

# Econ 675 Assignment 1

Nathan Mather

September 28, 2018

## Contents

<b>1</b>	<b>Question 1: Simple Linear Regression with Measurement Error</b>	<b>1</b>
1.1	OLS estimator . . . . .	1
1.2	Standard Errors . . . . .	2
1.3	t-test . . . . .	3
1.4	Second measurement, Consistency . . . . .	3
1.5	Second measurement, Distribution . . . . .	3
1.6	Second measurement, Inference . . . . .	3
1.7	Validation sample, Consistency . . . . .	4
1.8	Validation sample, Distribution . . . . .	4
1.9	Validation sample, Inference . . . . .	4
1.10	FE estimator, Consistency . . . . .	5
1.11	FE estimator, time dependence . . . . .	5
1.12	FE estimator, time dependence . . . . .	6
<b>2</b>	<b>Question 2: Implementing Least-Squares Estimators</b>	<b>6</b>
2.1	part 1 . . . . .	6
2.2	Part 2 . . . . .	6
2.3	Part 3 . . . . .	7
2.4	Part 4 . . . . .	7
2.5	Part 5 . . . . .	7
<b>3</b>	<b>Question 3: Analysis of Experiments</b>	<b>9</b>
3.1	Neyman's approach . . . . .	9
3.2	Fisher's approach . . . . .	9
3.3	Power calculations . . . . .	11
<b>4</b>	<b>Appendix</b>	<b>12</b>
4.1	R Code . . . . .	12
4.2	Stata Code . . . . .	22

## 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  $\hat{\beta}_{ls} \rightarrow_p \lambda\beta$

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  $\beta$  is positive and a positive bias in  $\hat{\beta}_{ls}$  if the true  $\beta$  is negative (an attenuation bias). In order to determine the magnitude of the bias consider the following.

$$\begin{aligned}\hat{\beta}_{ls} &= \frac{\text{Cov}(\tilde{x}, y)}{\text{Var}(\tilde{x})} = \frac{\text{Cov}(x + \mu, \beta x + \epsilon)}{\text{Var}(x + \mu)} = \frac{\beta \text{Cov}(x, x) + \text{Cov}(x, \epsilon) + \text{Cov}(\mu, \beta x) + \text{Cov}(\mu, \epsilon)}{\text{Var}(x + \mu)} \\ &= \frac{\beta \text{Var}(x)}{\text{Var}(x + \mu)} \rightarrow_p \frac{\beta \sigma_x^2}{\sigma_x^2 + \sigma_\mu^2} = \lambda \beta\end{aligned}$$

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

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

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

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

$$\begin{aligned}\hat{\sigma}_\epsilon^2(\tilde{x}'\tilde{x}/n)^{-1} &= \frac{\hat{\sigma}_\epsilon^2}{\hat{\sigma}_x^2} \rightarrow_p \frac{\sigma_\epsilon^2 + (1 - \lambda)^2 \beta^2 \sigma_x^2 + \lambda^2 \beta^2 \sigma_\mu^2}{\sigma_x^2 + \sigma_\mu^2} \\ &= \frac{\sigma_x^2}{\sigma_x^2 + \sigma_\mu^2} \left( \frac{\sigma_\epsilon^2}{\sigma_x^2} \right) + \frac{\sigma_x^2}{\sigma_x^2 + \sigma_\mu^2} (1 - \lambda)^2 \beta^2 + \frac{\sigma_\mu^2}{\sigma_x^2 + \sigma_\mu^2} \lambda^2 \beta^2 = \lambda \left( \frac{\sigma_\epsilon^2}{\sigma_x^2} \right) + \lambda (1 - \lambda)^2 \beta^2 + (1 - \lambda) \lambda^2 \beta^2\end{aligned}$$

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

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

multiplying the first term by  $\lambda$  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

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

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  $E[\tilde{x}\epsilon] = 0$

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

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

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

Now to show it is consistent

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

since  $E[\tilde{x}'\epsilon] = 0$  so  $\frac{\tilde{x}'\epsilon}{n} \rightarrow_p 0$  by LLN

### 1.5 Second measurement, Distribution

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

Now using the CLT we get

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

Now all together we get

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

### 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.

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

We also showed above that

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

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

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

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

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

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

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

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

## 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:

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n (\tilde{x}_{i2} - \tilde{x}_{i1})^2 &\rightarrow_p \mathbb{E}[(\tilde{x}_{i2} - \tilde{x}_{i1})^2] = \mathbb{E}[(x_{i2} - x_{i1} + u_{i2} - u_{i1})^2] \\ &= \mathbb{E}[(x_{i2} - x_{i1})^2] + \mathbb{E}[(u_{i2} - u_{i1})^2] + 2\mathbb{E}[(x_{i2} - x_{i1})(u_{i2} - u_{i1})] = \sigma_{\Delta x}^2 + \sigma_{\Delta u}^2 \end{aligned}$$

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

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n (\tilde{x}_{i2} - \tilde{x}_{i1})(y_{i2} - y_{i1}) &\rightarrow_p \mathbb{E}[(\tilde{x}_{i2} - \tilde{x}_{i1})(y_{i2} - y_{i1})] \\ &= \mathbb{E}[(x_{i2} - x_{i1} + u_{i2} - u_{i1})(x_{i2}\beta - x_{i1}\beta + e_{i2} - e_{i1})] \\ &= \mathbb{E}[(x_{i2} - x_{i1})^2]\beta + \mathbb{E}[(x_{i2} - x_{i1})(e_{i2} - e_{i1})] + \mathbb{E}[(x_{i2} - x_{i1})(u_{i2} - u_{i1})]\beta + \mathbb{E}[(u_{i2} - u_{i1})(e_{i2} - e_{i1})] \\ &= \mathbb{E}[(x_{i2} - x_{i1})^2]\beta = \sigma_{\Delta x}^2\beta \end{aligned}$$

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

$$\hat{\beta}_{FE} \rightarrow_p \frac{\sigma_{\Delta x}^2}{\sigma_{\Delta x}^2 + \sigma_{\Delta u}^2} \beta$$

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

### 1.11 FE estimator, time dependence

Covariance stationarity implies that  $\sigma_{xt}^2 = \sigma_x^2$  and  $\sigma_{ut}^2 = \sigma_u^2$  for  $t \in \{1, 2\}$  this means that

$$\sigma_{\Delta x}^2 = \mathbb{V}[x_{i2} - x_{i1}] = \mathbb{V}[x_{i2}] + \mathbb{V}[x_{i1}] - 2\text{Cov}[x_{i2}, x_{i1}] = 2\sigma_x^2 - 2\text{Cov}[x_{i2}, x_{i1}]$$

Thus we get

$$\begin{aligned} \gamma &= \frac{2(\sigma_x^2 - \text{Cov}[x_{i2}, x_{i1}])}{2(\sigma_x^2 - \text{Cov}[x_{i2}, x_{i1}] + \sigma_u^2 - \text{Cov}[u_{i2}, u_{i1}])} \\ &= \frac{\sigma_x^2 - \rho_x \sigma_x^2}{\sigma_x^2 - \rho_x \sigma_x^2 + \sigma_u^2 - \rho_u \sigma_u^2} = \frac{\sigma_x^2(1 - \rho_x)}{\sigma_x^2(1 - \rho_x) + \sigma_u^2(1 - \rho_u)} = \frac{\sigma_x^2}{\sigma_x^2 + \sigma_u^2 \frac{1 - \rho_u}{1 - \rho_x}} \end{aligned}$$

## 1.12 FE estimator, time dependence

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

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

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  $\beta$ . 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

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

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  $\beta$  the first term is irrelevant as it does not include a  $\beta$ . 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  $(\frac{1}{n}x'Wx) \rightarrow_p A$

We also have that  $\sqrt{n}(\frac{1}{n}x'W\epsilon) \rightarrow_d \mathcal{N}(0, B)$  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

The results from R are below

variable	beta	se	t_test	p_value	CI.L	CI.U
const	6485.553	4513.513	1.437	0.151	-2384.895	15356.001
treat	1535.482	638.238	2.406	0.017	281.147	2789.817
black	-2592.377	794.999	-3.261	0.001	-4154.796	-1029.957
age	39.341	40.470	0.972	0.332	-40.196	118.877
educ	-740.540	944.679	-0.784	0.434	-2597.126	1116.046
educ_sq	60.082	53.768	1.117	0.264	-45.589	165.754
earn74	-0.030	0.104	-0.288	0.774	-0.234	0.174
black_earn74	0.175	0.132	1.330	0.184	-0.084	0.434
u74	1316.032	1505.927	0.874	0.383	-1643.580	4275.644
u75	-1167.688	1275.416	-0.916	0.360	-3674.274	1338.898

The results from stata are These are the STATA results

beta	se	t_test	p_value	CI.L	CI.U
6485.5531	4513.5125	1.4369	0.1515	-2385.4508	15356.5570
1535.4824	638.2380	2.4058	0.0166	281.0688	2789.8961
-2592.3766	794.9991	-3.2609	0.0012	-4154.8937	-1029.8595
39.3405	40.4701	0.9721	0.3315	-40.2007	118.8817
-740.5400	944.6787	-0.7839	0.4335	-2597.2421	1116.1622
60.0823	53.7684	1.1174	0.2644	-45.5958	165.7604
-0.0299	0.1037	-0.2879	0.7735	-0.2337	0.1740
0.1754	0.1318	1.3304	0.1841	-0.0837	0.4344
1316.0320	1505.9270	0.8739	0.3827	-1643.7657	4275.8296
-1167.6884	1275.4156	-0.9155	0.3604	-3674.4316	1339.0548

### 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)	6485.553	4513.513	1.437	0.151
treat	1535.482	638.238	2.406	0.017
black	-2592.377	794.999	-3.261	0.001
age	39.341	40.470	0.972	0.332
educ	-740.540	944.679	-0.784	0.434
educ_sq	60.082	53.768	1.117	0.264
earn74	-0.030	0.104	-0.288	0.774
black_earn74	0.175	0.132	1.330	0.184
u74	1316.032	1505.927	0.874	0.383
u75	-1167.688	1275.416	-0.916	0.360

The STATA results also coincide

	(1)	(2)
VARIABLES	earn78	earn78
treat	1,535** (638.2)	1,535** (638.2)
black	-2,592*** (795.0)	-2,592*** (795.0)
age	39.34 (40.47)	39.34 (40.47)
educ	-740.5 (944.7)	-740.5 (944.7)
educ_sq	60.08 (53.77)	60.08 (53.77)
earn74	-0.0299 (0.104)	-0.0299 (0.104)
black_earn74	0.175 (0.132)	0.175 (0.132)
u74	1,316 (1,506)	1,316 (1,506)
u75	-1,168 (1,275)	-1,168 (1,275)
Constant	6,486 (4,514)	6,486 (4,514)
Observations	445	445
R-squared	0.063	0.063

Robust standard errors in parentheses

\*\*\* p<0.01, \*\* p<0.05, \* p<0.1



### 3 Question 3: Analysis of Experiments

#### 3.1 Neyman's approach

##### 3.1.1 a

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

Now note that since  $T_i$  is random:

$$E[Y_i(T_i)|T_i = 1] = E[Y_i(1)]$$

$$E[Y_i(T_i)|T_i = 0] = E[Y_i(0)]$$

Together this gives us:

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

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

The results from R are in the table below. The results from STATA were identical. these are the R results

TDM est	Conservative SE	CI Lower	CI Upper
1794.343	670.997	479.214	3109.473

The STATA output is identical

#### 3.2 Fisher's approach

##### 3.2.1 a

The results in R are in the tables below.

DM P value
0.006
KS P value
0.040

The results in STATA were .0050025 and .03651826 respectively. So they do not differ by much. There is some randomness to this because of the bootstrap so it is expected that its not the same.

### 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
5000.000	0.000
4750.000	0.000
4500.000	0.000
4250.000	0.001
4000.000	0.001
3750.000	0.002
3500.000	0.004
3250.000	0.025
3000.000	0.055
2750.000	0.134
2500.000	0.267
2250.000	0.466
2000.000	0.748
1750.000	0.940
1500.000	0.645
1250.000	0.415
1000.000	0.214
750.000	0.096
500.000	0.042
250.000	0.019
0.000	0.003
-250.000	0.002
-500.000	0.001
-750.000	0.000
-1000.000	0.000
-1250.000	0.000
-1500.000	0.000

In STATA using the bootstrap method we get

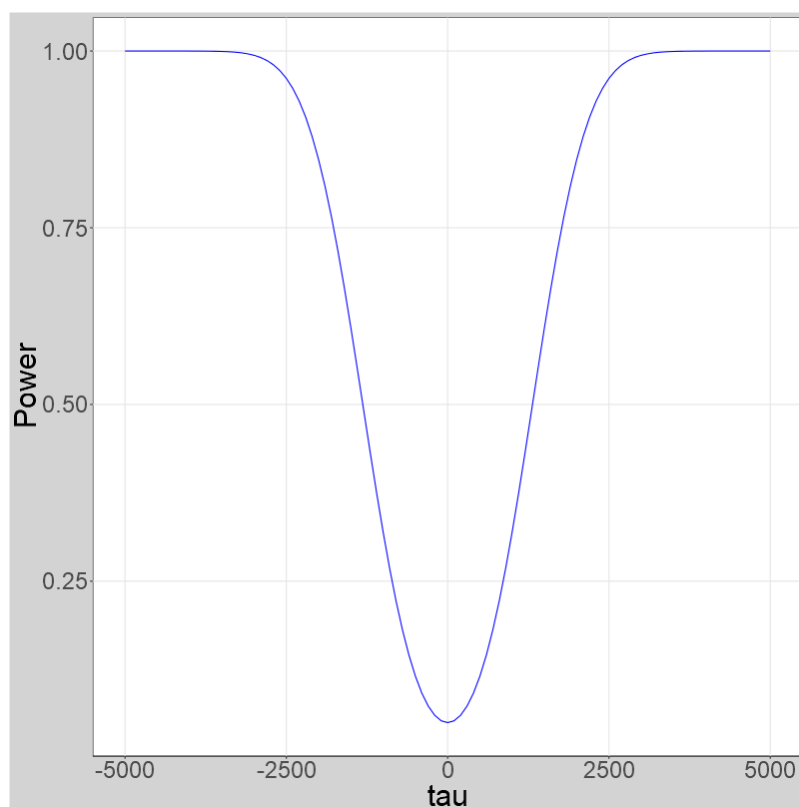
	Observed Coef.	Bootstrap Std. Err.	z	P> z	Normal-based [95% Conf. Interval]	
diff	<b>1794.343</b>	<b>670.1254</b>	<b>2.68</b>	<b>0.007</b>	<b>480.9214</b>	<b>3107.765</b>

The numbers are different because in one case I use the chart method and in the other I use the bootstrap. The bootstrap method gives similar results to R in from 3.1 part B using the neyman approach.

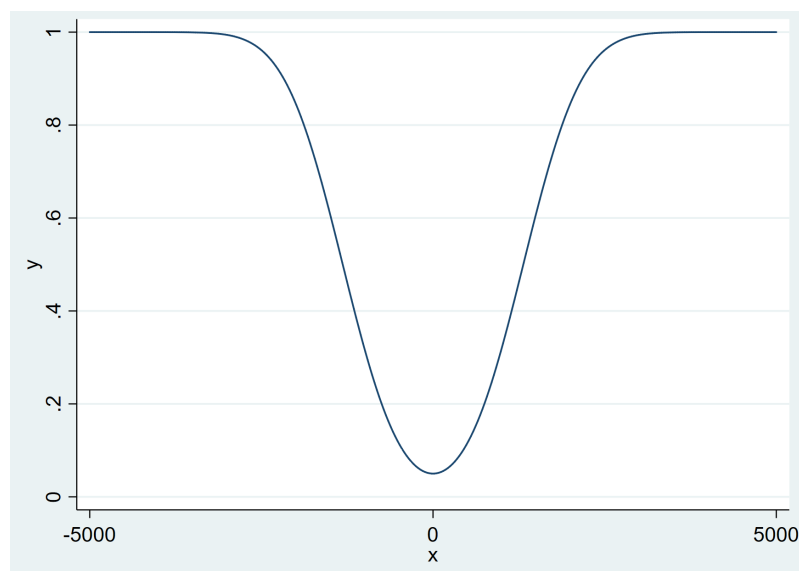
### 3.3 Power calculations

#### 3.3.1 a

Here is the graph from R



and now the graph from stata



#### 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 ====  
#####  
  
#####  
# ==== generate 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 mean
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
lalonge_dt <- fread(paste0("C:/Users/Nmath_000/Documents/MI_school/Second ",
                           "Year/675 Applied Econometrics/hw/hw1/LaLonde_1986.csv"))

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

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

```

```

lalonge_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(lalonge_dt[, c("const", x_vars), with = FALSE])

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

# grab the results
results_2_5_a <- lalonge_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
lalonge_lm <- lm(reg_form, lalonge_dt)

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

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

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

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

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

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

```



```

# get variance for treatment and no treatment
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")

```

```

#####
# ==== 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 ====
#####

```

```

# save plot #note paste0 is so it fits on markdown pdf
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)

#####
# ==== run markdown to print code ====
#####

rmarkdown::render(input = "C:/Users/Nmath_000/Documents/Code/courses/econ 675/ps_1_675_markdown.Rmd",
                  output_format = "pdf_document",
                  output_file = paste0("C:/Users/Nmath_000/Documents/Code/courses/econ 675/PS_1_tex/"))

```

## 4.2 Stata Code

```

1  clear all
2  set more off, perm
3
4  * set working directory
5  global dir "C:\Users\Nmath_000\Documents\MI_school\Second Year\675 Applied
   Econometrics\hw\hw1"
6
7  *import data
8  import delimited using "$dir\LaLonde_1986.csv"
9
10 *****
11 * question 2 *
12 *****
13
14
15 * create needed variables
16 gen educ_sq = educ^2
17 gen black_earn74 = black*earn74
18 gen const = 1
19
20 * store needed variables in locals
21 *local y earn76
22 *local x const treat black age educ educ_sq earn74 black_earn74 u74 u75
23
24 * use mata
25 mata:
26
27
28 y = st_data(., "earn78")
29 x = st_data(., ("const", "treat", "black", "age", "educ", "educ_sq", "earn74", "black_earn74",
   , "u74", "u75"))
30
31 n_row = rows(x)
32 n_col = cols(x)
33
34 b = invsym(cross(x,x))*cross(x,y)
35
36 bc = cholinv(cross(x,x))*cross(x,y)
37
38 diff = b-bc
39
40 diff
41
42 my_resid = y - x*b
43 d = diag(my_resid:*my_resid:*(n_row/(n_row-n_col)))
44
45 v = invsym(cross(x, x))*(x' * d * x) * invsym(cross(x, x))
46
47 se = sqrt(diagonal(v))
48
49 tstat = b ./ se
50
51 p_value = 2*ttail(n_row-n_col, abs(tstat))
52
53 CI_L = b - (se) * invt(n_row-n_col, .975 )
54 CI_U = b + (se) * invt(n_row-n_col, .975 )
55
56 all_data = b, se, tstat, p_value, CI_L, CI_U
57 all_data
58 end
59
60 cd "C:\Users\Nmath_000\Documents\Code\courses\econ 675\PS_1_tex\"
61 mmat2tex all_data using stata_2_5_a_raw.tex , replace
62
63 // now run regression
64 reg earn78 treat black age educ educ_sq earn74 black_earn74 u74 u75, robust
65
66 outreg2 using stata_2_5_b.tex
67
68 // nice, they match

```

```

69
70 *****
71 * question 3 *
72 *****
73
74 *****
75 * neyman *
76 *****
77
78 sum earn78 if treat==0
79 local N0 = r(N)
80 local mu0 = r(mean)
81 local sd0 = r(sd)
82 local V0 = r(Var)/r(N)
83 local sig_sq0 = r(Var)
84
85 sum earn78 if treat==1
86 local N1 = r(N)
87 local mu1 = r(mean)
88 local sd1 = r(sd)
89 local V1 = r(Var)/r(N)
90 local sig_sq1 = r(Var)
91
92 local tau = `mu1' - `mu0'
93 local v = sqrt(`V1' + `V0')
94 local T = `tau' / `v'
95 local pval = 2*normal(-abs(`T'))
96
97 local mu0 = round(`mu0', .01)
98 local mu1 = round(`mu1', .0001)
99 local sd0 = round(`sd0', .01)
100 local sd1 = round(`sd1', .0001)
101
102 di "`tau'"
103
104
105 local CIlower = `tau' - invnormal(0.975)*`v'
106 local CIupper = `tau' + invnormal(0.975)*`v'
107
108 di "`CIlower'"
109 di "`CIupper'"
110
111 *****
112 * fisher *
113 *****
114
115
116 * Using difference in means estimator
117 permute treat diffmean=(r(mu_2)-r(mu_1)), reps(1999) nowarn: ttest earn78, by(treat)
118 matrix pval = r(p)
119 display "p-val = " pval[1,1]
120
121 * Using KS statistic
122 permute treat ks=r(D), reps(1999) nowarn: ksmirnov earn78, by(treat)
123 matrix pval = r(p)
124 display "p-val = " pval[1,1]
125
126 *****
127 * 95% confidence interval*
128 *****
129
130
131 * Infer missing values under the null of constant treatment effect
132 gen Y1_imputed = earn78
133 replace Y1_imputed = earn78 + `tau' if treat==0
134
135 gen Y0_imputed = earn78
136 replace Y0_imputed = earn78 - `tau' if treat==1
137
138 * Write program to put into bootstrap function

```



```

139 program define meandiff, rclass
140     summarize    Y1 imputed if treat==1
141     local        tau1 = r(mean)
142     sum          Y0 imputed if treat==0
143     local        tau0 = r(mean)
144     return       scalar meandiff = `tau1' - `tau0'
145 end
146
147 * Run bootstrap function using meandiff program
148 eststo I: bootstrap diff = r(meandiff), reps(1999): meandiff
149
150 esttab I using stata_3_2_2_b.tex, mtitle("I") replace
151
152 *****
153 *power funciton *
154 *****
155
156 twoway function y= 1 - normal(invnormal(0.975)-x/`v') + normal(-invnormal(0.975)-x/`v'),
range(-5000 5000)
157
158
159 mata: mata clear
160 mata:
161
162
163     function myfunc(N, s0, s1, p, tau){
164
165         return(1 - normal(invnormal(0.975)-tau/sqrt(1/N*s1*(1/p)+1/N*s0*(1/(1-p)))) +
166             normal(-invnormal(0.975)-tau/sqrt(1/N*s1*(1/p)+1/N*s0*(1/(1-p)))) -0.8)
167
168     }
169     s0 = 30072466.58373794
170     s1 = 61896056.06715253
171     p   = 2/3
172     tau = 1000
173     p
174     tau
175     s0
176     s1
177
178
179     mm_root(x=., &myfunc(), 1000, 1500, 0, 10000, s0,s1, p ,tau)
180
181     x
182
183 end
184
185
186
187

```