Economics 675: Applied Microeconometrics
Fall 2018 - Assignment 4

Paul R. Organ*

November 11, 2018

Contents

*prorgan@umich.edu

1 Question 1: Estimating Equations

1.1 Moment Conditions

For each estimator, we use the Law of Iterated Expectations to show the desired result.

(a) IPW: use that $D_i(t) = \mathbf{1}(T_i = t)$ and $p_t(\mathbf{X}_i) = \mathbb{P}[T_i = t|\mathbf{X}_i]$.

$$\begin{aligned}\mathbb{E}\Big[\frac{D_i(t)\cdot g(Y_i(t))}{p_t(\mathbf{X}_i)}\Big] &= \mathbb{E}\left[\mathbb{E}\Big[\frac{D_i(t)\cdot g(Y_i(t))}{p_t(\mathbf{X}_i)}|\mathbf{X}_i\Big]\right] \\ &= \mathbb{E}\left[\mathbb{E}\Big[\frac{D_i(t)}{p_t(\mathbf{X}_i)}|\mathbf{X}_i\Big]\mathbb{E}\Big[g(Y_i(t))|\mathbf{X}_i\Big]\right] \\ &= \mathbb{E}\left[\mathbb{E}\Big[g(Y_i(t))|\mathbf{X}_i\Big]\right] = \mathbb{E}[g(Y_i(t))] = \theta_t(g)\end{aligned}$$

(b) Regression Imputation: simply plug in for $e_t(g;\mathbf{X}_i)$, and by LIE the result follows.

$$\mathbb{E}[e_t(g;\mathbf{X}_i)] = \mathbb{E}\big[\mathbb{E}[g(Y_i(t))|\mathbf{X}_i]\big] = \mathbb{E}[g(Y_i(t))] = \theta_t(g)$$

(c) Doubly Robust: by LIE, we can show that the third term equals zero in expectation, and the result for IPW will hold here.

$$\begin{aligned}\mathbb{E}\Big[\frac{e_t(g;\mathbf{X}_i)}{p_t(\mathbf{X}_i)}\cdot(D_i(t)-p_t(\mathbf{X}_i))\Big] &= \mathbb{E}\left[\mathbb{E}\Big[\frac{e_t(g;\mathbf{X}_i)}{p_t(\mathbf{X}_i)}\cdot(D_i(t)-p_t(\mathbf{X}_i))|\mathbf{X}_i\Big]\right] \\ &= \mathbb{E}\left[\frac{e_t(g;\mathbf{X}_i)}{p_t(\mathbf{X}_i)}\cdot\mathbb{E}\Big[(D_i(t)-p_t(\mathbf{X}_i))|\mathbf{X}_i\Big]\right] \\ &= \mathbb{E}\left[\frac{e_t(g;\mathbf{X}_i)}{p_t(\mathbf{X}_i)}\cdot 0\right] = 0\end{aligned}$$

1.2 Plug-in Estimators for $\theta_t(g)$

Looking at the equations above, we see that most of the components are simply in the data. What must be estimated are the functions e_t and p_t (the conditional expectation and the propensity score). Once these are estimated, it is straightforward to plug the estimates into the different forms of $\theta_t(g)$. We can estimate p_t using a probit or logit regression, and estimate e_t using nonparametric regression of Y_i on $\mathbf{X}_i$, for the treated observations.

1.3 Causal Estimators for σ_t^2

We want to estimate $\sigma_2^2 = \mathbb{V}[Y_i(t)]$. Note that $\mathbb{V}[Y_i(t)] = \mathbb{E}[Y_i(t)^2] - (\mathbb{E}[Y_i(t)])^2$, so we need to estimate each of these two components. We can do this using probit, logit, nonparametric regression, etc.

Once we estimate the two components and plug them in, we have an estimate for σ_t^2. We could apply the delta method to our two components to determine the resulting distribution of our estimate, and do inference as desired.

2 Question 2: Estimating Average Treatment Effects

See the code in Appendix A (R) and Appendix B (Stata). The tables containing my results are below.

A brief discussion of things I noted during this exercise:

- PSID data is much larger than the original experimental data, so we end up with a big imbalance in terms of the size of control vs. treatment, relative to the initial experimental data.
- There may be a big difference in the characteristics of the PSID sample, as that control appears to give very different results.
- Although we see that the results can change quite a bit between different models and within models (switching among different covariates), it is somewhat comforting that the confidence intervals generally all show a consistent qualitative result.
- The built-in commands for Stata were excellent for these applications. In R, I had to search harder to find packages that would accomplish what we want with the necessary flexibility, and didn't even find everything I wanted (hence some holes in my ATT table for R).

Table 1: ATE Results: Stata

	Experimental Data				PSID Control			
	tau	se	CI low	CI high	tau	se	CI low	CI high
Diff-in-Means	1,794.34	671.00	475.61	3,113.07	-15,204.78	657.08	-16,493.21	-13,916.35
OLS: a	1,582.17	658.74	287.46	2,876.88	6,302.40	1,209.32	3,931.10	8,673.69
OLS: b	1,506.90	657.12	215.35	2,798.45	4,699.26	1,027.12	2,685.23	6,713.29
OLS: c	1,501.37	662.91	198.41	2,804.34	4,284.34	1,031.34	2,262.03	6,306.65
RI: a	1,462.27	642.24	203.50	2,721.04	-11,195.04	1,741.33	-14,607.97	-7,782.10
RI: b	1,454.13	643.49	192.92	2,715.34	-10,398.22	3,293.40	-16,853.16	-3,943.28
RI: c	1,427.53	642.95	167.37	2,687.69	-11,920.18	3,834.63	-19,435.92	-4,404.44
IPW: a	1,539.44	646.77	271.79	2,807.09	-9,047.57	2,787.86	-14,511.67	-3,583.47
IPW: b	1,469.95	646.42	202.99	2,736.92	-381.76	3,737.29	-7,706.71	6,943.19
IPW: c	1,467.59	645.78	201.88	2,733.30	-2,477.38	3,637.74	-9,607.22	4,652.47
DR: a	1,476.66	641.95	218.47	2,734.85	-9,983.53	1,290.82	-12,513.48	-7,453.58
DR: b	1,450.72	641.84	192.73	2,708.70	-1,909.14	2,056.36	-5,939.53	2,121.25
DR: c	1,406.72	648.27	136.13	2,677.31	-3,321.24	1,701.09	-6,655.31	12.83
NN: a	1,829.80	779.60	301.81	3,357.78	-15,619.49	1,153.32	-17,879.95	-13,359.03
NN: b	1,875.83	734.95	435.35	3,316.31	-9,349.56	3,974.77	-17,139.97	-1,559.16
NN: c	1,671.74	726.06	248.69	3,094.79	-9,561.97	4,033.54	-17,467.56	-1,656.37
Propensity: a	1,566.12	635.54	320.48	2,811.76	-9,190.27	4,894.19	-18,782.70	402.16
Propensity: b	1,343.83	751.12	-128.35	2,816.01	7,363.27	9,865.01	-11,971.80	26,698.33
Propensity: c	1,700.98	710.38	308.66	3,093.30	6,688.80	11,027.80	-14,925.30	28,302.89

Table 2: ATE Results: R

	Experimental Data				PSID Control			
	tau	se	CI low	CI high	tau	se	CI low	CI high
Diff-in-Means	1794.34	671.00	479.19	3109.50	-15204.78	657.08	-16492.65	-13916.91
OLS: a	1582.17	658.74	291.04	2873.30	6302.40	1209.32	3932.13	8672.66
OLS: b	1506.90	657.12	218.95	2794.85	4699.26	1027.12	2686.11	6712.41
OLS: c	1501.37	662.91	202.07	2800.68	4284.34	1031.34	2262.91	6305.77
RI: a	1462.27	128.12	1211.16	1713.38	-11195.04	187.60	-11562.73	-10827.35
RI: b	1454.13	147.58	1164.86	1743.39	-10398.22	187.14	-10765.02	-10031.42
RI: c	1427.53	162.08	1109.85	1745.20	-11920.18	210.01	-12331.80	-11508.56
IPW: a	1537.40	629.76	303.07	2771.72	-12297.56	885.51	-14033.16	-10561.96
IPW: b	1469.62	630.67	233.51	2705.72	-11620.39	856.71	-13299.53	-9941.24
IPW: c	1468.10	641.70	210.37	2725.84	-13091.69	692.05	-14448.11	-11735.28
DR: a	1473.17	629.75	238.87	2707.48	-11756.52	885.32	-13491.74	-10021.29
DR: b	1451.05	630.67	214.95	2687.16	-11072.66	856.48	-12751.36	-9393.96
DR: c	1423.88	641.70	166.15	2681.61	-12447.86	691.75	-13803.69	-11092.02
NN: a	1311.82	371.29	584.09	2039.55	801.08	498.21	-175.40	1777.57
NN: b	1811.39	369.81	1086.56	2536.21	119.37	532.14	-923.63	1162.36
NN: c	1925.96	362.83	1214.81	2637.10	65.93	546.32	-1004.86	1136.72
Propensity: a	1349.50	369.26	625.76	2073.24	910.40	498.89	-67.42	1888.21
Propensity: b	1643.99	377.06	904.96	2383.02	140.42	543.13	-924.13	1204.96
Propensity: c	1768.10	365.87	1050.98	2485.21	1.94	541.63	-1059.66	1063.54

Table 3: ATT Results: Stata

	Experimental Data				PSID Control			
	tau	se	CI low	CI high	tau	se	CI low	CI high
Diff-in-Means	1,794.34	671.00	475.61	3,113.07	-15,204.78	657.08	-16,493.21	-13,916.35
OLS: a	1,582.17	658.74	287.46	2,876.88	6,302.40	1,209.32	3,931.10	8,673.69
OLS: b	1,506.90	657.12	215.35	2,798.45	4,699.26	1,027.12	2,685.23	6,713.29
OLS: c	1,501.37	662.91	198.41	2,804.34	4,284.34	1,031.34	2,262.03	6,306.65
RI: a	1,726.60	688.76	376.65	3,076.55	8,543.16	1,233.32	6,125.89	10,960.43
RI: b	1,809.70	693.87	449.74	3,169.65	4,932.66	1,088.73	2,798.79	7,066.52
RI: c	1,844.61	694.82	482.78	3,206.43	5,558.08	1,074.07	3,452.94	7,663.23
IPW: a	1,765.85	697.80	398.20	3,133.51	2,721.88	887.40	982.61	4,461.15
IPW: b	1,742.79	701.03	368.80	3,116.78	2,203.26	935.74	369.26	4,037.27
IPW: c	1,777.07	700.65	403.82	3,150.31	2,422.25	968.31	524.40	4,320.10
DR: a	1,727.42	692.29	370.56	3,084.28	1,918.64	732.20	483.55	3,353.73
DR: b	1,759.51	689.39	408.33	3,110.68	2,179.83	745.06	719.53	3,640.12
DR: c	1,834.75	695.03	472.52	3,196.97	2,272.77	828.99	647.99	3,897.56
NN: a	1,558.16	776.73	35.79	3,080.52	1,671.33	1,226.74	-733.03	4,075.70
NN: b	1,731.61	732.36	296.20	3,167.01	2,043.34	1,006.33	70.96	4,015.71
NN: c	1,137.43	813.44	-456.88	2,731.73	2,003.80	935.02	171.19	3,836.40
Propensity: a	1,708.00	775.66	187.73	3,228.27	2,641.14	814.39	1,044.97	4,237.32
Propensity: b	1,854.86	788.39	309.64	3,400.07	1,364.56	1,132.33	-854.76	3,583.89
Propensity: c	1,888.36	779.07	361.41	3,415.31	1,332.53	1,292.20	-1,200.15	3,865.20

Table 4: ATT Results: R

	Experimental Data				PSID Control			
	tau	se	CI low	CI high	tau	se	CI low	CI high
Diff-in-Means	1794.34	671.00	479.19	3109.50	-15204.78	657.08	-16492.65	-13916.91
OLS: a	1582.17	658.74	291.04	2873.30	6302.40	1209.32	3932.13	8672.66
OLS: b	1506.90	657.12	218.95	2794.85	4699.26	1027.12	2686.11	6712.41
OLS: c	1501.37	662.91	202.07	2800.68	4284.34	1031.34	2262.91	6305.77
RI: a	1726.60	195.68	1343.07	2110.14	8543.16	716.18	7139.44	9946.88
RI: b	1809.70	245.13	1329.24	2290.15	4932.66	494.80	3962.85	5902.46
RI: c	1844.61	253.48	1347.79	2341.42	5558.08	512.24	4554.09	6562.08
IPW: a	1737.12				1764.69			
IPW: b	1723.66				-1808.50			
IPW: c	1781.97				-894.15			
DR: a								
DR: b								
DR: c								
NN: a	1570.53	201.68	1175.23	1965.83	3474.95	161.79	3157.84	3792.06
NN: b	1798.66	189.45	1427.34	2169.98	1405.11	231.54	951.29	1858.92
NN: c	2161.69	172.13	1824.32	2499.07	1802.20	207.61	1395.28	2209.12
Propensity: a	1492.42	206.23	1088.21	1896.62	3692.47	157.61	3383.56	4001.38
Propensity: b	1591.83	204.00	1192.00	1991.66	1905.45	199.81	1513.81	2297.08
Propensity: c	1833.09	191.05	1458.64	2207.54	1516.56	226.46	1072.71	1960.41

3 Question 3: Post-model Selection Inference

For all parts of Question 3, see the code in Appendix A (R) and Appendix B (Stata).

Recall from the problem set that we have three estimators:

1. $\hat{\beta}$, from the "long" regression (equation (2) in the problem set);
2. $\tilde{\beta}$, from the "short" regression (equation (3) in the problem set);
3. $\check{\beta}$, equal to $\hat{\beta}$ if the t-statistic on $\hat{\gamma}$ from the "long" regression is at least 1.96, and equal to $\tilde{\beta}$ otherwise.

3.1 Summary Statistics and Kernel Density Plots

Table 5 shows summary statistics for my simulated estimators from R; Table 6 shows the same from Stata.

Table 5: Summary Statistics: R

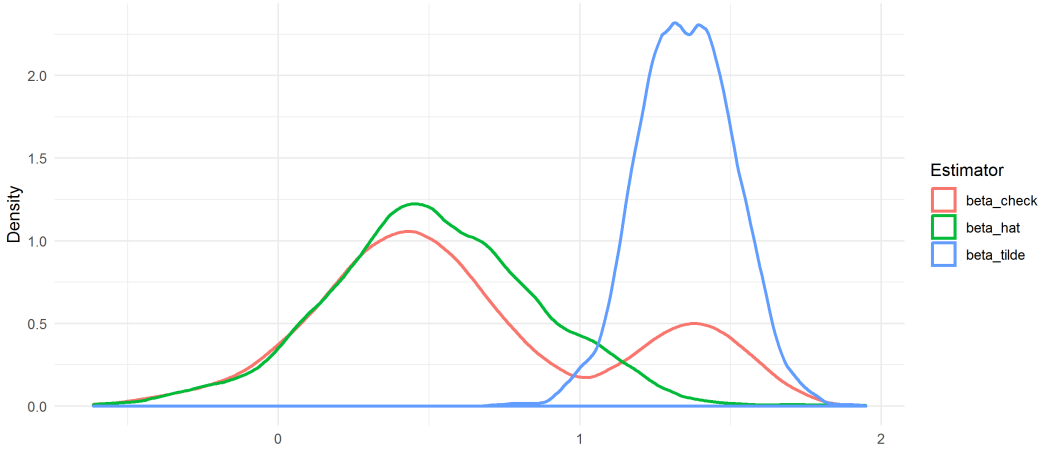
	min	mean	median	max
$\hat{\beta}$	-0.61	0.51	0.50	1.95
$\tilde{\beta}$	0.76	1.35	1.35	1.88
$\check{\beta}$	-0.61	0.63	0.51	1.79

Table 6: Summary Statistics: Stata

	min	mean	median	max
$\hat{\beta}$	-0.55	0.50	0.50	1.64
$\tilde{\beta}$	0.94	1.36	1.36	1.94
$\check{\beta}$	-0.55	0.61	0.61	1.86

Figure 1 shows kernel density plots (using the Epanechnikov kernel) for my simulated estimators from R; Figure 2 shows the same from Stata.

Figure 1: Kernel Density Plots: R



3.2 Empirical Coverage Rates

We know the true value of β_0 is 0.5. In running the replication 1,000 times, we also check if the 95% confidence intervals for each estimator include this true value of β_0. The summary results are presented in Table 7.

Figure 2: Kernel Density Plots: STATA

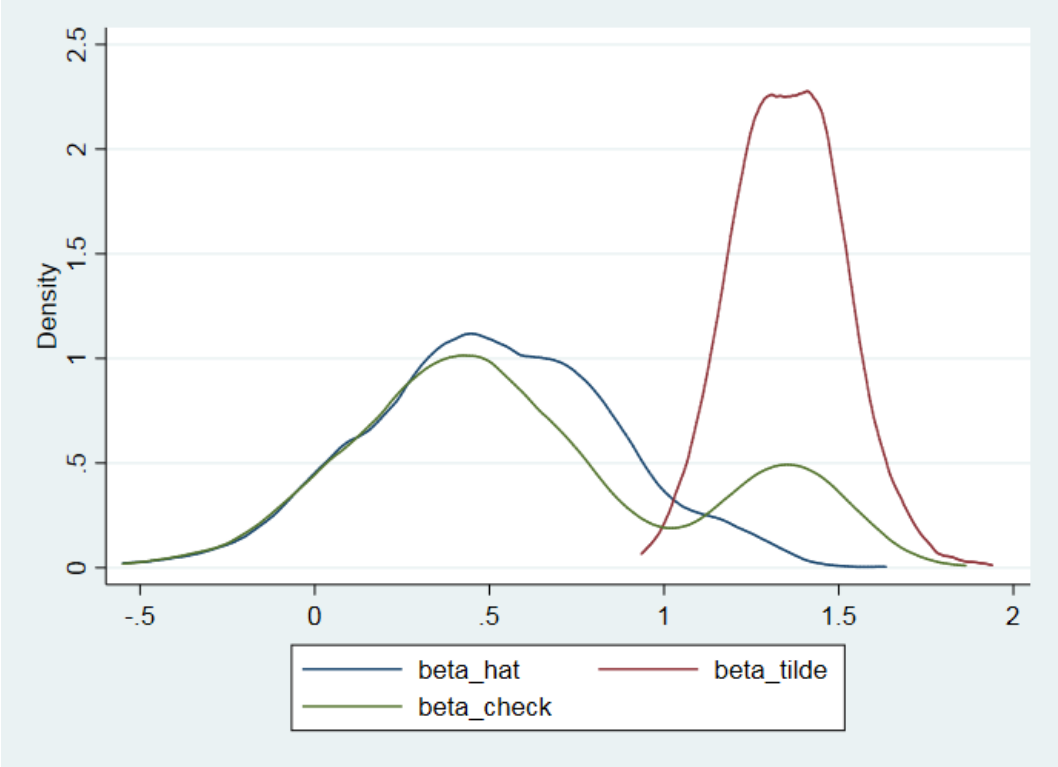


Table 7: Empirical Coverage Rates

	$\hat{\beta}$	$\tilde{\beta}$	$\breve{\beta}$
R	93.6%	16.8%	78.0%
STATA	93.9%	0.01%	74.2%

A R Code

```
#############################################################################################
# Author: Paul R. Organ
# Purpose: ECON 675, PS4
# Last Update: Nov 11, 2018
#############################################################################################
# Preliminaries
options(stringsAsFactors = F)

# packages
require(tidyverse) # data cleaning and manipulation
require(magrittr)  # syntax
require(ggplot2)   # plots
require(sandwich)  # robust standard errors
require(xtable)    # tables for LaTeX
require(boot)      # bootstrapping
require(MatchIt)   # matching
require(CausalGAM) # IPW and Doubly Robust

setwd('C:/Users/prorgan/Box/Classes/Econ 675/Problem Sets/PS4')

#############################################################################################
### Question 2: Estimating Average Treatment Effects
#############################################################################################
# Q2 setup

# load data
df <- read_csv('LaLonde_all.csv')

# add treatment variable in expanded dataset
df %<>% mutate(t = ifelse(treat == 1, 1, 0))

# add other variables we will use in the analysis
df %<>% mutate(logre74 = log(re74 + 1), logre75 = log(re75 + 1),
               age2 = age^2, educ2 = educ^2, age3 = age^3,
               bXu74 = black * u74, eXre74 = educ * logre74)

# empty s to fill with results (1 + 6*3)
ate <- matrix(NA, nrow=19,ncol=8) %>% as.data.frame

# add headings
names(ate) <- c('e_tau', 'e_se', 'e_CI_l', 'e_CI_h', 'p_tau', 'p_se',
                'p_CI_l', 'p_CI_h')

# we need one for ATE, one for ATT
att <- ate

# define three model specifications
va <- c('age', 'educ', 'black', 'hisp', 'married',
        'nodegr', 'logre74', 'logre75')
vb <- c(va, 'age2', 'educ2', 'u74', 'u75')
vc <- c(vb, 'age3', 'bXu74', 'eXre74')
```

```
a <- paste('re78', paste(c('t', va),
                         collapse = ' + '), sep = ' ~ ') %>% as.formula()
b <- paste('re78', paste(c('t', vb),
                         collapse = ' + '), sep = ' ~ ') %>% as.formula()
c <- paste('re78', paste(c('t', vc),
                         collapse = ' + '), sep = ' ~ ') %>% as.formula()

ta <- paste('t', paste(va, collapse = ' + '), sep = ' ~ ') %>% as.formula()
tb <- paste('t', paste(vb, collapse = ' + '), sep = ' ~ ') %>% as.formula()
tc <- paste('t', paste(vc, collapse = ' + '), sep = ' ~ ') %>% as.formula()

ya <- paste('re78', paste(va, collapse = ' + '), sep = ' ~ ') %>% as.formula()
yb <- paste('re78', paste(vb, collapse = ' + '), sep = ' ~ ') %>% as.formula()
yc <- paste('re78', paste(vc, collapse = ' + '), sep = ' ~ ') %>% as.formula()

################################################################################
# Q2.1: Difference-in-Means

# experimental data
r1e <- lm(re78 ~ t, data = df %>% filter(treat != 2))

# extract tau
ate$e_tau[1]  <- r1e$coefficients['t']

# hederosketastic robust standard errors
ate$e_se[1]   <- diag(vcovHC(r1e, type = "HC2")) %>% sqrt() %>% .['t']

# PSID data
r1p <- lm(re78 ~ t, data = df %>% filter(treat != 0))

# extract tau
ate$p_tau[1]  <- r1p$coefficients['t']

# hederosketastic robust standard errors
ate$p_se[1]   <- diag(vcovHC(r1p, type = "HC2")) %>% sqrt() %>% .['t']

# ATE and ATT are same here
att[1,] <- ate[1,]

################################################################################
# Q2.2: Linear Least-Squares

# three specifications for each

# OLS for experimental data
r2ea <- lm(a, data = df %>% filter(treat != 2))
r2eb <- lm(b, data = df %>% filter(treat != 2))
r2ec <- lm(c, data = df %>% filter(treat != 2))

# extract tau
ate$e_tau[2]  <- r2ea$coefficients['t']
ate$e_tau[3]  <- r2eb$coefficients['t']
ate$e_tau[4]  <- r2ec$coefficients['t']
```

```
# robust standard errors
ate$e_se[2]   <- diag(vcovHC(r2ea, type = "HC1")) %>% sqrt() %>% .['t']
ate$e_se[3]   <- diag(vcovHC(r2eb, type = "HC1")) %>% sqrt() %>% .['t']
ate$e_se[4]   <- diag(vcovHC(r2ec, type = "HC1")) %>% sqrt() %>% .['t']

# OLS for PSID data
r2pa <- lm(a, data = df %>% filter(treat != 0))
r2pb <- lm(b, data = df %>% filter(treat != 0))
r2pc <- lm(c, data = df %>% filter(treat != 0))

# extract tau
ate$p_tau[2]  <- r2pa$coefficients['t']
ate$p_tau[3]  <- r2pb$coefficients['t']
ate$p_tau[4]  <- r2pc$coefficients['t']

# robust standard errors
ate$p_se[2]   <- diag(vcovHC(r2pa, type = "HC1")) %>% sqrt() %>% .['t']
ate$p_se[3]   <- diag(vcovHC(r2pb, type = "HC1")) %>% sqrt() %>% .['t']
ate$p_se[4]   <- diag(vcovHC(r2pc, type = "HC1")) %>% sqrt() %>% .['t']

# again, ATE is same as ATT here
att[2:4,] <- ate[2:4,]

##############################################################################
# Q2.3: Regression Imputation (rows 5-7)

df_e <- df %>% filter(treat != 2)
df_p <- df %>% filter(treat != 0)

n_e <- nrow(df_e)
n_p <- nrow(df_p)

## covariates a
# treated
beta1 <- lm(ya, data = df %>% filter(treat == 1)) %>% .$coefficients
# untreated experimental
beta0_e <- lm(ya, data = df %>% filter(treat == 0)) %>% .$coefficients
# untreated PSID
beta0_p <- lm(ya, data = df %>% filter(treat == 2)) %>% .$coefficients

# tauhat for each
Z_e <- as.matrix(df_e[,va]) %>% cbind(1,.)
tauhat_e <- Z_e %*% (beta1-beta0_e)

Z_p <- as.matrix(df_p[,va]) %>% cbind(1,.)
tauhat_p <- Z_p %*% (beta1-beta0_p)

# save results (ATE)
ate[5,1] <- mean(tauhat_e)
ate[5,2] <- sd(tauhat_e)/sqrt(n_e) # not right!

ate[5,5] <- mean(tauhat_p)
ate[5,6] <- sd(tauhat_p)/sqrt(n_p) # not right!
```

```
# save results (ATT)
att[5,1] <- mean(tauhat_e[df_e$t==1])
att[5,2] <- sd(tauhat_e[df_e$t==1])/sqrt(sum(df_e$t==1)) # not right!

att[5,5] <- mean(tauhat_p[df_p$t==1])
att[5,6] <- sd(tauhat_p[df_p$t==1])/sqrt(sum(df_p$t==1)) # not right!

## covariates b
# treated
beta1 <- lm(yb, data = df %>% filter(treat == 1)) %>% .$coefficients
# untreated experimental
beta0_e <- lm(yb, data = df %>% filter(treat == 0)) %>% .$coefficients
# untreated PSID
beta0_p <- lm(yb, data = df %>% filter(treat == 2)) %>% .$coefficients

# tauhat for each
Z_e <- as.matrix(df_e[,vb]) %>% cbind(1,.)
tauhat_e <- Z_e %*% (beta1-beta0_e)

Z_p <- as.matrix(df_p[,vb]) %>% cbind(1,.)
tauhat_p <- Z_p %*% (beta1-beta0_p)

# save results (ATE)
ate[6,1] <- mean(tauhat_e)
ate[6,2] <- sd(tauhat_e)/sqrt(n_e) # not right!

ate[6,5] <- mean(tauhat_p)
ate[6,6] <- sd(tauhat_p)/sqrt(n_p) # not right!

# save results (ATT)
att[6,1] <- mean(tauhat_e[df_e$t==1])
att[6,2] <- sd(tauhat_e[df_e$t==1])/sqrt(sum(df_e$t==1)) # not right!

att[6,5] <- mean(tauhat_p[df_p$t==1])
att[6,6] <- sd(tauhat_p[df_p$t==1])/sqrt(sum(df_p$t==1)) # not right!

## covariates c
# treated
beta1 <- lm(yc, data = df %>% filter(treat == 1)) %>% .$coefficients
# untreated experimental
beta0_e <- lm(yc, data = df %>% filter(treat == 0)) %>% .$coefficients
# untreated PSID
beta0_p <- lm(yc, data = df %>% filter(treat == 2)) %>% .$coefficients

# tauhat for each
Z_e <- as.matrix(df_e[,vc]) %>% cbind(1,.)
tauhat_e <- Z_e %*% (beta1-beta0_e)

Z_p <- as.matrix(df_p[,vc]) %>% cbind(1,.)
tauhat_p <- Z_p %*% (beta1-beta0_p)

# save results (ATE)
ate[7,1] <- mean(tauhat_e)
ate[7,2] <- sd(tauhat_e)/sqrt(n_e) # not right!
```

```
ate[7,5] <- mean(tauhat_p)
ate[7,6] <- sd(tauhat_p)/sqrt(n_p) # not right!

# save results (ATT)
att[7,1] <- mean(tauhat_e[df_e$t==1])
att[7,2] <- sd(tauhat_e[df_e$t==1])/sqrt(sum(df_e$t==1)) # not right!

att[7,5] <- mean(tauhat_p[df_p$t==1])
att[7,6] <- sd(tauhat_p[df_p$t==1])/sqrt(sum(df_p$t==1)) # not right!

################################################################################
# Q2.4: Inverse Probability Weighting (rows 8-10)

# this is my initial approach, superseded by use of CausalGAM package later
# trimming for PSID data drops a lot of observations
# not really matching my Stata results

# first estimate propensity scores (we use a probit here)
r4ae <- glm(ta, data = df_e, family = binomial(link = 'probit'))
r4be <- glm(tb, data = df_e, family = binomial(link = 'probit'))
r4ce <- glm(tc, data = df_e, family = binomial(link = 'probit'))
df_e %<>% mutate(prop_a = predict(r4ae, df_e, type='response'),
                 prop_b = predict(r4be, df_e, type='response'),
                 prop_c = predict(r4ce, df_e, type='response'))

r4ap <- glm(ta, data = df_p, family = binomial(link = 'probit'))
r4bp <- glm(tb, data = df_p, family = binomial(link = 'probit'))
r4cp <- glm(tc, data = df_p, family = binomial(link = 'probit'))
df_p %<>% mutate(prop_a = predict(r4ap, df_p, type='response'),
                 prop_b = predict(r4bp, df_p, type='response'),
                 prop_c = predict(r4cp, df_p, type='response'))

# trim df_p for interior predicted propensities
df_p %<>% filter(prop_a >= .001, prop_b >= .001, prop_c >= .001,
                 prop_a <= .999, prop_b <= .999, prop_c <= .999)

# calculate probability of treatment
n_p <- nrow(df_p)

phat_e = sum(df_e$t==1)/n_e
phat_p = sum(df_p$t==1)/n_p

# build components for ATE
df_e %<>% mutate(left_a = t*re78/prop_a, right_a = (1-t)*re78/(1-prop_a),
                 left_b = t*re78/prop_b, right_b = (1-t)*re78/(1-prop_b),
                 left_c = t*re78/prop_c, right_c = (1-t)*re78/(1-prop_c))
df_p %<>% mutate(left_a = t*re78/prop_a, right_a = (1-t)*re78/(1-prop_a),
                 left_b = t*re78/prop_b, right_b = (1-t)*re78/(1-prop_b),
                 left_c = t*re78/prop_c, right_c = (1-t)*re78/(1-prop_c))

# calculate ATEs
ate[8,1]  <- mean(df_e$left_a) - mean(df_e$right_a)
ate[8,5]  <- mean(df_p$left_a) - mean(df_p$right_a)
```

```
ate[9,1]  <- mean(df_e$left_b) - mean(df_e$right_b)
ate[9,5]  <- mean(df_p$left_b) - mean(df_p$right_b)
ate[10,1] <- mean(df_e$left_c) - mean(df_e$right_c)
ate[10,5] <- mean(df_p$left_c) - mean(df_p$right_c)

# build components for ATT
df_e %<>% mutate(left_a = t*re78/phat_e, right_a = (prop_a/phat_e)*(1-t)*re78/(1-prop_a),
                 left_b = t*re78/phat_e, right_b = (prop_b/phat_e)*(1-t)*re78/(1-prop_b),
                 left_c = t*re78/phat_e, right_c = (prop_c/phat_e)*(1-t)*re78/(1-prop_c))
df_p %<>% mutate(left_a = t*re78/phat_p, right_a = (prop_a/phat_p)*(1-t)*re78/(1-prop_a),
                 left_b = t*re78/phat_p, right_b = (prop_b/phat_p)*(1-t)*re78/(1-prop_b),
                 left_c = t*re78/phat_p, right_c = (prop_c/phat_p)*(1-t)*re78/(1-prop_c))

# calculate ATTs
att[8,1]  <- mean(df_e$left_a) - mean(df_e$right_a)
att[8,5]  <- mean(df_p$left_a) - mean(df_p$right_a)
att[9,1]  <- mean(df_e$left_b) - mean(df_e$right_b)
att[9,5]  <- mean(df_p$left_b) - mean(df_p$right_b)
att[10,1] <- mean(df_e$left_c) - mean(df_e$right_c)
att[10,5] <- mean(df_p$left_c) - mean(df_p$right_c)

################################################################################
# Q2.4 and 2.5: Inverse Probability Weighting (rows 8-10) using CausalGAM
# thanks to Ani Yadav for showing me this package

# redefine for use here
df_e <- df %>% filter(treat != 2) %>% as.data.frame
df_p <- df %>% filter(treat != 0) %>% as.data.frame

# experimental data, three covar sets
cgam_a_e <- estimate.ATE(pscore.formula = ta, pscore.family = binomial,
                         outcome.formula.t = ya, outcome.formula.c = ya,
                         outcome.family = gaussian,
                         treatment.var = 't', data = df_e,
                         divby0.action = 'truncate', divby0.tol = 0.001,
                         var.gam.plot = F, nboot = 0)

cgam_b_e <- estimate.ATE(pscore.formula = tb, pscore.family = binomial,
                         outcome.formula.t = yb, outcome.formula.c = yb,
                         outcome.family = gaussian,
                         treatment.var = 't', data = df_e,
                         divby0.action = 'truncate', divby0.tol = 0.001,
                         var.gam.plot = F, nboot = 0)

cgam_c_e <- estimate.ATE(pscore.formula = tc, pscore.family = binomial,
                         outcome.formula.t = yc, outcome.formula.c = yc,
                         outcome.family = gaussian,
                         treatment.var = 't', data = df_e,
                         divby0.action = 'truncate', divby0.tol = 0.001,
                         var.gam.plot = F, nboot = 0)

# PSID data, three covar sets
# increase the tolerance here (else it fails)
cgam_a_p <- estimate.ATE(pscore.formula = ta, pscore.family = binomial,
```

```
                            outcome.formula.t = ya, outcome.formula.c = ya,
                            outcome.family = gaussian,
                            treatment.var = 't', data = df_p,
                            divby0.action = 'truncate', divby0.tol = 0.03,
                            var.gam.plot = F, nboot = 0)

cgam_b_p <- estimate.ATE(pscore.formula = tb, pscore.family = binomial,
                            outcome.formula.t = yb, outcome.formula.c = yb,
                            outcome.family = gaussian,
                            treatment.var = 't', data = df_p,
                            divby0.action = 'truncate', divby0.tol = 0.03,
                            var.gam.plot = F, nboot = 0)

cgam_c_p <- estimate.ATE(pscore.formula = tc, pscore.family = binomial,
                            outcome.formula.t = yc, outcome.formula.c = yc,
                            outcome.family = gaussian,
                            treatment.var = 't', data = df_p,
                            divby0.action = 'truncate', divby0.tol = 0.06,
                            var.gam.plot = F, nboot = 0)

# save ATEs
ate[8,1]  <- cgam_a_e$ATE.IPW.hat
ate[8,5]  <- cgam_a_p$ATE.IPW.hat
ate[9,1]  <- cgam_b_e$ATE.IPW.hat
ate[9,5]  <- cgam_b_p$ATE.IPW.hat
ate[10,1] <- cgam_c_e$ATE.IPW.hat
ate[10,5] <- cgam_c_p$ATE.IPW.hat

# save SEs
ate[8,2]  <- cgam_a_e$ATE.IPW.asymp.SE
ate[8,6]  <- cgam_a_p$ATE.IPW.asymp.SE
ate[9,2]  <- cgam_b_e$ATE.IPW.asymp.SE
ate[9,6]  <- cgam_b_p$ATE.IPW.asymp.SE
ate[10,2] <- cgam_c_e$ATE.IPW.asymp.SE
ate[10,6] <- cgam_c_p$ATE.IPW.asymp.SE

################################################################################
# Q2.5: Doubly Robust (using CausalGAM from above) (rows 11-13)

# save ATEs
ate[11,1] <- cgam_a_e$ATE.AIPW.hat
ate[11,5] <- cgam_a_p$ATE.AIPW.hat
ate[12,1] <- cgam_b_e$ATE.AIPW.hat
ate[12,5] <- cgam_b_p$ATE.AIPW.hat
ate[13,1] <- cgam_c_e$ATE.AIPW.hat
ate[13,5] <- cgam_c_p$ATE.AIPW.hat

# save SEs
ate[11,2] <- cgam_a_e$ATE.AIPW.asymp.SE
ate[11,6] <- cgam_a_p$ATE.AIPW.asymp.SE
ate[12,2] <- cgam_b_e$ATE.AIPW.asymp.SE
ate[12,6] <- cgam_b_p$ATE.AIPW.asymp.SE
ate[13,2] <- cgam_c_e$ATE.AIPW.asymp.SE
ate[13,6] <- cgam_c_p$ATE.AIPW.asymp.SE
```

```
# missing ATTs - doesn't seem CausalGAM can do ATT

################################################################################
# Q2.6: Nearest Neighbor Matching (rows 14-16)

# useful sources:
# https://cran.r-project.org/web/packages/MatchIt/vignettes/matchit.pdf

# redefine dataframes
df_e <- df %>% filter(treat != 2)
df_p <- df %>% filter(treat != 0)

# see https://r.iq.harvard.edu/docs/matchit/2.4-20/Examples2.html
# function to calculate ate and att using NN methods
# can't figure out how to pass a dataframe to the matchit command
# tried a million ways but none working, so writing two functions...
nnResultsE <- function(tmodel, ymodel, est){
    # using MatchIt
  reg_match <- matchit(tmodel, df_e, method = 'nearest')

  df_t <- match.data(reg_match, 'treat')
  df_c <- match.data(reg_match, 'control')

  # predict values and treatment effects
  reg_t       <- lm(ymodel, df_t)
  df_c$y_pred <- predict(reg_t, df_c)

  df_c %<>% mutate(teffect = y_pred-re78)

  reg_c <- lm(ymodel, df_c)
  df_t$y_pred  <- predict(reg_c, df_t)

  df_t %<>% mutate(teffect = re78-y_pred)

  # stack and predict treatment effects, etc.
  comb <- bind_rows(df_c, df_t)

  ate <- mean(comb$teffect)
  att <- mean(comb$teffect[comb$t==1])

  n_att = nrow(df_t)
  n_ate = n_att*2

  ate_se <- sd(comb$teffect)/sqrt(n_ate)
  att_se <- sd(comb$teffect[comb$t==1])/sqrt(att)

  out_ate = matrix(c(ate, ate_se, NA, NA),1,4)
  out_att = matrix(c(att, att_se, NA, NA),1,4)

  if(est == 'ate'){
    return(out_ate)
  } else {
    return(out_att)
```

```
  }
}

nnResultsP <- function(tmodel, ymodel, est){
  # using MatchIt
  reg_match <- matchit(tmodel, df_p, method = 'nearest')

  df_t <- match.data(reg_match, 'treat')
  df_c <- match.data(reg_match, 'control')

  # predict values and treatment effects
  reg_t       <- lm(ymodel, df_t)
  df_c$y_pred <- predict(reg_t, df_c)

  df_c %<>% mutate(teffect = y_pred-re78)

  reg_c <- lm(ymodel, df_c)
  df_t$y_pred  <- predict(reg_c, df_t)

  df_t %<>% mutate(teffect = re78-y_pred)

  # stack and predict treatment effects, etc.
  comb <- bind_rows(df_c, df_t)

  ate <- mean(comb$teffect)
  att <- mean(comb$teffect[comb$t==1])

  n_att = nrow(df_t)
  n_ate = n_att*2

  ate_se <- sd(comb$teffect)/sqrt(n_ate)
  att_se <- sd(comb$teffect[comb$t==1])/sqrt(att)

  out_ate = matrix(c(ate, ate_se, NA, NA),1,4)
  out_att = matrix(c(att, att_se, NA, NA),1,4)

  if(est == 'ate'){
    return(out_ate)
  } else {
    return(out_att)
  }
}

# save results
ate[14,1:4] <- nnResultsE(ta, ya, 'ate')
ate[14,5:8] <- nnResultsP(ta, ya, 'ate')

ate[15,1:4] <- nnResultsE(tb, yb, 'ate')
ate[15,5:8] <- nnResultsP(tb, yb, 'ate')

ate[16,1:4] <- nnResultsE(tc, yc, 'ate')
ate[16,5:8] <- nnResultsP(tc, yc, 'ate')

att[14,1:4] <- nnResultsE(ta, ya, 'att')
```

```
att[14,5:8] <- nnResultsP(ta, ya, 'att')

att[15,1:4] <- nnResultsE(tb, yb, 'att')
att[15,5:8] <- nnResultsP(tb, yb, 'att')

att[16,1:4] <- nnResultsE(tc, yc, 'att')
att[16,5:8] <- nnResultsP(tc, yc, 'att')

#############################################################################################
# Q2.7: Propensity Score Matching (rows 17-19)

# using same code as above, with probit for distance (propensity score)
# https://www.r-bloggers.com/using-the-r-matchit-package-for-propensity-score-analysis/

# redefine dataframes
df_e <- df %>% filter(treat != 2)
df_p <- df %>% filter(treat != 0)

# only difference here is the matchit command uses probit
psResultsE <- function(tmodel, ymodel, est){
  # using MatchIt
  reg_match <- matchit(tmodel, df_e, method = 'nearest', distance = 'probit')

  df_t <- match.data(reg_match, 'treat')
  df_c <- match.data(reg_match, 'control')

  # predict values and treatment effects
  reg_t       <- lm(ymodel, df_t)
  df_c$y_pred <- predict(reg_t, df_c)

  df_c %<>% mutate(teffect = y_pred-re78)

  reg_c <- lm(ymodel, df_c)
  df_t$y_pred  <- predict(reg_c, df_t)

  df_t %<>% mutate(teffect = re78-y_pred)

  # stack and predict treatment effects, etc.
  comb <- bind_rows(df_c, df_t)

  ate <- mean(comb$teffect)
  att <- mean(comb$teffect[comb$t==1])

  n_att = nrow(df_t)
  n_ate = n_att*2

  ate_se <- sd(comb$teffect)/sqrt(n_ate)
  att_se <- sd(comb$teffect[comb$t==1])/sqrt(att)

  out_ate = matrix(c(ate, ate_se, NA, NA),1,4)
  out_att = matrix(c(att, att_se, NA, NA),1,4)

  if(est == 'ate'){
    return(out_ate)
```

```
  } else {
    return(out_att)
  }
}

psResultsP <- function(tmodel, ymodel, est){
  # using MatchIt
  reg_match <- matchit(tmodel, df_p, method = 'nearest', distance = 'probit')

  df_t <- match.data(reg_match, 'treat')
  df_c <- match.data(reg_match, 'control')

  # predict values and treatment effects
  reg_t       <- lm(ymodel, df_t)
  df_c$y_pred <- predict(reg_t, df_c)

  df_c %<>% mutate(teffect = y_pred-re78)

  reg_c <- lm(ymodel, df_c)
  df_t$y_pred  <- predict(reg_c, df_t)

  df_t %<>% mutate(teffect = re78-y_pred)

  # stack and predict treatment effects, etc.
  comb <- bind_rows(df_c, df_t)

  ate <- mean(comb$teffect)
  att <- mean(comb$teffect[comb$t==1])

  n_att = nrow(df_t)
  n_ate = n_att*2

  ate_se <- sd(comb$teffect)/sqrt(n_ate)
  att_se <- sd(comb$teffect[comb$t==1])/sqrt(att)

  out_ate = matrix(c(ate, ate_se, NA, NA),1,4)
  out_att = matrix(c(att, att_se, NA, NA),1,4)

  if(est == 'ate'){
    return(out_ate)
  } else {
    return(out_att)
  }
}

# save results
ate[17,1:4] <- psResultsE(ta, ya, 'ate')
ate[17,5:8] <- psResultsP(ta, ya, 'ate')

ate[18,1:4] <- psResultsE(tb, yb, 'ate')
ate[18,5:8] <- psResultsP(tb, yb, 'ate')

ate[19,1:4] <- psResultsE(tc, yc, 'ate')
ate[19,5:8] <- psResultsP(tc, yc, 'ate')
```

```
att[17,1:4] <- psResultsE(ta, ya, 'att')
att[17,5:8] <- psResultsP(ta, ya, 'att')

att[18,1:4] <- psResultsE(tb, yb, 'att')
att[18,5:8] <- psResultsP(tb, yb, 'att')

att[19,1:4] <- psResultsE(tc, yc, 'att')
att[19,5:8] <- psResultsP(tc, yc, 'att')

################################################################################
# Q2 final steps

# confidence intervals
ate %<>% mutate(e_CI_l = e_tau - 1.96*e_se,
                e_CI_h = e_tau + 1.96*e_se,
                p_CI_l = p_tau - 1.96*p_se,
                p_CI_h = p_tau + 1.96*p_se)

att %<>% mutate(e_CI_l = e_tau - 1.96*e_se,
                e_CI_h = e_tau + 1.96*e_se,
                p_CI_l = p_tau - 1.96*p_se,
                p_CI_h = p_tau + 1.96*p_se)

rownames(ate) <- c(
  'Diff-in-Means',
  'OLS: a', 'OLS: b', 'OLS: c',
  'RI: a', 'RI: b', 'RI: c',
  'IPW: a', 'IPW: b', 'IPW: c',
  'DR: a', 'DR: b', 'DR: c',
  'NN: a', 'NN: b', 'NN: c',
  'Propensity: a', 'Propensity: b', 'Propensity: c'
)

rownames(att) <- rownames(ate)

# write to LaTeX
xtable(ate)
xtable(att)

################################################################################
### Question 3: Post-model Selection Inference
################################################################################
# Q3 setup
rm(list = ls())
gc()

################################################################################
# Q3.1: Summary Statistics and Kernel Density Plots

# replications
M <- 1000

# sample size
```

```
n <- 50

# define data generating process
dgp <- function(n){
  x   <- rnorm(n,0,1)
  z   <- .85*x + sqrt(1-.85)*rnorm(n,0,1)
  eps <- rnorm(n,0,1)

  y = 1 + .5*x + z + eps

  out = data.frame(y=y,x=x,z=z)

  return(out)
}

# empty matrices to store estimates and indicator of coverage
est <- matrix(NA, nrow=M, ncol = 3)
cov <- matrix(NA, nrow=M, ncol = 3)

# replicate M times and save estimates
set.seed(22)
for(i in 1:M){
  # generate sample
  df <- dgp(n)

  # run 'long' regression: Equation (2)
  long <- lm(y ~ x + z, data = df)

  # extract first estimate
  beta_hat <- long$coefficients['x']
  est[i,1] <- beta_hat

  # get SE to calculate confidence interval and check coverage
  se_hat <- sqrt(vcovHC(long, 'HC1')['x','x'])
  lb_hat <- beta_hat - 1.96*se_hat
  ub_hat <- beta_hat + 1.96*se_hat
  cov[i,1] <- lb_hat <= .5 & ub_hat >= .5

  # save gamma over SE gamma (using robust std errors)
  gamma_hat <- long$coefficients['z']
  gamma_se  <- sqrt(vcovHC(long, 'HC1')['z','z'])
  tstat <- gamma_hat/gamma_se

  # run 'short' regression: Equation (3)
  short <- lm(y ~ x, data = df)

  # extract second estimate
  beta_tilde <- short$coefficients['x']
  est[i,2]   <- beta_tilde

  # get SE to calculate confidence interval and check coverage
  se_tilde <- sqrt(vcovHC(short, 'HC1')['x','x'])
  lb_tilde <- beta_tilde - 1.96*se_hat
  ub_tilde <- beta_tilde + 1.96*se_hat
```

```
  cov[i,2] <- lb_tilde <= .5 & ub_tilde >= .5

  # save third estimate (beta_check) based on t-stat on gamma
  est[i,3] <- ifelse(tstat >= 1.96, beta_hat, beta_tilde)
  cov[i,3] <- ifelse(tstat >= 1.96, cov[i,1], cov[i,2])
}

# summary statistic of distribution of each of the estimators
tab <- matrix(NA,nrow=3,ncol=4)
for(i in 1:3){
  tab[i,1] <- min(est[,i])
  tab[i,2] <- mean(est[,i])
  tab[i,3] <- median(est[,i])
  tab[i,4] <- max(est[,i])
}

# add labels
colnames(tab) <- c('min', 'mean', 'median', 'max')
rownames(tab) <- c('beta_hat', 'beta_tilde', 'beta_check')

# write to LaTeX
xtable(tab %>% round(3))

# kernel density plots
est %<>% as.data.frame
names(est) <- c('beta_hat', 'beta_tilde', 'beta_check')
est <- gather(est, key = 'Estimator', value = 'x')

# plot kernel densities
p <- ggplot(est, aes(x = x, group = Estimator, color = Estimator)) +
  geom_density(kernel = 'epanechnikov', size = 1) +
  theme_minimal() +
  labs(x = '', y = 'Density')
p

# save for LaTeX
ggsave('q3_r.png')

##############################################################################
# Q3.2: Empirical Coverage Rates

# we know beta_0 is actually .5, how often do we cover that?
# in the loop above, we calculated coverage for each replication
# now, summarize for each estimator:

cov_hat   <- mean(cov[,1])
cov_tilde <- mean(cov[,2])
cov_check <- mean(cov[,3])

##############################################################################
```

B Stata Code

```
*******************************************************************************
* Author: Paul R. Organ
* Purpose: ECON 675, PS4
* Last Update: Oct 30, 2018
*******************************************************************************
clear all
set more off
capture log close

cd "C:\Users\prorgan\Box\Classes\Econ 675\Problem Sets\PS4"
log using ps4.log, replace

*******************************************************************************
*** Question 2: Estimating Average Treatment Effects
*******************************************************************************
* Q2 setup

* load data
import delimited "LaLonde_all.csv"

* add treatment variable in expanded dataset
gen t = 0
replace t = 1 if treat == 1

* add other variables we will use in the analysis
gen logre74 = log(re74 + 1)
gen logre75 = log(re75 + 1)
gen age2 = age^2
gen educ2 = educ^2
gen age3 = age^3
gen bXu74 = black * u74
gen eXre74 = educ * logre74

* define vars for model specifications
local a = "age educ black hisp married nodegr logre74 logre75"
local b = "age educ black hisp married nodegr logre74 logre75
           age2 educ2 u74 u75"
local c = "age educ black hisp married nodegr logre74 logre75
           age2 educ2 u74 u75 age3 bXu74 eXre74"

* empty tables to fill with results (1+6*3)
matrix ate = J(19,8,.)
matrix att = J(19,8,.)

*******************************************************************************
* Q2.1: Difference-in-Means (row 1)

* experimental data
qui reg re78 t if (treat != 2), vce(hc2)

* pull out results, and assign them to our empty table for ATE
matrix out = r(table)
```

```
matrix ate[1,1] = out["b","t"]
matrix ate[1,2] = out["se","t"]
matrix ate[1,3] = out["ll","t"]
matrix ate[1,4] = out["ul","t"]

* PSID data
qui reg re78 t if (treat != 0), vce(hc2)

* pull out results, and assign them to our empty table for ATE
matrix out = r(table)
matrix ate[1,5] = out["b","t"]
matrix ate[1,6] = out["se","t"]
matrix ate[1,7] = out["ll","t"]
matrix ate[1,8] = out["ul","t"]

* same for ATT
forvalues j = 1/8{
        matrix att[1,`j'] = ate[1,`j']
}

*******************************************************************************
* Q2.2: Linear Least-Squares (rows 2-4)

* experimental data, three ways
qui reg re78 t `a' if (treat != 2), vce(r)
matrix out = r(table)
matrix ate[2,1] = out["b","t"]
matrix ate[2,2] = out["se","t"]
matrix ate[2,3] = out["ll","t"]
matrix ate[2,4] = out["ul","t"]

qui reg re78 t `b' if (treat != 2), vce(r)
matrix out = r(table)
matrix ate[3,1] = out["b","t"]
matrix ate[3,2] = out["se","t"]
matrix ate[3,3] = out["ll","t"]
matrix ate[3,4] = out["ul","t"]

qui reg re78 t `c' if (treat != 2), vce(r)
matrix out = r(table)
matrix ate[4,1] = out["b","t"]
matrix ate[4,2] = out["se","t"]
matrix ate[4,3] = out["ll","t"]
matrix ate[4,4] = out["ul","t"]

* PSID data
qui reg re78 t `a' if (treat != 0), vce(r)
matrix out = r(table)
matrix ate[2,5] = out["b","t"]
matrix ate[2,6] = out["se","t"]
matrix ate[2,7] = out["ll","t"]
matrix ate[2,8] = out["ul","t"]

qui reg re78 t `b' if (treat != 0), vce(r)
```

```
matrix out = r(table)
matrix ate[3,5] = out["b","t"]
matrix ate[3,6] = out["se","t"]
matrix ate[3,7] = out["ll","t"]
matrix ate[3,8] = out["ul","t"]

qui reg re78 t `c' if (treat != 0), vce(r)
matrix out = r(table)
matrix ate[4,5] = out["b","t"]
matrix ate[4,6] = out["se","t"]
matrix ate[4,7] = out["ll","t"]
matrix ate[4,8] = out["ul","t"]

* same for ATT
forvalues i = 2/4{
        forvalues j = 1/8{
                matrix att[`i',`j'] = ate[`i',`j']
        }
}

*******************************************************************************
* Q2.3: Regression Imputation (rows 5-7)

* ate, experimental data, three ways
qui teffects ra (re78 `a') (t) if (treat != 2), ate
matrix out = r(table)
matrix ate[5,1] = out["b",1]
matrix ate[5,2] = out["se",1]
matrix ate[5,3] = out["ll",1]
matrix ate[5,4] = out["ul",1]

qui teffects ra (re78 `b') (t) if (treat != 2), ate
matrix out = r(table)
matrix ate[6,1] = out["b",1]
matrix ate[6,2] = out["se",1]
matrix ate[6,3] = out["ll",1]
matrix ate[6,4] = out["ul",1]

qui teffects ra (re78 `c') (t) if (treat != 2), ate
matrix out = r(table)
matrix ate[7,1] = out["b",1]
matrix ate[7,2] = out["se",1]
matrix ate[7,3] = out["ll",1]
matrix ate[7,4] = out["ul",1]

* att, experimental data, three ways
qui teffects ra (re78 `a') (t) if (treat != 2), atet
matrix out = r(table)
matrix att[5,1] = out["b",1]
matrix att[5,2] = out["se",1]
matrix att[5,3] = out["ll",1]
matrix att[5,4] = out["ul",1]

qui teffects ra (re78 `b') (t) if (treat != 2), atet
```

```
matrix out = r(table)
matrix att[6,1] = out["b",1]
matrix att[6,2] = out["se",1]
matrix att[6,3] = out["ll",1]
matrix att[6,4] = out["ul",1]

qui teffects ra (re78 `c') (t) if (treat != 2), atet
matrix out = r(table)
matrix att[7,1] = out["b",1]
matrix att[7,2] = out["se",1]
matrix att[7,3] = out["ll",1]
matrix att[7,4] = out["ul",1]

* ate, PSID data, three ways
qui teffects ra (re78 `a') (t) if (treat != 0), ate
matrix out = r(table)
matrix ate[5,5] = out["b",1]
matrix ate[5,6] = out["se",1]
matrix ate[5,7] = out["ll",1]
matrix ate[5,8] = out["ul",1]

qui teffects ra (re78 `b') (t) if (treat != 0), ate
matrix out = r(table)
matrix ate[6,5] = out["b",1]
matrix ate[6,6] = out["se",1]
matrix ate[6,7] = out["ll",1]
matrix ate[6,8] = out["ul",1]

qui teffects ra (re78 `c') (t) if (treat != 0), ate
matrix out = r(table)
matrix ate[7,5] = out["b",1]
matrix ate[7,6] = out["se",1]
matrix ate[7,7] = out["ll",1]
matrix ate[7,8] = out["ul",1]

* att, PSID data, three ways
qui teffects ra (re78 `a') (t) if (treat != 0), atet
matrix out = r(table)
matrix att[5,5] = out["b",1]
matrix att[5,6] = out["se",1]
matrix att[5,7] = out["ll",1]
matrix att[5,8] = out["ul",1]

qui teffects ra (re78 `b') (t) if (treat != 0), atet
matrix out = r(table)
matrix att[6,5] = out["b",1]
matrix att[6,6] = out["se",1]
matrix att[6,7] = out["ll",1]
matrix att[6,8] = out["ul",1]

qui teffects ra (re78 `c') (t) if (treat != 0), atet
matrix out = r(table)
matrix att[7,5] = out["b",1]
matrix att[7,6] = out["se",1]
```

```
matrix att[7,7] = out["ll",1]
matrix att[7,8] = out["ul",1]

*******************************************************************************
* Q2.4: Inverse Probability Weighting (rows 8-10)

* ate, experimental data, three ways
qui teffects ipw (re78) (t `a', probit) if (treat != 2), ate iterate(50)
matrix out = r(table)
matrix ate[8,1] = out["b",1]
matrix ate[8,2] = out["se",1]
matrix ate[8,3] = out["ll",1]
matrix ate[8,4] = out["ul",1]

qui teffects ipw (re78) (t `b', probit) if (treat != 2), ate iterate(50)
matrix out = r(table)
matrix ate[9,1] = out["b",1]
matrix ate[9,2] = out["se",1]
matrix ate[9,3] = out["ll",1]
matrix ate[9,4] = out["ul",1]

qui teffects ipw (re78) (t `c', probit) if (treat != 2), ate iterate(50)
matrix out = r(table)
matrix ate[10,1] = out["b",1]
matrix ate[10,2] = out["se",1]
matrix ate[10,3] = out["ll",1]
matrix ate[10,4] = out["ul",1]

* att, experimental data, three ways
qui teffects ipw (re78) (t `a', probit) if (treat != 2), atet iterate(50)
matrix out = r(table)
matrix att[8,1] = out["b",1]
matrix att[8,2] = out["se",1]
matrix att[8,3] = out["ll",1]
matrix att[8,4] = out["ul",1]

qui teffects ipw (re78) (t `b', probit) if (treat != 2), atet iterate(50)
matrix out = r(table)
matrix att[9,1] = out["b",1]
matrix att[9,2] = out["se",1]
matrix att[9,3] = out["ll",1]
matrix att[9,4] = out["ul",1]

qui teffects ipw (re78) (t `c', probit) if (treat != 2), atet iterate(50)
matrix out = r(table)
matrix att[10,1] = out["b",1]
matrix att[10,2] = out["se",1]
matrix att[10,3] = out["ll",1]
matrix att[10,4] = out["ul",1]

* predicted propensity scores for PSID data are too close to 0 or 1
* so we need to predict, then only use interior data
* note new if condition (keep if treated, or PSID and interior prop score)
```

```
* three ways, PSID data, ate and att
qui probit t `a' if (treat != 0)
capture: drop prop
predict prop
qui teffects ipw (re78) (t `a') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate iterate(50)
matrix out = r(table)
matrix ate[8,5] = out["b",1]
matrix ate[8,6] = out["se",1]
matrix ate[8,7] = out["ll",1]
matrix ate[8,8] = out["ul",1]

qui teffects ipw (re78) (t `a') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet iterate(50)
matrix out = r(table)
matrix att[8,5] = out["b",1]
matrix att[8,6] = out["se",1]
matrix att[8,7] = out["ll",1]
matrix att[8,8] = out["ul",1]

qui probit t `b' if (treat != 0)
capture: drop prop
predict prop
qui teffects ipw (re78) (t `b') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate iterate(50)
matrix out = r(table)
matrix ate[9,5] = out["b",1]
matrix ate[9,6] = out["se",1]
matrix ate[9,7] = out["ll",1]
matrix ate[9,8] = out["ul",1]

qui teffects ipw (re78) (t `b') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet iterate(50)
matrix out = r(table)
matrix att[9,5] = out["b",1]
matrix att[9,6] = out["se",1]
matrix att[9,7] = out["ll",1]
matrix att[9,8] = out["ul",1]

qui probit t `c' if (treat != 0)
capture: drop prop
predict prop
qui teffects ipw (re78) (t `c') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate iterate(50)
matrix out = r(table)
matrix ate[10,5] = out["b",1]
matrix ate[10,6] = out["se",1]
matrix ate[10,7] = out["ll",1]
matrix ate[10,8] = out["ul",1]

qui teffects ipw (re78) (t `c') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet iterate(50)
matrix out = r(table)
matrix att[10,5] = out["b",1]
```

```
matrix att[10,6] = out["se",1]
matrix att[10,7] = out["ll",1]
matrix att[10,8] = out["ul",1]

*******************************************************************************
* Q2.5: Doubly Robust (rows 11-13)

* ate, experimental data, three ways
qui teffects ipwra (re78 `a') (t `a', probit) if (treat != 2), ate iterate(50)
matrix out = r(table)
matrix ate[11,1] = out["b",1]
matrix ate[11,2] = out["se",1]
matrix ate[11,3] = out["ll",1]
matrix ate[11,4] = out["ul",1]

qui teffects ipwra (re78 `b') (t `b', probit) if (treat != 2), ate iterate(50)
matrix out = r(table)
matrix ate[12,1] = out["b",1]
matrix ate[12,2] = out["se",1]
matrix ate[12,3] = out["ll",1]
matrix ate[12,4] = out["ul",1]

qui teffects ipwra (re78 `c') (t `c', probit) if (treat != 2), ate iterate(50)
matrix out = r(table)
matrix ate[13,1] = out["b",1]
matrix ate[13,2] = out["se",1]
matrix ate[13,3] = out["ll",1]
matrix ate[13,4] = out["ul",1]

* att, experimental data, three ways
qui teffects ipwra (re78 `a') (t `a', probit) if (treat != 2), atet iterate(50)
matrix out = r(table)
matrix att[11,1] = out["b",1]
matrix att[11,2] = out["se",1]
matrix att[11,3] = out["ll",1]
matrix att[11,4] = out["ul",1]

qui teffects ipwra (re78 `b') (t `b', probit) if (treat != 2), atet iterate(50)
matrix out = r(table)
matrix att[12,1] = out["b",1]
matrix att[12,2] = out["se",1]
matrix att[12,3] = out["ll",1]
matrix att[12,4] = out["ul",1]

qui teffects ipwra (re78 `c') (t `c', probit) if (treat != 2), atet iterate(50)
matrix out = r(table)
matrix att[13,1] = out["b",1]
matrix att[13,2] = out["se",1]
matrix att[13,3] = out["ll",1]
matrix att[13,4] = out["ul",1]

* predicted propensity scores for PSID data are too close to 0 or 1
* so we need to predict, then only use interior data
```

```
* three ways, PSID data, ate and att
qui probit t `a' if (treat != 0)
capture: drop prop
predict prop
qui teffects ipwra (re78 `a') (t `a') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate iterate(50)
matrix out = r(table)
matrix ate[11,5] = out["b",1]
matrix ate[11,6] = out["se",1]
matrix ate[11,7] = out["ll",1]
matrix ate[11,8] = out["ul",1]

qui teffects ipwra (re78 `a') (t `a') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet iterate(50)
matrix out = r(table)
matrix att[11,5] = out["b",1]
matrix att[11,6] = out["se",1]
matrix att[11,7] = out["ll",1]
matrix att[11,8] = out["ul",1]

qui probit t `b' if (treat != 0)
capture: drop prop
predict prop
qui teffects ipwra (re78 `b') (t `b') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate iterate(50)
matrix out = r(table)
matrix ate[12,5] = out["b",1]
matrix ate[12,6] = out["se",1]
matrix ate[12,7] = out["ll",1]
matrix ate[12,8] = out["ul",1]

qui teffects ipwra (re78 `b') (t `b') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet iterate(50)
matrix out = r(table)
matrix att[12,5] = out["b",1]
matrix att[12,6] = out["se",1]
matrix att[12,7] = out["ll",1]
matrix att[12,8] = out["ul",1]

qui probit t `c' if (treat != 0)
capture: drop prop
predict prop
qui teffects ipwra (re78 `c') (t `c') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate iterate(50)
matrix out = r(table)
matrix ate[13,5] = out["b",1]
matrix ate[13,6] = out["se",1]
matrix ate[13,7] = out["ll",1]
matrix ate[13,8] = out["ul",1]

qui teffects ipwra (re78 `c') (t `c') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet iterate(50)
matrix out = r(table)
matrix att[13,5] = out["b",1]
```

```
matrix att[13,6] = out["se",1]
matrix att[13,7] = out["ll",1]
matrix att[13,8] = out["ul",1]

*******************************************************************************
* Q2.6: Nearest Neighbor Matching (rows 14-16)

* ate, experimental data, three ways
qui teffects nnmatch (re78 `a') (t) if (treat != 2), ate nneighbor(1) metric(maha)
matrix out = r(table)
matrix ate[14,1] = out["b",1]
matrix ate[14,2] = out["se",1]
matrix ate[14,3] = out["ll",1]
matrix ate[14,4] = out["ul",1]

qui teffects nnmatch (re78 `b') (t) if (treat != 2), ate nneighbor(1) metric(maha)
matrix out = r(table)
matrix ate[15,1] = out["b",1]
matrix ate[15,2] = out["se",1]
matrix ate[15,3] = out["ll",1]
matrix ate[15,4] = out["ul",1]

qui teffects nnmatch (re78 `c') (t) if (treat != 2), ate nneighbor(1) metric(maha)
matrix out = r(table)
matrix ate[16,1] = out["b",1]
matrix ate[16,2] = out["se",1]
matrix ate[16,3] = out["ll",1]
matrix ate[16,4] = out["ul",1]

* att, experimental data, three ways
qui teffects nnmatch (re78 `a') (t) if (treat != 2), atet nneighbor(1) metric(maha)
matrix out = r(table)
matrix att[14,1] = out["b",1]
matrix att[14,2] = out["se",1]
matrix att[14,3] = out["ll",1]
matrix att[14,4] = out["ul",1]

qui teffects nnmatch (re78 `b') (t) if (treat != 2), atet nneighbor(1) metric(maha)
matrix out = r(table)
matrix att[15,1] = out["b",1]
matrix att[15,2] = out["se",1]
matrix att[15,3] = out["ll",1]
matrix att[15,4] = out["ul",1]

qui teffects nnmatch (re78 `c') (t) if (treat != 2), atet nneighbor(1) metric(maha)
matrix out = r(table)
matrix att[16,1] = out["b",1]
matrix att[16,2] = out["se",1]
matrix att[16,3] = out["ll",1]
matrix att[16,4] = out["ul",1]

* ate, PSID data, three ways
qui teffects nnmatch (re78 `a') (t) if (treat != 0), ate nneighbor(1) metric(maha)
matrix out = r(table)
```

```
matrix ate[14,5] = out["b",1]
matrix ate[14,6] = out["se",1]
matrix ate[14,7] = out["ll",1]
matrix ate[14,8] = out["ul",1]

qui teffects nnmatch (re78 `b') (t) if (treat != 0), ate nneighbor(1) metric(maha)
matrix out = r(table)
matrix ate[15,5] = out["b",1]
matrix ate[15,6] = out["se",1]
matrix ate[15,7] = out["ll",1]
matrix ate[15,8] = out["ul",1]

qui teffects nnmatch (re78 `c') (t) if (treat != 0), ate nneighbor(1) metric(maha)
matrix out = r(table)
matrix ate[16,5] = out["b",1]
matrix ate[16,6] = out["se",1]
matrix ate[16,7] = out["ll",1]
matrix ate[16,8] = out["ul",1]

* att, PSID data, three ways
qui teffects nnmatch (re78 `a') (t) if (treat != 0), atet nneighbor(1) metric(maha)
matrix out = r(table)
matrix att[14,5] = out["b",1]
matrix att[14,6] = out["se",1]
matrix att[14,7] = out["ll",1]
matrix att[14,8] = out["ul",1]

qui teffects nnmatch (re78 `b') (t) if (treat != 0), atet nneighbor(1) metric(maha)
matrix out = r(table)
matrix att[15,5] = out["b",1]
matrix att[15,6] = out["se",1]
matrix att[15,7] = out["ll",1]
matrix att[15,8] = out["ul",1]

qui teffects nnmatch (re78 `c') (t) if (treat != 0), atet nneighbor(1) metric(maha)
matrix out = r(table)
matrix att[16,5] = out["b",1]
matrix att[16,6] = out["se",1]
matrix att[16,7] = out["ll",1]
matrix att[16,8] = out["ul",1]

*****************************************************************************
* Q2.7: Propensity Score Matching (rows 17-19)

* ate, experimental data, three ways
qui teffects psmatch (re78) (t `a', probit) if (treat != 2), ate
matrix out = r(table)
matrix ate[17,1] = out["b",1]
matrix ate[17,2] = out["se",1]
matrix ate[17,3] = out["ll",1]
matrix ate[17,4] = out["ul",1]

qui teffects psmatch (re78) (t `b', probit) if (treat != 2), ate
matrix out = r(table)
```

```
matrix ate[18,1] = out["b",1]
matrix ate[18,2] = out["se",1]
matrix ate[18,3] = out["ll",1]
matrix ate[18,4] = out["ul",1]

qui teffects psmatch (re78) (t `c', probit) if (treat != 2), ate
matrix out = r(table)
matrix ate[19,1] = out["b",1]
matrix ate[19,2] = out["se",1]
matrix ate[19,3] = out["ll",1]
matrix ate[19,4] = out["ul",1]

* att, experimental data, three ways
qui teffects psmatch (re78) (t `a', probit) if (treat != 2), atet
matrix out = r(table)
matrix att[17,1] = out["b",1]
matrix att[17,2] = out["se",1]
matrix att[17,3] = out["ll",1]
matrix att[17,4] = out["ul",1]

qui teffects psmatch (re78) (t `b', probit) if (treat != 2), atet
matrix out = r(table)
matrix att[18,1] = out["b",1]
matrix att[18,2] = out["se",1]
matrix att[18,3] = out["ll",1]
matrix att[18,4] = out["ul",1]

qui teffects psmatch (re78) (t `c', probit) if (treat != 2), atet
matrix out = r(table)
matrix att[19,1] = out["b",1]
matrix att[19,2] = out["se",1]
matrix att[19,3] = out["ll",1]
matrix att[19,4] = out["ul",1]

* three ways, PSID data, ate and att
qui probit t `a' if (treat != 0)
capture: drop prop
predict prop
qui teffects psmatch (re78) (t `a') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate
matrix out = r(table)
matrix ate[17,5] = out["b",1]
matrix ate[17,6] = out["se",1]
matrix ate[17,7] = out["ll",1]
matrix ate[17,8] = out["ul",1]

qui teffects psmatch (re78) (t `a') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet
matrix out = r(table)
matrix att[17,5] = out["b",1]
matrix att[17,6] = out["se",1]
matrix att[17,7] = out["ll",1]
matrix att[17,8] = out["ul",1]
```

```
qui probit t `b' if (treat != 0)
capture: drop prop
predict prop
qui teffects psmatch (re78) (t `b') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate
matrix out = r(table)
matrix ate[18,5] = out["b",1]
matrix ate[18,6] = out["se",1]
matrix ate[18,7] = out["ll",1]
matrix ate[18,8] = out["ul",1]

qui teffects psmatch (re78) (t `b') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet
matrix out = r(table)
matrix att[18,5] = out["b",1]
matrix att[18,6] = out["se",1]
matrix att[18,7] = out["ll",1]
matrix att[18,8] = out["ul",1]

qui probit t `c' if (treat != 0)
capture: drop prop
predict prop
qui teffects psmatch (re78) (t `c') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), ate
matrix out = r(table)
matrix ate[19,5] = out["b",1]
matrix ate[19,6] = out["se",1]
matrix ate[19,7] = out["ll",1]
matrix ate[19,8] = out["ul",1]

qui teffects psmatch (re78) (t `c') ///
 if (treat==1 | (treat==2 & prop >= .0001 & prop <= .9999) ), atet
matrix out = r(table)
matrix att[19,5] = out["b",1]
matrix att[19,6] = out["se",1]
matrix att[19,7] = out["ll",1]
matrix att[19,8] = out["ul",1]

*******************************************************************************
* Q2: Output table to Excel, then convert that to LaTeX

* write to c2 so we can add row, heading columns in Excel
putexcel set ate_s, modify
putexcel b3 = matrix(ate)
putexcel close

putexcel set att_s, modify
putexcel b3 = matrix(att)
putexcel close

*******************************************************************************
*** Question 3: Post-model Selection Inference
*******************************************************************************
* Q3 setup
```

```
clear all

set seed 22
set obs 50

*******************************************************************************
* Q3.1: Summary Statistics and Kernel Density Plots

* this doesn't work...can we use bootstrap somehow? or do in mata?

* number of replications
local M = 1000
set matsize 11000

* empty matrices to store estimates and indicator of coverage
matrix est = J(`M',3,.)
matrix cov = J(`M',3,.)

* initial values we will replace during replication
gen x = rnormal(0,1)
gen z = .85*x + sqrt(1-.85)*rnormal(0,1)
gen eps = rnormal(0,1)
gen y = 1 + .5*x + z + eps

* loop for M replications
forvalues i = 1/`M'{
        qui replace x = rnormal(0,1)
        qui replace z = .85*x + sqrt(1-.85)*rnormal(0,1)
        qui replace eps = rnormal(0,1)
        qui replace y = 1 + .5*x + z + eps

        * long regression
        qui reg y x z, r

        * extract first estimate
        local beta_hat = _b["x"]
        matrix est[`i',1] = `beta_hat'

        * get SE and calculate coverage of true beta_0 = .5
        local se_hat = _se["x"]
        local lb_hat = `beta_hat' - 1.96 * `se_hat'
        local ub_hat = `beta_hat' + 1.96 * `se_hat'
        local cov_hat = (.5 >= `lb_hat') & (.5 <= `ub_hat')
        matrix cov[`i',1] = `cov_hat'

        * save gamma over se gamma
        local gamma_hat = _b["z"]
        local gamma_se  = _se["z"]
        local tstat = `gamma_hat'/`gamma_se'

        * short regression
        qui reg y x, r
        local beta_tilde = _b["x"]
        matrix est[`i',2] = `beta_tilde'
```

```
        * get SE and calculate coverage of true beta_0 = .5
        local se_tilde = _se["x"]
        local lb_tilde = `beta_tilde' - 1.96 * `se_tilde'
        local ub_tilde = `beta_tilde' + 1.96 * `se_tilde'
        local cov_tilde = (.5 >= `lb_tilde') & (.5 <= `ub_tilde')
        matrix cov[`i',2] = `cov_tilde'

        * third estimate
        local beta_check = cond(`tstat' >= 1.96, `beta_hat', `beta_tilde')
        matrix est[`i',3] = cond(`tstat' >= 1.96, `beta_hat', `beta_tilde')
        matrix cov[`i',3] = cond(`tstat' >= 1.96, `cov_hat', `cov_tilde')
}

* turn results into variables
svmat est
svmat cov

* drop old data
drop x
drop z
drop eps
drop y

* rename variables
rename est1 beta_hat
rename est2 beta_tilde
rename est3 beta_check
rename cov1 cov_hat
rename cov2 cov_tilde
rename cov3 cov_check

* write summary statistics to latex
outreg2 using q3.tex, replace sum(log) ///
        keep(beta_hat beta_tilde beta_check) ///
        eqkeep(min mean median max) ///
        dec(2)

* kernel densities
twoway kdensity beta_hat, k(epanechnikov) || ///
 kdensity beta_tilde, k(epanechnikov) || ///
 kdensity beta_check, k(epanechnikov) ///
 leg(lab(1 "beta_hat") lab(2 "beta_tilde") lab(3 "beta_check")) ///
 ytitle("Density") xtitle("")

* save
graph export q3_s.png, replace

*******************************************************************************
* Q3.2: Empirical Coverage Rates

* calculate these here, report them in LaTeX
sum(cov_hat)
sum(cov_tilde)
```

```
sum(cov_check)

*******************************************************************************
log close
*******************************************************************************
```