to zero and the level of noise in our observations is high.

2 Question 2: Implementing Least-Squares Estimators

2.1 part 1

Start by adding and subtracting $x\tilde{\beta}$ to get

$$(y - x\tilde{\beta} + x\tilde{\beta} - x\beta)'W(y - x\tilde{\beta} + x\tilde{\beta} - x\beta)$$

$$= (y - x\tilde{\beta})'W(y - x\tilde{\beta}) + (y - x\tilde{\beta})'W(x\tilde{\beta} - x\beta) + (x\tilde{\beta} - x\beta)'W(y - x\tilde{\beta}) + (x\tilde{\beta} - x\beta)'W(x\tilde{\beta} - x\beta)$$

$$= (y - x\tilde{\beta})'W(y - x\tilde{\beta}) + 2(x\tilde{\beta} - x\beta)'W(y - x\tilde{\beta}) + (x\tilde{\beta} - x\beta)'W(x\tilde{\beta} - x\beta)$$

Now we need to find $\tilde{\beta}$ to minimize this equation. We want to set the middle term to zero so we need a $\tilde{\beta}$ such that $\tilde{\beta}'x'W(y - x\tilde{\beta}) = \beta'x'W(y - x\tilde{\beta})$

we pick $\tilde{\beta}$ such that $x'W(y - x\tilde{\beta}) = 0$ giving us

$$\tilde{\beta} = (x'W'x)^{-1}(x'Wy)$$

Now when we minimize over β the first term is irrelevent as it does not include a β. The middle term is 0 so it does not matter. The last term is positive semi definite and so it is minimized by setting $\beta = \tilde{\beta}$

2.2 Part 2

$$\sqrt{n}(\hat{\beta}(w) - \beta) = \sqrt{n}((x'Wx)^{-1}x'W(x\beta + \epsilon) - \beta) = \sqrt{n}((x'Wx)^{-1}x'W\epsilon)$$

$$= ((\frac{1}{n}x'Wx)^{-1}\sqrt{n}(\frac{1}{n}x'W\epsilon))$$

under appropriate assumptions we have by LLN that $(\frac{1}{n}x'Wx) \to_p A$
We also have that $\sqrt{n}(\frac{1}{n}x'W\epsilon) \to_d \mathcal{N}(0, B)$ by CLT
In this case we get $B = \frac{1}{n}\mathbb{V}[x'W\epsilon] = \frac{1}{n}\mathbb{E}[x'W\epsilon'\epsilon Wx]$
And we have that $V(W) = A^{-1}BA^{-1}$

2.3 Part 3

To estimate $V(W) = A^{-1}BA^{-1}$ we are mostly just putting hats on things

$$\hat{A} = \frac{1}{n}(x'\hat{W}x)$$
$$\hat{B} = \frac{1}{n}(x'\hat{W}\hat{\epsilon}'\hat{\epsilon}\hat{W}x)$$

so that gives us

$$\hat{V}(W) = \frac{1}{n}(x'\hat{W}x)^{-1}(x'\hat{W}\hat{\epsilon}'\hat{\epsilon}\hat{W}x)(x'\hat{W}x)^{-1}$$

2.4 Part 4

See code in the appendix. The results do not change between the regular and Cholesky inverse.

2.5 Part 5

2.5.1 a

variable	beta	se	t_test	p_value	CI_L	CI_U
const	6485.55	4513.51	1.44	0.15	-2384.89	15356.00
treat	1535.48	638.24	2.41	0.02	281.15	2789.82
black	-2592.38	795.00	-3.26	0.00	-4154.80	-1029.96
age	39.34	40.47	0.97	0.33	-40.20	118.88
educ	-740.54	944.68	-0.78	0.43	-2597.13	1116.05
educ_sq	60.08	53.77	1.12	0.26	-45.59	165.75
earn74	-0.03	0.10	-0.29	0.77	-0.23	0.17
black_earn74	0.18	0.13	1.33	0.18	-0.08	0.43
u74	1316.03	1505.93	0.87	0.38	-1643.58	4275.64
u75	-1167.69	1275.42	-0.92	0.36	-3674.27	1338.90

2.5.2 b

They coincide because I made sure to weight the variance matrix properly and used HC1 in the sandwich package in R

term	estimate	std.error	statistic	p.value
(Intercept)	6485.55	4513.51	1.44	0.15
treat	1535.48	638.24	2.41	0.02
black	-2592.38	795.00	-3.26	0.00
age	39.34	40.47	0.97	0.33
educ	-740.54	944.68	-0.78	0.43
educ_sq	60.08	53.77	1.12	0.26
earn74	-0.03	0.10	-0.29	0.77
black_earn74	0.18	0.13	1.33	0.18
u74	1316.03	1505.93	0.87	0.38
u75	-1167.69	1275.42	-0.92	0.36

3 Question 3: Analysis of Experiments

3.1 Neyman's approach

3.1.1 a

$$\mathrm{E}[T_{DM}] = \mathrm{E}[\bar{Y}_1] - \mathrm{E}[\bar{Y}_0] = \mathrm{E}\left[\frac{1}{N_1}\sum_{i=1}^{n} D_i(1)Y_i\right] - \mathrm{E}\left[\frac{1}{n-N_1}\sum_{i=1}^{n} D_i(0)Y_i\right]$$
$$= \frac{1}{N_1}\sum_{i=1}^{n}(D_i(1)\mathrm{E}[Y_i]) - \frac{1}{n-N_1}\sum_{i=1}^{n}(D_i(0)\mathrm{E}[Y_i])$$
$$= \frac{1}{N_1}\sum_{i=1}^{n}(D_i(1))\,\mathrm{E}[Y_i(T_i)|T_i = 1] - \frac{1}{n-N_1}\sum_{i=1}^{n}(D_i(0))\,\mathrm{E}[Y_i(T_i)|T_i = 0]$$

Now note that since T_i is random:

$$\mathrm{E}[Y_i(T_i)|T_i = 1] = \mathrm{E}[Y_i(1)]$$
$$\mathrm{E}[Y_i(T_i)|T_i = 0] = \mathrm{E}[Y_i(0)]$$

Together this gives us:

$$\mathrm{E}[T_{DM}] = \mathrm{E}[Y_i(1)] - \mathrm{E}[Y_i(0)]$$

or

$$\tau_{ATE} = \frac{1}{n}\sum_{i=1}^{n} Y_i(1) - \frac{1}{n}\sum_{i=1}^{n} Y_i(0)$$

The estimate from the data is 1794.34 and can be seen in the table in part 2

3.1.2 b

TDM est	Conservative SE	CI Lower	CI Upper
1794.34	671.00	479.21	3109.47

3.2 Fisher's approach

3.2.1 a

DM P value
0.01

KS P value
0.04

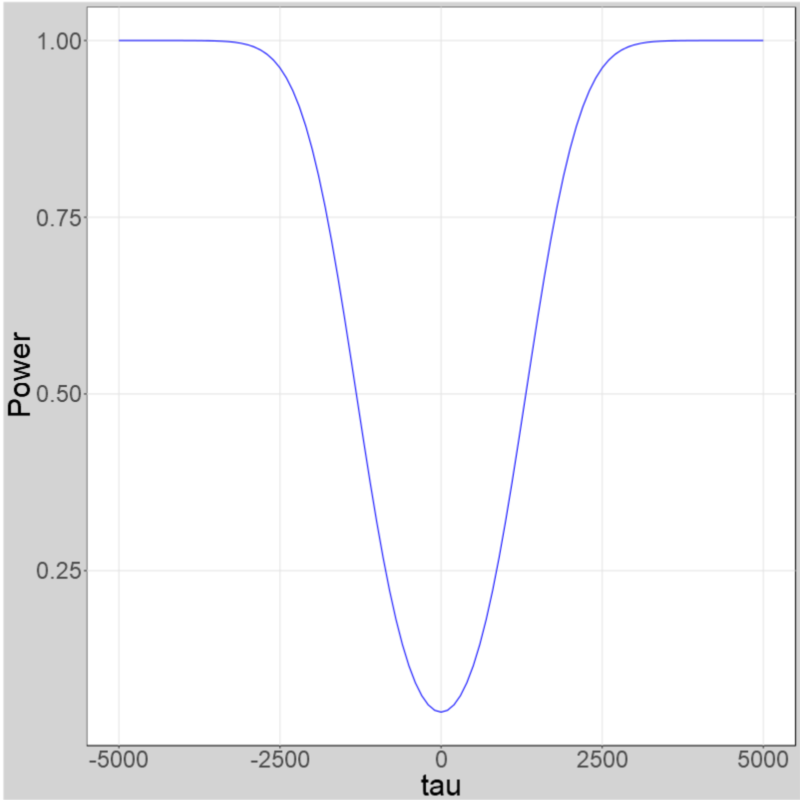
3.2.2 b

To find the confidence interval I calculate fisher's exact P value for a range of Null hypotheses. The table for this calculation is below. We can then look the table and look for p values closest to our .05 cutoff. This gives us a confidence interval of 500 to 3000. Using a more detailed set of points I can find that the confidence interval is more precisely 540 to 3055.

Hypothesized Treatment Effect	p_value
5000.00	0.00
4750.00	0.00
4500.00	0.00
4250.00	0.00
4000.00	0.00
3750.00	0.00
3500.00	0.01
3250.00	0.02
3000.00	0.05
2750.00	0.13
2500.00	0.24
2250.00	0.46
2000.00	0.74
1750.00	0.94
1500.00	0.66
1250.00	0.38
1000.00	0.22
750.00	0.08
500.00	0.04
250.00	0.02
0.00	0.01
-250.00	0.00
-500.00	0.00
-750.00	0.00
-1000.00	0.00
-1250.00	0.00
-1500.00	0.00

3.3 Power calculations

3.3.1 a



3.3.2 b

The math can be seen in my code. The number required is 1437

4 Appendix

4.1 R Code

pset 2 Labor

```
#=======================================#
# ==== Load packages and clear data ====
#=======================================#

library(data.table)
library(Matrix)
library(lmtest)
library(sandwich)
library(broom)
library(ggplot2)
library(stats)
# clear objects and script
rm(list = ls(pos = ".GlobalEnv"), pos = ".GlobalEnv")
options(scipen = 999)
cat("\f")

#=============================#
# ==== Question 2  part 4 ====
#=============================#

  #================================#
  # ==== geneerate random data ====
  #================================#

    # set n_col and n_row
    n_col <- 10
    n_row <- 100
    n_cell <- n_col*n_row

    # create random matrices
    y_data <- matrix(runif(n_row, 0, 100), nrow = n_row, ncol = 1)
    x_data <- matrix(runif(n_cell, 0, 1), nrow = n_row, ncol = n_col)

  #================================#
  # ==== write function for q2 ====
  #================================#

    # commented out, but usefull for line by line debug
    # x = x_data
    # y = y_data

    # function
    mat_reg <- function(x = NULL, y = NULL, opt_chol = FALSE, CI_level = .95){

      # get matrix size parameters
      n_col <- ncol(x)
      n_row <- nrow(x)

    #=======================#
    # ==== estimate beta ====
```

```
#========================#

  # check which inverse function to use
  if(!opt_chol){

    # use standard inverse
    B <- Matrix::solve(Matrix::crossprod(x, x))%*%(Matrix::crossprod(x, y))
  }else{

    # use cholesy inverse
    chol_m <- chol(Matrix::crossprod(x, x))
    B<- chol2inv(chol_m)%*%(Matrix::crossprod(x, y))

  }

#=====================#
# ==== estimate V ====
#=====================#

  # calculate residuals
  my_resid <- y - x%*%B

  # calculate middle part of variance matrix. the mear
  M_diag <- diag(as.numeric(my_resid^2*(n_row/(n_row-n_col))), nrow = n_row, ncol = n_row)
  M <- (t(x) %*% M_diag %*% x)


  # see if I need to use cholesky
  if(!opt_chol){

    # calculate asymptotic variance
    V <- solve(crossprod(x, x)) %*% M  %*% solve(crossprod(x, x))

  }else{

    A_inv <-   chol2inv(chol_m) %*% M  %*% chol2inv(chol_m)
    V <- A_inv

    }
  sqrt(diag(V))

#======================#
# ==== other stats ====
#======================#

  # start by putting beta and  diagonal of variance in a data.table
  out_dt <- data.table(beta = as.numeric(B), V_hat = diag(V) )

  # calculate standard errors
  out_dt[, se := sqrt(V_hat)]

  # calculate t test
  out_dt[, t_test := beta/(se)]
```

```
      # calculate p values
      out_dt[, p_value := 2*(1- pt((abs(t_test)), n_row - n_col))]

      # calculate confidence interval
      out_dt[, CI_L := beta - (se) * qt(1-((1-CI_level)/2), n_row )]
      out_dt[, CI_U := beta + (se) * qt(1-((1-CI_level)/2), n_row )]

      # drop v_hat cause I dont need it
      out_dt[, V_hat := NULL]

      # create list to return
      out_list <- list()

      out_list[["results"]] <- out_dt
      out_list[["varcov"]] <- V

      return(out_list)

}

#=======================================#
# ==== run function on random data ====
#=======================================#

  # run on random data with and without cholesky
  reg_1 <- mat_reg(x = x_data, y = y_data, opt_chol = FALSE)
  reg_2 <- mat_reg(x = x_data, y = y_data, opt_chol = TRUE)

  # compare coefficients, differences are just floating point errors
  coeff_diff <- reg_1[["results"]][, beta] - reg_2[["results"]][, beta]

  # compare varcov NOTE: differences are just floating point errors
  all.equal(reg_1$varcov, reg_2$varcov)
  reg_1$varcov -  reg_2$varcov

#============================#
# ==== Question 2 part 5 ====
#============================#

  #===========================#
  # ==== matrix function ====
  #===========================#

    # load daata #note paste is so it fits on pdf in markdown
    lalonde_dt <- fread(pasate0("C:/Users/Nmath_000/Documents/MI_school/Second",
                                "Year/675 Applied Econometrics/hw/hw1/LaLonde_1986.csv"))

    # grab y matrix
    y_la <- as.matrix(lalonde_dt[, earn78])

    # create other vars for regression
    lalonde_dt[, educ_sq := educ^2]
    lalonde_dt[, black_earn74 := black*earn74]
```

```r
    lalonde_dt[, const := 1]

     # grab x vars
    x_vars <- c("treat", "black", "age", "educ",
                "educ_sq", "earn74","black_earn74",
                "u74","u75")

    # make x matrix
    x_la <- as.matrix(lalonde_dt[, c("const", x_vars), with = FALSE])

    # run function on this data
    lalonde_reg <- mat_reg(x = x_la, y = y_la)

    # grab the results
    results_2_5_a <- lalonde_reg[["results"]]

    # add in coef label
    results_2_5_a[, variable := c("const", x_vars)]

    # put variables in front
    setcolorder(results_2_5_a, c("variable", setdiff( colnames(results_2_5_a), "variable")))

  #====================#
  # ==== using lm ====
  #====================#

    # get regression formula
    reg_form <- as.formula(paste("earn78~", paste(x_vars, collapse="+")))

    # run regression
    lalonde_lm <- lm(reg_form, lalonde_dt)

    # get summary, NOTE: these are NOT robust standard errors
    lalong_lm_dt <- summary(lalonde_lm)$coefficients

    # get robust standard errors. I use HC2 to match my math above
    # any differnces are floating point errors
    lm_robust <- coeftest(lalonde_lm, vcov = vcovHC(lalonde_lm, type="HC1"))

    results_2_5_b <- data.table(tidy(lm_robust))

#======================#
# ==== Question 3 ====
#======================#


  #=================#
  # ==== neyman ====
  #=================#

    # 3.1.a calculate ATE
    TDM <- lalonde_dt[treat == 1, mean(earn78)] - lalonde_dt[treat == 0, mean(earn78)]
```

```r
  # get variance for treatment and no treatnment
  s1_sq <- lalonde_dt[treat == 1, var(earn78)]
  s0_sq <- lalonde_dt[treat == 0, var(earn78)]

  # get V_tdm
  V_tdm <- s1_sq/lalonde_dt[treat == 1, .N] + s0_sq/lalonde_dt[treat == 0, .N]

  # get standard error
  se_tdm <- sqrt(V_tdm)

  # constuct 95% convidence interval
   tdm_CI_L <- TDM - se_tdm * qnorm(.975)
   tdm_CI_U <- TDM + se_tdm * qnorm(.975)

   # put together resuts
   results_3_1_b <- data.table("TDM est" = TDM,
                               "Conservative SE" = se_tdm,
                               "CI Lower" = tdm_CI_L,
                               "CI Upper" =  tdm_CI_U)

#==================#
# ==== fisher ====
#==================#

   # definitions for line by line debug
   # in_data= lalonde_dt
   # y_var = "earn78"
   # treat_var = "treat"
   # opt_test_stat= "DM"
   # n_iter = 10
   # null_hyp = 5000

  # write function for fisher p value
  fisher_p <- function(in_data       = NULL,
                       y_var         = NULL,
                       treat_var     = NULL,
                       null_hyp      = 0,
                       opt_test_stat = "DM",
                       n_iter        = 1999){

    # check that a test has ben species
    if(!opt_test_stat %chin% c("DM", "KS")){
      stop("Specify either DM ot KS test")
    }

    # check for non-zero null under the KS test (function doesn't do that)
    if(opt_test_stat == "KS" & null_hyp != 0){
      stop("The KS test is not compatibe with a non-zero null at the moment")
    }

    # copy data so I can create y(0) and y(1) cols without altering input data set
    data_c <- copy(in_data)
```

```
# create colums for sharp null treated and untreated y variables
data_c[get(treat_var) == 1, y_1 := get(y_var) ]
data_c[get(treat_var) == 0, y_1 := get(y_var) + null_hyp ]
data_c[get(treat_var) == 0, y_0 := get(y_var) ]
data_c[get(treat_var) == 1, y_0 := get(y_var) - null_hyp ]

# create a data.table for the results of bootstrap
sim_data <- data.table(iteration = c(1:(n_iter+1)))

# get the number of treated vars
n_treat <- nrow(data_c[get(treat_var) == 1, ])
n_row <- nrow(data_c)

# do actual test
if(opt_test_stat == "DM"){

  # get mean of treatment
  m_t <- data_c[get(treat_var) == 1, mean(get(y_var))]

  # get mean of untreated
  m_unt <- data_c[get(treat_var) == 0, mean(get(y_var))]

  test_1 <-m_t - m_unt - null_hyp

}
if(opt_test_stat == "KS"){
  ksout <- suppressWarnings(ks.test(data_c[get(treat_var) == 1, get(y_var)],
                                    data_c[get(treat_var) == 0, get(y_var)] ))
  test_1 <- ksout$statistic
}

# put results of actual data in table
sim_data[iteration == 1, test := test_1]

# for each iteration
for(i in 2:(n_iter + 1)){

  # create a permutation
  sample_i_1 <- sample.int(n = n_row, size = n_treat)
  sample_i_0 <- setdiff(c(1: n_row), sample_i_1)

  # calculate the averate treatment effect for this given sample
  if(opt_test_stat == "DM"){

    test_i <- data_c[sample_i_1, mean(y_1)] - data_c[sample_i_0, mean(y_0)] - null_hyp
  }
  if(opt_test_stat == "KS"){
    ksout <- suppressWarnings(ks.test(data_c[sample_i_1, y_1], data_c[sample_i_0, y_0] ))
    test_i <- ksout$statistic
  }


  # store this value in the data table
```

```
      sim_data[ i, test := test_i]
    }

    # get absolute value and rank of the tests
    sim_data[, abs_test := abs(test)]
    sim_data[, test_rank := frank(abs_test)]

    # get p value
    p_value <- (nrow(sim_data) - sim_data[iteration == 1, test_rank] + 1)/nrow(sim_data)

    return(p_value)


  }

  # run function on data
  results_3_2_a_DM <- fisher_p(in_data       = lalonde_dt,
                               y_var         = "earn78",
                               treat_var     = "treat",
                               null_hyp      = 0,
                               opt_test_stat = "DM",
                               n_iter        = 999)

  results_3_2_a_KS <- fisher_p(in_data       = lalonde_dt,
                               y_var         = "earn78",
                               treat_var     = "treat",
                               null_hyp      = 0,
                               opt_test_stat = "KS",
                               n_iter        = 999)

  # make it fancy for output
  results_3_2_a_DM <- data.table("DM P value" =   results_3_2_a_DM )
  results_3_2_a_KS <- data.table("KS P value" =   results_3_2_a_KS )
  #==============================================#
  # ==== construct 95% confidence interval ====
  #==============================================#

    # run fcuntions on a range of data
    grid <- seq(5000,-1500,-5)

    dm_p_list <- lapply(grid,
                        fisher_p,
                        in_data= lalonde_dt,
                        y_var = "earn78",
                        treat_var = "treat",
                        opt_test_stat= "DM",
                        n_iter = 999)

    results_3_2_b <- data.table(hyp_treat = grid, p_value = dm_p_list)

    # make it pretty
    setnames(results_3_2_b, "hyp_treat", "Hypothesized Treatment Effect")
```

```r
#==============================#
# ==== Power calculations ====
#==============================#

# plot attributes from EA
plot_attributes <- theme(plot.background = element_rect(fill = "lightgrey"),
                        panel.grid.major.x = element_line(color = "gray90"),
                        panel.grid.minor  = element_blank(),
                        panel.background = element_rect(fill = "white",
                                                        colour = "black") ,
                        panel.grid.major.y = element_line(color = "gray90"),
                        text = element_text(size= 30),
                        plot.title = element_text(vjust=0,
                                                  colour = "#0B6357",
                                                  face = "bold",
                                                  size = 30))


  # write power function
  power_function <- function(x, se= NULL) {
    1 - pnorm(qnorm(0.975)-x/se) + pnorm(-qnorm(0.975)-x/se)
  }

  # plot function
  power_plot <- ggplot(data = data.frame(x = 0), mapping = aes(x = x))
 power_plot <- power_plot + stat_function(fun = power_function,
                                          args = list(se=results_3_1_b$`Conservative SE`),
                                          color = "blue")
 power_plot <- power_plot + xlim(-5000,5000) + xlab("tau") + ylab("Power")  + plot_attributes
 power_plot

#=========================#
# ==== find needed n ====
#=========================#

 # Parameterize the equation
 p     = 2/3
 tau   = 1000

 # Write down the power function, which implicitly defines N
 Fun <- function(N, s.0 = s0_sq, s.1 = s1_sq){
   -0.8 + 1 - pnorm(qnorm(0.975)-tau/sqrt(1/N*s.1*(1/p)+1/N*s.0*(1/(1-p)))) +
     pnorm(-qnorm(0.975)-tau/sqrt(1/N*s.1*(1/p)+1/N*s.0*(1/(1-p))))
 }

 # Solve for N
 N.sol <- uniroot(Fun,c(0,100000000))$root


#=====================#
# ==== save stuff ====
#=====================#
```

```r
# save plot
png( paste0("C:/Users/Nmath_000/Documents/Code/courses/econ 675/PS_1_tex/",
            "power_func_r.png", height = 800, width = 800, type = "cairo"))
print(power_plot)
dev.off()

# save results #badcode so lazy
res_objects <- ls()[grepl("results", ls())]

save_tex_tables <- function(obj_name = NULL){

  table <- get(obj_name)

  print(xtable(table, type = "latex"),
        file = paste0("C:/Users/Nmath_000/Documents/Code/courses/econ 675/PS_1_tex/",
                      obj_name, ".tex"),
        include.rownames = FALSE,
        floating = FALSE)

}

lapply(res_objects, save_tex_tables)
```

4.2 Stata Code

```
clear all
set more off, perm

* set working directory
global dir "C:\Users\Nmath_000\Documents\MI_school\Second Year\675 Applied Econometrics\hw\hw1"

*import data
import delimited using "$dir\LaLonde_1986.csv"

**************
* question 2 *
**************


* create needed variables
gen educ_sq = educ^2
gen black_earn74 = black*earn74
gen const = 1

* store needed variables in locals
*local y earn76
*local x const treat black age educ educ_sq earn74 black_earn74 u74 u75

* use mata
mata:


y = st_data(., "earn78")
x = st_data(., ("const", "treat", "black", "age", "educ", "educ_sq", "earn74","black_earn74", "u74","u75"))

n_row = rows(x)
n_col = cols(x)

b = invsym(cross(x,x))*cross(x,y)

bc = cholinv(cross(x,x))*cross(x,y)

diff = b-bc

diff

my_resid = y - x*b
d = diag(my_resid:*my_resid:*(n_row/(n_row-n_col)))

v = invsym(cross(x, x))*(x' * d * x) * invsym(cross(x, x))

se = sqrt(diagonal(v))

tstat = b :/ se

p_value = 2*ttail(n_row-n_col, abs(tstat))

CI_L = b - (se) * invt(n_row-n_col, .975 )
CI_U = b + (se) * invt(n_row-n_col, .975 )

all_data = b, se, tstat, p_value, CI_L, CI_U
all_data
end

// now run regression
reg earn78 treat black age educ educ_sq earn74 black_earn74 u74 u75, robust

// nice, they match

**************
* question 3 *
**************

```

```
**********
* neyman *
**********

sum earn78 if treat==0
local N0 = r(N)
local mu0 = r(mean)
local sd0 = r(sd)
local V0 = r(Var)/r(N)
local sig_sq0 = r(Var)

sum earn78 if treat==1
local N1 = r(N)
local mu1 = r(mean)
local sd1 = r(sd)
local V1 = r(Var)/r(N)
local sig_sq1 = r(Var)

local tau = `mu1'-`mu0'
local v = sqrt(`V1'+`V0')
local T = `tau'/`v'
local pval = 2*normal(-abs(`T'))

local mu0 = round(`mu0', .01)
local mu1 = round(`mu1', .0001)
local sd0 = round(`sd0', .01)
local sd1 = round(`sd1', .0001)

di "`tau'"


local CIlower = `tau' - invnormal(0.975)*`v'
local CIupper = `tau' + invnormal(0.975)*`v'

di "`CIlower'"
di "`CIupper'"

**********
* fisher *
**********

* Using difference in means estimator
permute treat diffmean=(r(mu_2)-r(mu_1)), reps(1999) nowarn: ttest earn78, by(treat)
matrix pval = r(p)
display "p-val = " pval[1,1]

* Using KS statistic
permute treat ks=r(D), reps(1999) nowarn: ksmirnov earn78, by(treat)
matrix pval = r(p)
display "p-val = " pval[1,1]

**************************
* 95% confidence interval*
**************************


* Infer missing values under the null of constant treatment effect
gen     Y1_imputed = earn78
replace Y1_imputed = earn78 + `tau' if treat==0

gen     Y0_imputed = earn78
replace Y0_imputed = earn78 - `tau' if treat==1

bootstrap treat diffmean=(r(mu_2)-r(mu_1)), reps(1999) nowarn: ttest earn78, by(treat)

*****************
*power funciton *
*****************

twoway function y= 1 - normal(invnormal(0.975)-x/`v') + normal(-invnormal(0.975)-x/`v'),
```

```
range(-5000 5000)


mata: mata clear
mata:


 function myfunc(N, s0, s1, p, tau){

   return(1 - normal(invnormal(0.975)-tau/sqrt(1/N*s1*(1/p)+1/N*s0*(1/(1-p)))) +
       normal(-invnormal(0.975)-tau/sqrt(1/N*s1*(1/p)+1/N*s0*(1/(1-p)))) -0.8)

 }
 s0 =  30072466.58373794
 s1 =  61896056.06715253
    p     = 2/3
   tau   = 1000
   p
   tau
 s0
 s1


  mm_root(x=., &myfunc(), 1000, 1500, 0, 10000, s0,s1, p ,tau)

     x

end
```