Economics 675: Applied Microeconometrics
Fall 2018 - Assignment 3

Paul R. Organ

October 26, 2018

Contents

1 Question 1: Non-linear Least Squares

1.1 Theory

1. We can use the "add and subtract" method to determine under what conditions $\boldsymbol{\beta}_0$ is identifiable:

$$\begin{aligned}\mathbb{E}[(y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2] &= \mathbb{E}[(y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}_0) + \mu(\mathbf{x}_i'\boldsymbol{\beta}_0) - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2] \\ &= \mathbb{E}[(y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}_0))^2] + \mathbb{E}[(\mu(\mathbf{x}_i'\boldsymbol{\beta}_0) - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2] + 2\mathbb{E}[(y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}_0))(\mu(\mathbf{x}_i'\boldsymbol{\beta}_0) - \mu(\mathbf{x}_i'\boldsymbol{\beta}))] \\ &= \mathbb{E}[(y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}_0))^2] + \mathbb{E}[(\mu(\mathbf{x}_i'\boldsymbol{\beta}_0) - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2]\end{aligned}$$

The interaction term drops out by the Law of Iterated Expectations. Then, we want to minimize this by choosing $\boldsymbol{\beta}_0 = \boldsymbol{\beta}$, so we will have identification if $\mathbb{E}[(\mu(\mathbf{x}_i'\boldsymbol{\beta}_0) - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2] \neq 0$, for all $\boldsymbol{\beta}_0 \neq \boldsymbol{\beta}$.

2. We can take the derivative of the M-estimator to yield the Z-estimator with the following condition:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu(\mathbf{x}_i'\hat{\boldsymbol{\beta}}))\dot{\mu}(\mathbf{x}_i'\hat{\boldsymbol{\beta}})\mathbf{x}_i = \frac{1}{n}\sum_{i=1}^{n} m(y_i', \mathbf{x}_i, \hat{\boldsymbol{\beta}}_n) = 0$$

Then, we can assume $\hat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}_0 \to_p 0$ (really we assume the objective function is uniformly consistent) and if we have differentiability, we can derive the usual M-estimation (sandwich) form of the asymptotic variance:

$$\mathbf{H}_0 = \mathbb{E}\Big[\frac{\partial}{\partial\beta} m(y_i', \mathbf{x}_i, \boldsymbol{\beta}_0)\Big] = -\mathbb{E}\Big[\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta}_0)^2\mathbf{x}_i\mathbf{x}_i'\Big]$$

$$\boldsymbol{\Sigma}_0 = \mathbb{V}[m(y_i', \mathbf{x}_i, \boldsymbol{\beta}_0)] = \mathbb{E}[\sigma^2(\mathbf{x}_i)\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta}_o)^2\mathbf{x}_i\mathbf{x}_i']$$

$$\sqrt{n}(\hat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}_0) \to_d \mathcal{N}(0, \mathbf{V}_0) \stackrel{d}{=} \mathcal{N}(0, \mathbf{H}_0^{-1}\boldsymbol{\Sigma}_0\mathbf{H}_0^{-1})$$

3. We can "put a hat" on our equations above to estimate $\mathbf{V}_0$. That is, let

$$\hat{\mathbf{H}}_n = -\hat{\mathbb{E}}\Big[\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta}_0)^2\mathbf{x}_i\mathbf{x}_i'\Big] = -\frac{1}{n}\sum_{i-1}^{n}\dot{\mu}(\mathbf{x}_i'\hat{\boldsymbol{\beta}}_n)^2\mathbf{x}_i\mathbf{x}_i'$$

$$\hat{\boldsymbol{\Sigma}}_n = \hat{\mathbb{E}}[\sigma^2(\mathbf{x}_i)\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta}_o)^2\mathbf{x}_i\mathbf{x}_i'] = \frac{1}{n}\hat{\varepsilon}_i^2\dot{\mu}(\mathbf{x}_i'\hat{\boldsymbol{\beta}}_n)^2\mathbf{x}_i\mathbf{x}_i'$$

replacing $\sigma^2(\mathbf{x}_i)$ with $\hat{\varepsilon}_i = y_i - \mu(\mathbf{x}_i'\hat{\boldsymbol{\beta}}_n)$. Then combine to yield: $\hat{\mathbf{V}}_n^{HC} = \hat{\mathbf{H}}_n^{-1}\hat{\boldsymbol{\Sigma}}_n\hat{\mathbf{H}}_n^{-1}$.

Regarding consistency, we can define our g function and apply the delta method. Let $g(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||^2 = \sum_d \boldsymbol{\beta}_d^2$. Then we have $\dot{g}(\boldsymbol{\beta}) = 2\boldsymbol{\beta}'$, and thus:

$$\sqrt{n}(||\hat{\boldsymbol{\beta}}_n||^2 - ||\boldsymbol{\beta}_0||^2) = \sqrt{n}(g(\hat{\boldsymbol{\beta}}_n) - g(\boldsymbol{\beta}_0)) \to_d \mathcal{N}(0, \dot{g}(\boldsymbol{\beta}_0)\mathbf{V}_0\dot{g}(\boldsymbol{\beta}_0)') = \mathcal{N}(0, 4\boldsymbol{\beta}_0'\mathbf{V}_0\boldsymbol{\beta}_0)$$

From this, we can easily construct the confidence interval as follows:

$$\text{CI}_{95\%} = \Big[||\hat{\boldsymbol{\beta}}_n||^2 \pm 1.96 \times \sqrt{\frac{4\hat{\boldsymbol{\beta}}_n'\hat{\mathbf{V}}_n\hat{\boldsymbol{\beta}}_n}{n}}\Big]$$

4. Here we will have the usual simplification, where parts of our sandwich form cancel out leaving a simpler form for the asymptotic variance.

If $\sigma^2(\mathbf{x}_i) = \sigma^2$, note that $\boldsymbol{\Sigma}_0 = \mathbb{E}[\sigma^2(\mathbf{x}_i)\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta}_0)^2\mathbf{x}_i\mathbf{x}_i'] = \sigma^2\mathbb{E}[\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta}_0)^2\mathbf{x}_i\mathbf{x}_i'] = -\sigma^2\mathbf{H}_0$. So we have:

$$\mathbf{V}_0 = -\sigma^2\mathbf{H}_0^{-1}\mathbf{H}_0\mathbf{H}_0^{-1} = -\sigma^2\mathbf{H}_0^{-1}$$

To estimate $\mathbf{V}_0$ in this case, we can do the same as above, i.e., put hats on each component. $\hat{\mathbf{H}}_n$ remains the same as above. To estimate σ^2, we use $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\hat{\boldsymbol{\beta}}_n))^2$.

Because the g function is the same as in part (3), the confidence interval will be the same, just using a different estimator for $\mathbf{V}_0$.

5. Using our 671 skills, and knowledge of the normal density, we have the following:

$$f_{y|\mathbf{x}}(y_i|\mathbf{x}_i) = \frac{1}{(\sqrt{2\pi\sigma^2})^n} \exp\Big(-\frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2 \Big)$$

Taking logs yields the log-likehood:

$$\ell_n = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2$$

The log-likelihood depends on two parameters, $\boldsymbol{\beta}$ and σ^2. Thus we have two FOCs:

$$\frac{\partial}{\partial\beta}\ell_n = \frac{1}{\sigma^2}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}))\dot{\mu}(\mathbf{x}_i'\boldsymbol{\beta})\mathbf{x}_i = 0$$

$$\frac{\partial}{\partial\sigma^2}\ell_n = -\frac{n}{4\pi\sigma^2} + \frac{1}{\sigma^4}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2 = 0$$

If we replace $\boldsymbol{\beta}$ with $\hat{\boldsymbol{\beta}}_n$, the FOC for $\boldsymbol{\beta}$ is nearly identical to that we derived in part (2) for the M-estimator (the only difference is the $\frac{1}{\sigma^2}$ term, which does not affect the choice of $\hat{\boldsymbol{\beta}}_n$). Thus we have $\hat{\boldsymbol{\beta}}_{ML} = \hat{\boldsymbol{\beta}}_n$.

For the MLE of σ^2, we can solve the FOC for σ^2 to yield $\sigma^2 = \frac{1}{n}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\boldsymbol{\beta}))^2$. To estimate this, we just need to put the hats on: $\hat{\sigma}^2_{ML} = \frac{1}{n}\sum_{i=1}^n (y_i - \mu(\mathbf{x}_i'\hat{\boldsymbol{\beta}}_{ML}))^2$ which is the same as we used in part (4) above.

6. Consider the following example, showing that with an unknown link function, $\boldsymbol{\beta}_0$ is not identifiable.

Take two pairs of parameters:
Pair 1: $\big(\mu_1(\cdot), \boldsymbol{\beta}_1\big)$
Pair 2: $\big(\mu_2(\cdot), \boldsymbol{\beta}_2\big)$, with $\mu_2(\cdot) = \mu_1(2\times\cdot)$ and $\boldsymbol{\beta}_2 = \frac{1}{2}\boldsymbol{\beta}_1$.
Then we have $\mu_2(\mathbf{x}'\boldsymbol{\beta}_2) = \mu_1\big(2\mathbf{x}'(\frac{1}{2}\boldsymbol{\beta}_1)\big) = \mu_1(\mathbf{x}'\boldsymbol{\beta}_1)$. The link functions can yield the same numerical result even with different parameters, and thus $\boldsymbol{\beta}$ is not identifiable.

To allow for identification, we can normalize $\boldsymbol{\beta}$ by requiring $||\boldsymbol{\beta}_0|| = 1$.

7. Here we have y_i taking on values of 1 or 0. We have:

$$\mu(\mathbf{x}_i'\boldsymbol{\beta}_0) = \mathbb{E}[y_i|\mathbf{x}_i] = \mathbb{E}[\mathbf{1}(\mathbf{x}_i'\boldsymbol{\beta}_0 - \varepsilon_i \geq 0)|\mathbf{x}_i] = \mathbb{E}[\mathbf{x}_i'\boldsymbol{\beta}_0 \geq \varepsilon_i|\mathbf{x}_i] = \mathbb{E}[\mathbf{x}_i'\boldsymbol{\beta}_0 \leq \varepsilon_i] = F(\mathbf{x}_i'\boldsymbol{\beta}_0)$$

We can drop the condition on $\mathbf{x}_i$ since the given F for $\varepsilon_i|\mathbf{x}_i$ is independent of $\mathbf{x}_i$.

In addition, since y_i takes values of 1 or 0, we have a Bernoulli r.v., and hence we have:

$$\sigma^2(\mathbf{x}_i) = F(\mathbf{x}_i'\boldsymbol{\beta}_0)\big(1 - F(\mathbf{x}_i'\boldsymbol{\beta}_0)\big)$$

Now we can plug in the logistic c.d.f. to the M- and Z-estimation from the start of this question to estimate $\boldsymbol{\beta}_0$. Note that as we saw in part (6), we need to make an assumption about the link function to allow identifiability. Here, we can assume $s_0 = 1$. Then we have:

$$M(y_i, \mathbf{x}_i, \boldsymbol{\beta}) = (y_i - F(\mathbf{x}_i'\boldsymbol{\beta}))^2$$

$$m(y_i, \mathbf{x}_i, \boldsymbol{\beta}) = (y_i - F(\mathbf{x}_i'\boldsymbol{\beta}))f(\mathbf{x}_i'\boldsymbol{\beta})\mathbf{x}_i$$

For the logistic distribution with $s_0 = 1$, we have $f(u) = F(u)\big(1 - F(u)\big)$, so the m equation becomes:

$$m(y_i, \mathbf{x}_i, \boldsymbol{\beta}) = \big(y_i - F(\mathbf{x}_i'\boldsymbol{\beta})\big)F(\mathbf{x}_i'\boldsymbol{\beta})\big(1 - F(\mathbf{x}_i'\boldsymbol{\beta})\big)\mathbf{x}_i$$

Then using our equations from part (2), we can estimate $\boldsymbol{\beta}_0$ with the Z-estimator $\mathbb{E}[m(y_i, \mathbf{x}_i, \boldsymbol{\beta})] = 0$, which also gives us $\mathbf{H}_0$ and $\boldsymbol{\Sigma}_0$ as follows:

$$\mathbf{H}_0 = -\mathbb{E}[f(\mathbf{x}_i'\boldsymbol{\beta}_0)\mathbf{x}_i\mathbf{x}_i'], \qquad \boldsymbol{\Sigma}_0 = \mathbb{E}[m(y_i, x_i, \boldsymbol{\beta}_0)m(y_i, x_i, \boldsymbol{\beta}_0)']$$

which we can plug into our sandwich estimator to yield $\mathbf{V}_0 = \mathbf{H}_0^{-1}\boldsymbol{\Sigma}_0\mathbf{H}_0^{-1}$.

8. First, we define the likelihood function as follows:

$$L_i(\boldsymbol{\beta}) = \Big(\frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}\Big)^{y_i}\Big(\frac{\exp(-\mathbf{x}_i'\boldsymbol{\beta})}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}\Big)^{1-y_i}$$

Then we have log-likelihood $\ell_i(\boldsymbol{\beta}) = -\log(1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})) - (1 - y_i)\mathbf{x}_i'\boldsymbol{\beta}$.

And from the log-likelihood, we derive the score function:

$$s_i(\boldsymbol{\beta}) = \frac{\partial}{\partial\beta}l_i(\boldsymbol{\beta}) = \frac{\mathbf{x}_i \exp(-\mathbf{x}_i'\boldsymbol{\beta})}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})} - \mathbf{x}_i(1 - y_i) = \mathbf{x}_i\Big(y_i - \frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}\Big) = \mathbf{x}_i\big(y_i - F(\mathbf{x}_i'\boldsymbol{\beta})\big)$$

And it follows that the MLE estimation and the NLS estimation lead to different results.

1.2 Implementation

(a) See the code in Appendix A for R and Appendix B for Stata. Table 1 shows my results from R; Table 2 shows the corresponding results from Stata. The results are essentially identical.

Table 1: Logistic Regression Results: R

	Estimate	Robust SE	Z-value	p-Value	CI-low	CI-high
(Intercept)	1.755	0.334	5.247	0.0000	1.100	2.411
S_age	1.333	0.123	10.832	0.0000	1.092	1.575
S_HHpeople	-0.067	0.023	-2.872	0.0041	-0.112	-0.021
log_SincpcP1	-0.119	0.044	-2.709	0.0068	-0.205	-0.033

Table 2: Logistic Regression Results: Stata

	(1)	(2)	(3)	(4)	(5)
VARIABLES	coef	se	tstat	pval	ci
S_age	1.333	0.123	10.832	0.000	1.092 - 1.575
S_HHpeople	-0.067	0.023	-2.872	0.004	-0.112 - -0.021
log_SincpcP1	-0.119	0.044	-2.708	0.007	-0.205 - -0.033
Constant	1.755	0.334	5.247	0.000	1.099 - 2.411

(b) Table 3 shows my results from R; Table 4 shows the corresponding results from Stata. The coefficient estimates match, while standard errors differ slightly (perhaps because of different random draws during bootstrapping).

(c) Figure 1 shows my results from R; Figure 2 shows the corresponding results from Stata. I plot the fitted propensity scores, along with three different kernel density estimates.

Table 3: Nonparametric Bootstrap Results: R

	Coefficient	Std Error	t-Value	p-Value	CI low	CI high
(Intercept)	1.755	0.332	5.285	0.0000	1.156	2.430
S_age	1.333	0.129	10.351	0.0000	1.117	1.608
S_HHpeople	-0.067	0.024	-2.747	0.0020	-0.110	-0.015
log(S_incomepc+1)	-0.119	0.042	-2.799	0.0100	-0.207	-0.044

Table 4: Nonparametric Bootstrap Results: STATA

	(1)	(2)	(3)	(4)	(5)
VARIABLES	coef	se	tstat	pval	ci
S_age	1.333	0.131	10.165	0.000	1.076 - 1.590
S_HHpeople	-0.067	0.024	-2.810	0.005	-0.113 - -0.020
log_SincpcP1	-0.119	0.046	-2.601	0.009	-0.208 - -0.029
Constant	1.755	0.344	5.098	0.000	1.080 - 2.430

Figure 1: Fitted Propensity Histogram and Kernel Densities: R

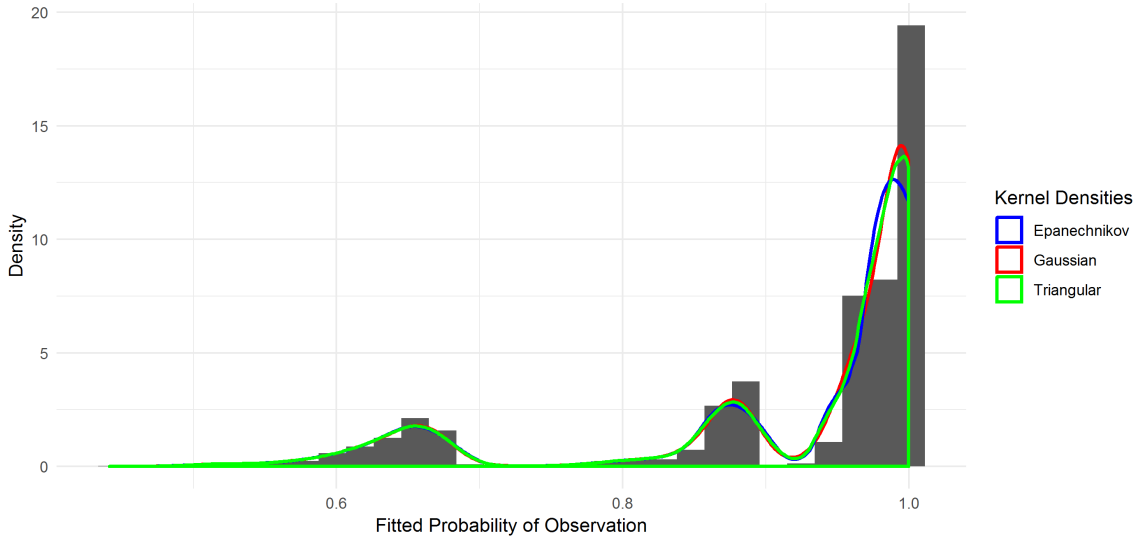
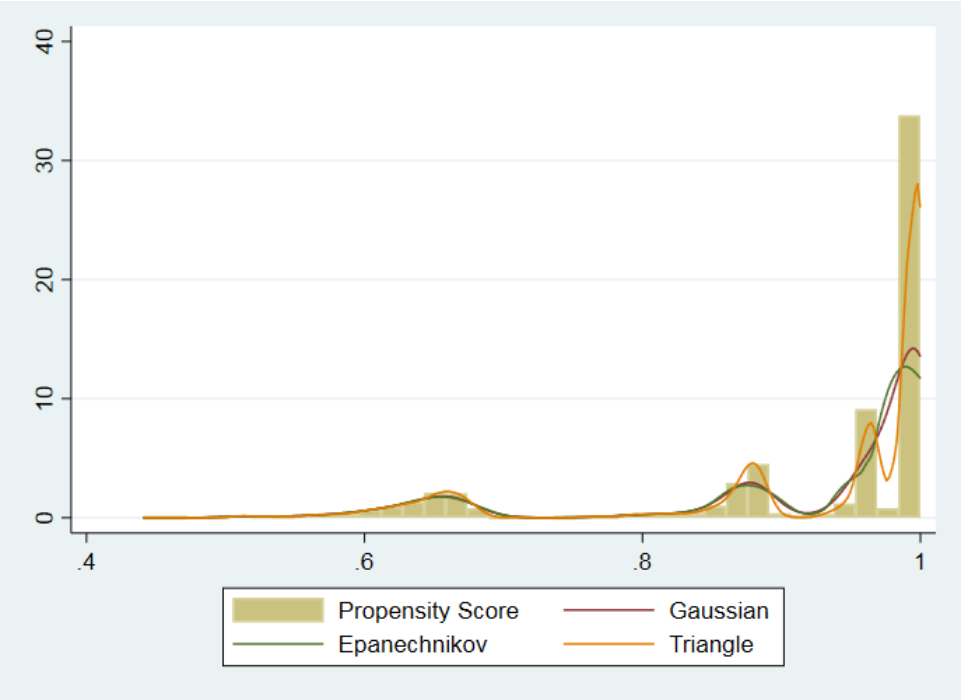


Figure 2: Fitted Propensity Histogram and Kernel Densities: STATA



2 Question 2: Semiparametric GMM with Missing Data

2.1 Theory

Here we rely on Wooldridge (2010), Section 14.4.3 ("Efficient Choice of Instruments under Conditional Moment Restrictions"), pp. 542-545. Equation (14.57) gives us the optimal choice of instrument. To apply this equation, we need two ingredients: (1) the conditional variance of the moment function m, and (2) the expectation of the gradient of the moment function m.

Here, by assumption we have $\mathbb{E}[m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|t_i, \mathbf{x}_i] = 0$. From this we also have our first ingredient:

$$\mathbb{V}[m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|t_i, \mathbf{x}_i] = F(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)\big(1 - F(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)\big) = \boldsymbol{\Omega}_0(\mathbf{x}_i) \text{ (using Wooldridge's notation)}$$

For the second ingredient, we take derivatives of the moment function m wrt both components of $\boldsymbol{\beta}$:

$$\begin{aligned}
\mathbb{E}\Big[\frac{\partial}{\partial \beta} m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|t_i, \mathbf{x}_i\Big] &= \begin{bmatrix} \mathbb{E}[\frac{\partial}{\partial \theta} m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|t_i, \mathbf{x}_i] \\ \mathbb{E}[\frac{\partial}{\partial \gamma} m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|t_i, \mathbf{x}_i] \end{bmatrix} \\
&= \begin{bmatrix} \mathbb{E}[-t_i \cdot f(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)|t_i, \mathbf{x}_i] \\ \mathbb{E}[-\mathbf{x}_i \cdot f(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)|t_i, \mathbf{x}_i] \end{bmatrix} \\
&= -f(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0) \begin{bmatrix} t_i \\ \mathbf{x}_i' \end{bmatrix} \\
&= \mathbf{R}_0(\mathbf{x}_i) \text{ (using Wooldridge's notation)}
\end{aligned}$$

Now applying Wooldridge's equation (14.57), we obtain the optimal choice of instruments:

$$\mathbf{Z}^*(\mathbf{x}_i) = \boldsymbol{\Omega}_0(\mathbf{x}_i)^{-1}\mathbf{R}_0(\mathbf{x}_i) = \frac{-f(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)}{F(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)\big(1 - F(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)} \begin{bmatrix} t_i \\ \mathbf{x}_i' \end{bmatrix}$$

which is the $\mathbf{g}_0(t_i, \mathbf{x}_i)$ given in the problem (times -1, which we can ignore since we are setting the condition equal to 0).

Finally, we know that if $F(\cdot)$ is the logistic cdf, then $f(\cdot) = F(\cdot)\big(1 - F(\cdot)\big)$. In this case, the top and bottom of our $\mathbf{Z}^*(\mathbf{x}_i)$ equation will cancel, leaving the optimal instruments as simply $\begin{bmatrix} t_i \\ \mathbf{x}_i' \end{bmatrix}$.

2.2 Missing Completely at Random (MCAR)

(a) From above, we have the optimal unconditional moment condition $\mathbf{0} = \mathbb{E}[\mathbf{g}_0(t_i, \mathbf{x}_i)m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)]$.

Under the MCAR assumption, we can replace y_i^* and y_i (conditional on $s_i = 1$), so we have:

$$\mathbb{E}[\mathbf{g}_0(t_i, \mathbf{x}_i)m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)] = \mathbb{E}[\mathbf{g}_0(t_i, \mathbf{x}_i)m(y_i^*, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|s_i = 1] = \mathbb{E}[\mathbf{g}_0(t_i, \mathbf{x}_i)m(y_i, t_i, \mathbf{x}_i, \boldsymbol{\beta}_0)|s_i = 1]$$

and putting a hat on this allows us to estimate $\hat{\boldsymbol{\beta}}_{MCAR}$ as stated in the problem. Because we know the form of the $\mathbf{g}_0$ function, we can estimate it and calculate the "optimal" instruments.

(b) See Table 5 for my results from R, and Table 6 for my results from Stata. For the code, see Appendices A and B.

Table 5: MCAR - Nonparametric Bootstrap Results: R

	Coefficient	Std Error	t-Value	p-Value	CI low	CI high
dpisofirme	-0.317	0.081	-3.925	0.0180	-0.446	-0.182
S_age	-0.244	0.020	-11.999	0.0000	-0.284	-0.204
S_HHpeople	0.024	0.014	1.720	0.0821	-0.003	0.051
log(S_incomepc+1)	0.033	0.014	2.365	0.0200	0.006	0.058

Table 6: MCAR - Nonparametric Bootstrap Results: STATA

VARIABLES	Coefficient	Std Error	t-Value	CI low	CI high
dpisofirme	-.316	.069	-4.541	-.452	-.186
S_age	-.246	.021	-11.504	-.288	-.215
S_HHpeople	.021	.015	1.382	-.009	-.054
log(S_incomepc+1)	.035	.015	2.339	.006	.064

2.3 Missing at Random (MAR)

(a) With the MAR assumption, we can proceed as we did in part 2.1, constructing the two ingredients for the optimal instrument equation. We will show that this process leads to the same result.

First, by MAR we can split out the conditional moment restriction, and then calculate the conditional variance restriction:

$$\mathbb{E}[s_i \cdot m(y_i^*, t_i, \mathbf{x}_i)|t_i, \mathbf{x}_i] = \mathbb{E}[s_i|t_i, \mathbf{x}_i] \cdot \mathbb{E}[m(y_i^*, t_i, \mathbf{x}_i)|t_i, \mathbf{x}_i] = 0$$

and the conditional variance follows (note $s_i^2 = s_i$ since $s_i \in \{0, 1\}$):

$$\mathbb{V}[s_i \cdot m(y_i^*, t_i, \mathbf{x}_i)|t_i, \mathbf{x}_i] = \mathbb{E}[s_i^2 \cdot m(y_i^*, t_i, \mathbf{x}_i)^2|t_i, \mathbf{x}_i] = \mathbb{E}[s_i|t_i, \mathbf{x}_i] \cdot F(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)\big(1 - F(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0)\big)$$

The second ingredient follows similarly, so that we have:

$$\mathbb{E}\Big[\frac{\partial}{\partial \beta}\Big(s_i \cdot m(y_i^*, t_i, \mathbf{x}_i)\Big)|t_i, \mathbf{x}_i\Big] = -\mathbb{E}[s_i|t_i, \mathbf{x}_i] \cdot f(t_i \cdot \theta_0 + \mathbf{x}_i'\gamma_0) \begin{bmatrix} t_i \\ \mathbf{x}_i' \end{bmatrix}$$

Combining the two ingredients, the $\mathbb{E}[s_i|t_i, \mathbf{x}_i]$ in each will cancel out, leaving exactly the result we found in part 2.1.

(b) As noted in the problem, the propensity score $p_0(t_i, \mathbf{x}_i) = \mathbb{E}[s_i|t_i, \mathbf{x}_i]$ is unknown. Hence to estimate $\tilde{\boldsymbol{\beta}}_{MAR}$, we must first estimate p_0, which we can then plug into the moment condition as specified in the problem.

(c) As I noted in part (b), we must first estimate p_0, and then can proceed with GMM. In this case, I use a logit regression to estimate p_0. Code is in the appendices (A and B). My results are in Tables 7 (R) and 8 (STATA).

Table 7: MAR - Nonparametric Bootstrap Results: R

	Coefficient	Std Error	t-Value	p-Value	CI low	CI high
dpisofirme	-0.304	0.084	-3.622	0.0240	-0.437	-0.133
S_age	-0.246	0.021	-11.943	0.0000	-0.285	-0.206
S_HHpeople	0.026	0.013	1.952	0.0801	-0.003	0.051
log(S_incomepc+1)	0.031	0.014	2.162	0.0260	0.003	0.060

Table 8: MAR - Nonparametric Bootstrap Results: Stata

VARIABLES	Coefficient	Std Error	t-Value	CI low	CI high
dpisofirme	-.308	.065	-4.747	-.436	-.181
S_age	-.245	.022	-11.038	-.289	-.202
S_HHpeople	.022	.015	1.405	-.009	.052
log(S_incomepc+1)	.034	.015	2.298	.005	.063

(d) My trimmed results are in Tables 9 (R) and 10 (Stata). See the appendices for the underlying code (A and B).

As can be seen in the tables, in this case, trimming does not affect the results.

Table 9: MAR - Nonparametric Bootstrap Results (Trimmed): R

	Coefficient	Std Error	t-Value	p-Value	CI low	CI high
dpisofirme	-0.304	0.084	-3.622	0.0240	-0.437	-0.133
S_age	-0.246	0.021	-11.943	0.0000	-0.285	-0.206
S_HHpeople	0.026	0.013	1.952	0.0801	-0.003	0.051
log(S_incomepc+1)	0.031	0.014	2.162	0.0260	0.003	0.060

Table 10: MAR - Nonparametric Bootstrap Results (Trimmed): Stata

VARIABLES	Coefficient	Std Error	t-Value	CI low	CI high
dpisofirme	-.308	.063	-4.9	-.432	-.185
S_age	-.245	.02	-12.223	-.285	-.206
S_HHpeople	.022	.015	1.438	-.008	.051
log(S_incomepc+1)	.034	.016	2.165	.003	.065

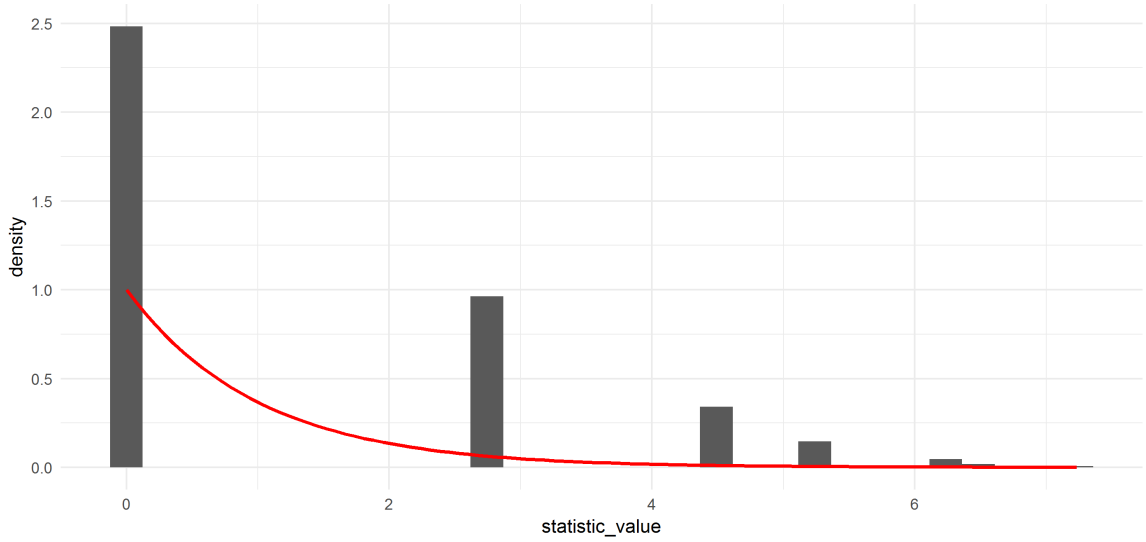
3 Question 3: When Bootstrap Fails

3.1 Nonparametric Bootstrap

I plot my results in Figures 3 (R) and 4 (STATA) below. See the Appendices for the underlying code.

Note from the figures that the bootstrap statistic takes only a few discrete values, even given the 599 replications. We see the bootstrap fails in this case.

Figure 3: Nonparametric Bootstrap - Histogram and Exp(1) Distribution: R



3.2 Parametric Bootstrap

I plot my results in Figures 5 (R) and 6 (STATA) below. See the Appendices for the underlying code.

Here we see that the bootstrap statistic is not limited to a few discrete values, but rather is a much closer approximation to the Exp(1) distribution.

3.3 Intuition

Because we are sampling with replacement, the probability $\mathbb{P}[\max_i x_i^* = max_i x_i]$ is simply $1 - \mathbb{P}[$none of the n draws are the max$]$. Thus we have:

$$\mathbb{P}[\max_i x_i^* = \max_i x_i] = 1 - \left(\frac{n-1}{n}\right)^n \to 1 - \frac{1}{e} > 0$$

So we have a strictly positive probability of the statistic equalling zero, and hence a mass point at zero (as we see in the figures).

Figure 4: Nonparametric Bootstrap - Histogram and Exp(1) Distribution: STATA

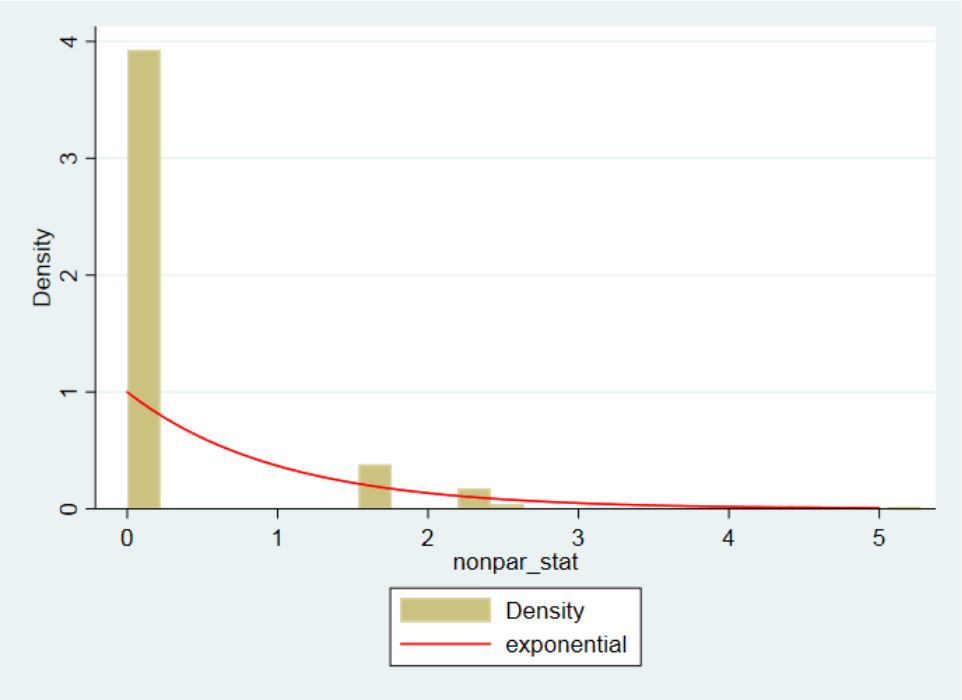


Figure 5: Parametric Bootstrap - Histogram and Exp(1) Distribution: R

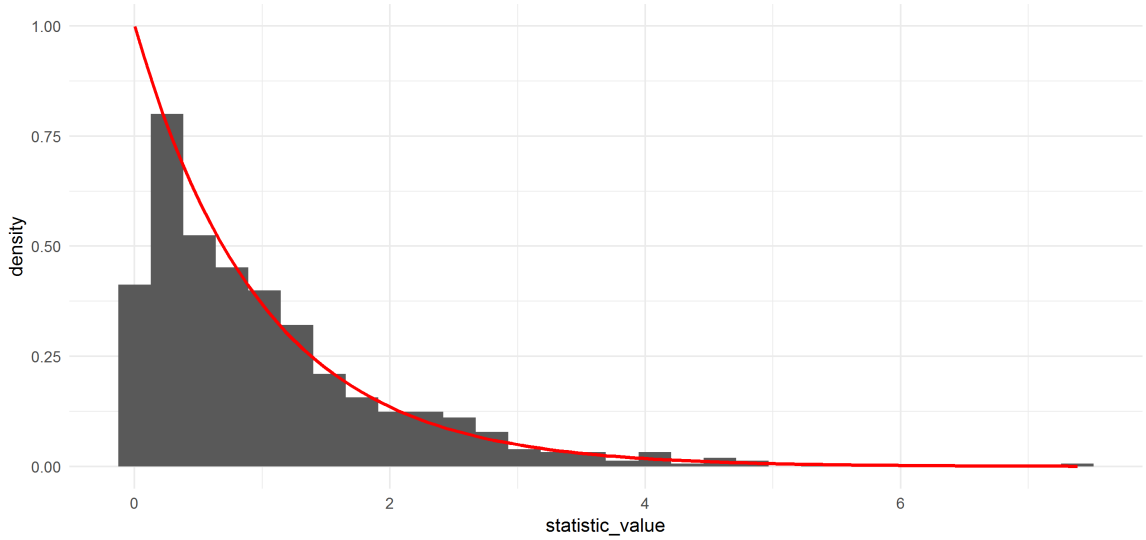
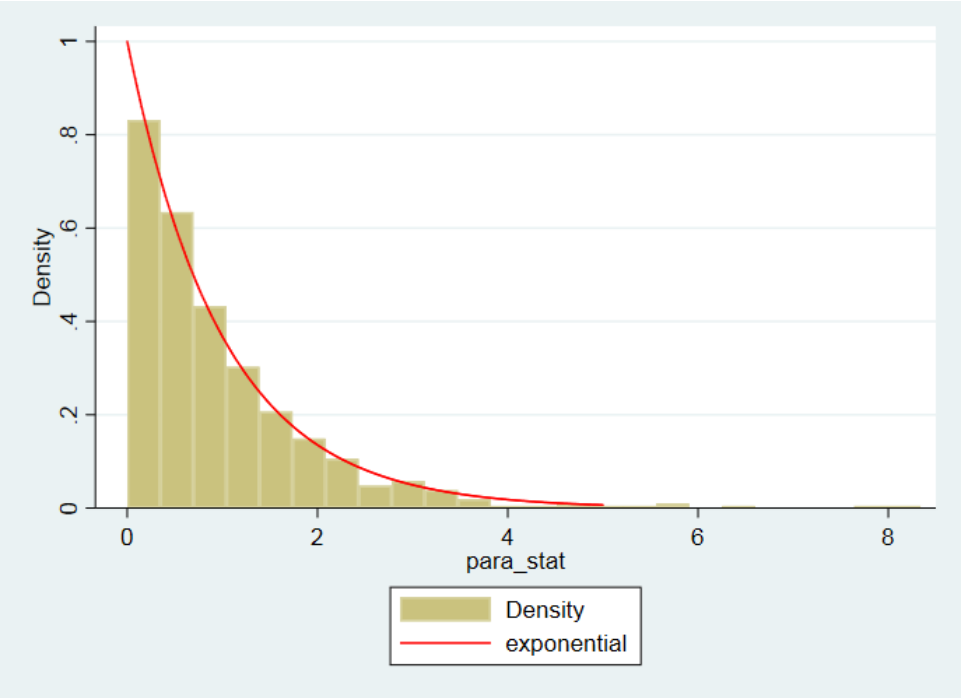


Figure 6: Parametric Bootstrap - Histogram and Exp(1) Distribution: STATA



A R Code

```
#############################################################################################
# Author: Paul R. Organ
# Purpose: ECON 675, PS3
# Last Update: Oct 26, 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(gmm)       # GMM estimation

setwd('C:/Users/prorgan/Box/Classes/Econ 675/Problem Sets/PS3')

#############################################################################################
### Question 1: Non-linear Least Squares
#############################################################################################
# Q1.9 setup

# read in data
df <- read_csv('pisofirme.csv')

# add variable to indicate having outcome data
df %<>% mutate(s = 1-dmissing)

# add log variable for use in regression [log(S_incomepc + 1)]
df %<>% mutate(log_SincpcP1 = log(S_incomepc + 1))

# logistic regression model
reg <- glm(s ~ S_age + S_HHpeople + log_SincpcP1,
           data = df, family = binomial(link = 'logit'))

# Q1.9a - estimates and statistics
# extract coef, var, t-stat, p-val, 95% CI

# robust standard errors
cov <- vcovHC(reg, type = "HC0")
std.err <- sqrt(diag(cov))

# for CI
q.val <- qnorm(0.975)

# create table
t1 <- cbind(
  Estimate = coef(reg)
  , 'Robust SE' = std.err
  , 'Z-value' = (coef(reg)/std.err)
```

```
  , 'p-Value' =  2 * pnorm(abs(coef(reg)/std.err), lower.tail = FALSE)
  , 'CI-low' = coef(reg) - q.val  * std.err
  , 'CI-high' = coef(reg) + q.val  * std.err
)

# output for LaTeX
xtable(t1, digits = c(0,3,3,3,4,3,3))

##############################################################################################
# Q1.9b - nonparametric bootstrap

# define statistic to bootstrap
boot_stat <- function(df, i){
  reg <- glm(s ~ S_age + S_HHpeople + log_SincpcP1,
             data = df[i, ], family = binomial(link = 'logit'))
  return(reg$coefficients)
}

# run bootstrap, 999 replications
set.seed(22)
boot_results <- boot(data = df, R = 999, statistic = boot_stat)

# generate table of desired results
t2 <- matrix(NA, nrow=4, ncol=6) %>% as.data.frame
names(t2) <- c('Coefficient', 'StdError', 'tValue', 'pValue', 'CIlow', 'CIhigh')

t2$Coefficient <- boot_results$t0
t2$StdError    <- apply(boot_results$t, 2, sd)
t2$tValue      <- t2$Coefficient / t2$StdError

# define function to calculate pvalue from bootstrap results
getPval <- function(coef, coef_b){
  2 * max( mean( (coef_b-coef) >= abs(coef) ),
  mean( (coef_b-coef) <= -1*abs(coef) ) )
}

# apply over covariates, also get confidence intervals
for(i in 1:nrow(t2)){
  t2$pValue[i] <- getPval(boot_results$t0[i], boot_results$t[,i])

  t2$CIlow[i]  <- quantile(boot_results$t[,i], .025)
  t2$CIhigh[i] <- quantile(boot_results$t[,i], .975)
}

# add rownames
row.names(t2) <- row.names(t1)

# output for LaTeX
xtable(t2, digits = c(0,3,3,3,4,3,3))

##############################################################################################
# Q1.9c - propensity scores

# use original logistic regression
```

```
reg <- glm(s ~ S_age + S_HHpeople + log_SincpcP1,
           data = df, family = binomial(link = 'logit'))

# get data for use in plots
props <- reg$fitted.values %>% as.data.frame
names(props) <- 'fitted'

# plot with a variety of kernels
p1 <- ggplot(props, aes(x=fitted)) +
  geom_histogram(aes(y=..density..)) +
  geom_density(kernel = 'gaussian', size=1, aes(color='Gaussian')) +
  geom_density(kernel = 'epanechnikov', size=1, aes(color='Epanechnikov')) +
  geom_density(kernel = 'triangular', size=1, aes(color='Triangular')) +
  scale_color_manual(name = 'Kernel Densities',
                     values = c(Gaussian='red',
                                Epanechnikov='blue',
                                Triangular='green')) +
  theme_minimal() +
  labs(x='Fitted Probability of Observation', y='Density')
p1

ggsave('q1_9c_R.png')

##############################################################################
### Question 2: Semiparametric GMM with Missing Data
##############################################################################
# clean up
rm(list = ls())
gc()

# read in data
df <- read_csv('pisofirme.csv')

# add variable to indicate having outcome data
df %<>% mutate(s = 1-dmissing)

# add log variable for us in regression [log(S_incomepc + 1)]
df %<>% mutate(log_SincpcP1 = log(S_incomepc + 1))

##############################################################################
# Q2.2b - feasible estimator

# since we are using the logistic CDF as our F, we have simplified instruments
# here we construct the moment condition 0=E[g*m] which is just
# (instruments)*(y_i - F(t\theta+x\gamma))
# function of theta and gamma (parameters) and t and X (data)
g_mcar <- function(beta, data){
  theta <- beta[1]
  gamma <- beta[2:4]
  # constructing the F(t\theta+x\gamma) term
  F_component <- plogis(data$dpisofirme * theta +
                        data$S_age * gamma[1] +
                        data$S_HHpeople * gamma[2] +
                        data$log_SincpcP1 * gamma[3])
```

```
  vars <- c('dpisofirme', 'S_age', 'S_HHpeople', 'log_SincpcP1')
  Zvars <- paste0('Z_', vars)

  for(i in 1:length(vars)){
    data[,Zvars[i]] <- data[,vars[i]]*(data$danemia - F_component)
  }

  out <- data[,Zvars] %>% as.matrix

  return(out)
}

# bootstrap statistic
boot_stat <- function(dat, i){
  gmm(g_mcar, dat[i,], t0=c(0,0,0,0), wmatrix='ident', vcov='iid')$coef
}

# some incomplete values seem to be messing things up (drops three observations)
df <- df[complete.cases(df[,c('dpisofirme','S_age','S_HHpeople','log_SincpcP1')]),]

# run bootstrap, 999 replications
ptm <- proc.time()
set.seed(22)
boot_results <- boot(data=df[df$s==1, ], R=999, statistic = boot_stat, stype = "i")
proc.time() - ptm
# runtime is 24 minutes

# generate table of desired results (showing same results as in Q1)
t3 <- matrix(NA, nrow=4, ncol=6) %>% as.data.frame
names(t3) <- c('Coefficient', 'StdError', 'tValue', 'pValue', 'CIlow', 'CIhigh')

t3$Coefficient <- boot_results$t0
t3$StdError    <- apply(boot_results$t, 2, sd)
t3$tValue      <- t3$Coefficient / t3$StdError

# define function to calculate pvalue from bootstrap results
getPval <- function(coef, coef_b){
  2 * max( mean( (coef_b-coef) >= abs(coef) ),
  mean( (coef_b-coef) <= -1*abs(coef) ) )
}

# apply over covariates, also get confidence intervals
for(i in 1:nrow(t3)){
  t3$pValue[i] <- getPval(boot_results$t0[i], boot_results$t[,i])

  t3$CIlow[i]  <- quantile(boot_results$t[,i], .025)
  t3$CIhigh[i] <- quantile(boot_results$t[,i], .975)
}

# add rownames
row.names(t3) <- c('dpisofirme', 'S_age', 'S_HHpeople', 'log(S_incomepc+1)')

# output for LaTeX
```

```
xtable(t3, digits = c(0,3,3,3,4,3,3))

##############################################################################
# Q2.3c - feasible estimator

# similar to above, but first estimate propensity score then plug into gmm
# only difference in this function is to add (*ipw) (weighting observations)
g_mar <- function(beta, data){
  # restrict to nonmissing data
  data <- data[data$s==1,]

  # beta has two components
  theta <- beta[1]
  gamma <- beta[2:4]

  # constructing the F(t\theta+x\gamma) term
  F_component <- plogis(data$dpisofirme * theta +
                        data$S_age * gamma[1] +
                        data$S_HHpeople * gamma[2] +
                        data$log_SincpcP1 * gamma[3])

  vars <- c('dpisofirme', 'S_age', 'S_HHpeople', 'log_SincpcP1')
  Zvars <- paste0('Z_', vars)

  for(i in 1:length(vars)){
    data[,Zvars[i]] <- data[,vars[i]]*(data$danemia - F_component)*data$ipw
  }

  out <- data[,Zvars] %>% as.matrix

  return(out)
}

# bootstrap statistic
# this has more changes - need to estimate p using logit
# then plug that into our data as "ipw" and run gmm
boot_stat <- function(dat, i){
  boot_data <- dat[i,]

  # estimate p using logit
  reg <- glm(s ~ dpisofirme + S_age + S_HHpeople + log_SincpcP1 - 1,
             data = boot_data, family = binomial(link = 'logit'))

  # get fitted values
  predicted_p <- reg$fitted

  # construct inverse prob weights
  boot_data$ipw <- 1 / predicted_p

  # run GMM using the constructed weights
  gmm(g_mar, boot_data, t0=c(0,0,0,0), wmatrix='ident', vcov='iid')$coef
}

# some incomplete values seem to be messing things up (drops three observations)
```

```
df <- df[complete.cases(df[,c('dpisofirme','S_age','S_HHpeople','log_SincpcP1')]),]

# run bootstrap, 999 replications
ptm <- proc.time()
set.seed(22)
boot_results <- boot(data=df, R=999, statistic = boot_stat, stype = "i")
proc.time() - ptm
# runtime is 35mins

# generate table of desired results (showing same results as in Q1)
t4 <- matrix(NA, nrow=4, ncol=6) %>% as.data.frame
names(t4) <- c('Coefficient', 'StdError', 'tValue', 'pValue', 'CIlow', 'CIhigh')

t4$Coefficient <- boot_results$t0
t4$StdError    <- apply(boot_results$t, 2, sd)
t4$tValue      <- t4$Coefficient / t4$StdError

# define function to calculate pvalue from bootstrap results
getPval <- function(coef, coef_b){
  2 * max( mean( (coef_b-coef) >= abs(coef) ),
  mean( (coef_b-coef) <= -1*abs(coef) ) )
}

# apply over covariates, also get confidence intervals
for(i in 1:nrow(t4)){
  t4$pValue[i] <- getPval(boot_results$t0[i], boot_results$t[,i])

  t4$CIlow[i]  <- quantile(boot_results$t[,i], .025)
  t4$CIhigh[i] <- quantile(boot_results$t[,i], .975)
}

# add rownames
row.names(t4) <- c('dpisofirme', 'S_age', 'S_HHpeople', 'log(S_incomepc+1)')

# output for LaTeX
xtable(t4, digits = c(0,3,3,3,4,3,3))

##############################################################################################
# Q2.3d - feasible estimator, with trimming

# same as above, but trimming observations with p<.1 (1/p>10)
g_mar_trim <- function(beta, data){
  # restrict to nonmissing data
  data <- data[data$s==1,]

  # trim observations
  data <- data[data$ipw<=10,]

  # beta has two components
  theta <- beta[1]
  gamma <- beta[2:4]

  # constructing the F(t\theta+x\gamma) term
  F_component <- plogis(data$dpisofirme * theta +
```

```
                              data$S_age * gamma[1] +
                              data$S_HHpeople * gamma[2] +
                              data$log_SincpcP1 * gamma[3])

  vars <- c('dpisofirme', 'S_age', 'S_HHpeople', 'log_SincpcP1')
  Zvars <- paste0('Z_', vars)

  for(i in 1:length(vars)){
    data[,Zvars[i]] <- data[,vars[i]]*(data$danemia - F_component)*data$ipw
  }

  out <- data[,Zvars] %>% as.matrix

  return(out)
}

# bootstrap statistic
# only change is to use the g_mar_trim function instead of g_mar
boot_stat_trim <- function(dat, i){
  boot_data <- dat[i,]

  # estimate p using logit
  reg <- glm(s ~ dpisofirme + S_age + S_HHpeople + log_SincpcP1 - 1,
             data = boot_data, family = binomial(link = 'logit'))

  # get fitted values
  predicted_p <- reg$fitted

  # construct inverse prob weights
  boot_data$ipw <- 1 / predicted_p

  # run GMM using the constructed weights
  gmm(g_mar_trim, boot_data, t0=c(0,0,0,0), wmatrix='ident', vcov='iid')$coef
}

# run bootstrap, 999 replications
ptm <- proc.time()
set.seed(22)
boot_results <- boot(data=df, R=999, statistic = boot_stat_trim, stype = "i")
proc.time() - ptm
# runtime is 39mins

# generate table of desired results (showing same results as in Q1)
t5 <- matrix(NA, nrow=4, ncol=6) %>% as.data.frame
names(t5) <- c('Coefficient', 'StdError', 'tValue', 'pValue', 'CIlow', 'CIhigh')

t5$Coefficient <- boot_results$t0
t5$StdError    <- apply(boot_results$t, 2, sd)
t5$tValue      <- t5$Coefficient / t5$StdError

# apply over covariates, also get confidence intervals
for(i in 1:nrow(t5)){
  t5$pValue[i] <- getPval(boot_results$t0[i], boot_results$t[,i])
```

```
  t5$CIlow[i] <- quantile(boot_results$t[,i], .025)
  t5$CIhigh[i] <- quantile(boot_results$t[,i], .975)
}

# add rownames
row.names(t5) <- c('dpisofirme', 'S_age', 'S_HHpeople', 'log(S_incomepc+1)')

# output for LaTeX
xtable(t5, digits = c(0,3,3,3,4,3,3))

##############################################################################
### Question 3: When Bootstrap Fails
##############################################################################
# clean up
rm(list = ls())
gc()

##############################################################################
# Q3.1 - nonparametric bootstrap

# sample size
n <- 1000

# generate sample
set.seed(123)
x <- runif(n,0,1)

# replications to run
reps <- 599

# empty vector to fill with bootstrap results
np_stats <- rep(NA,reps)

# nonparametric bootstrap
for(i in 1:reps){
  # pick 1000 times from x, with sampling
  x_star <- sample(x, n, replace = T)

  # calculate the statistic as specified
  stat <- n * (max(x) - max(x_star))

  # save stat and move to next i
  np_stats[i] <- stat
}

# data for plot
np_df <- data.frame(statistic_value = np_stats)

# plot our bootstrapped stats vs. exp(1) distribution
p3.1 <- ggplot(np_df) +
  geom_histogram(aes(x = statistic_value, y=..density..)) +
  stat_function(fun = function(x) dexp(x), size = 1, color = 'red') +
  theme_minimal()
p3.1
```

```
ggsave('q3_1_R.png')

##############################################################################
# Q3.2 - parametric bootstrap

# use same x as before

# empty vector to fill with bootstrap results
p_stats <- rep(NA, reps)

# parametric bootstrap
for(i in 1:reps){
  # generate sample of 1000 from uniform(0,max(x))
  x_star <- runif(n, 0, max(x))

  # calculate the statistic as specified
  stat <- n * (max(x) - max(x_star))

  # save stat and move to next i
  p_stats[i] <- stat
}

# data for plot
p_df <- data.frame(statistic_value = p_stats)

# plot our bootstrapped stats vs. exp(1) distribution
p3.2 <- ggplot(p_df) +
  geom_histogram(aes(x = statistic_value, y=..density..)) +
  stat_function(fun = function(x) dexp(x), size = 1, color = 'red') +
  theme_minimal()
p3.2

ggsave('q3_2_R.png')

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

B Stata Code

```
*******************************************************************************
* Author: Paul R. Organ
* Purpose: ECON 675, PS3
* Last Update: Oct 26, 2018
*******************************************************************************
clear all
set more off
capture log close

cd "C:\Users\prorgan\Box\Classes\Econ 675\Problem Sets\PS3"
log using ps3.log, replace

*******************************************************************************
*** Question 1: Non-linear Least Squares
*******************************************************************************
* Q1.9 setup
* load data
use pisofirme, clear

* add variable indicating having outcome data
gen s = 1-dmissing

* add log variable for use in regression [log(S_incomepc + 1)]
gen log_SincpcP1 = log(S_incomepc + 1)

* Q1.9a - estimates and statistics
* logit regression, robust standard errors
logit s S_age S_HHpeople log_SincpcP1, vce(robust)

* output for LaTeX
outreg2 using q1_9a.tex, side stats(coef se tstat pval ci) ///
 noaster noparen nor2 noobs dec(3) replace

* Q1.9b - nonparametric bootstrap
logit s S_age S_HHpeople log_SincpcP1, vce(bootstrap, reps(999))

* output for LaTeX
outreg2 using q1_9b.tex, side stats(coef se tstat pval ci) ///
 noaster noparen nor2 noobs dec(3) replace

* Q1.9c - propensity scores
* logit regression, robust standard errors
logit s S_age S_HHpeople log_SincpcP1, vce(robust)

* predict propensity score
predict p

* plot histogram, overlay kernel density
twoway histogram p || kdensity p, k(gaussian) || ///
 kdensity p, k(epanechnikov) || kdensity p, k(triangle) ///
 leg(lab(1 "Propensity Score") lab(2 "Gaussian") ///
        lab(3 "Epanechnikov") lab(4 "Triangle"))
```

```
* save
graph export q1_9c_S.png, replace

*******************************************************************************
*** Question 2: Semiparametric GMM with Missing Data
*******************************************************************************
* Q2.2b(MCAR)- feasible estimator

* gmm, four moment conditions
local vars = "dpisofirme S_age S_HHpeople log_SincpcP1"
gmm ((danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+S_HHpeople*{gamma2}+
log_SincpcP1*{gamma3})))*dpisofirme) ///
((danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+S_HHpeople*{gamma2}+
log_SincpcP1*{gamma3})))*S_age) ///
((danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+S_HHpeople*{gamma2}+
log_SincpcP1*{gamma3})))*S_HHpeople) ///
((danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+S_HHpeople*{gamma2}+
log_SincpcP1*{gamma3})))*log_SincpcP1), ///
instruments(`vars') winitial(identity) vce(boot)

* output for LaTeX
 mata:
        coef = st_matrix("e(b)")'
        se = st_matrix("e(se)")'

        tstat = coef:/se

        CI_low = coef - 1.96:*se
        CI_high = coef + 1.96:*se

        stats = round((coef,se,tstat,CI_low,CI_high),.001)

        st_matrix("stats",stats)
end
mat rownames stats = `vars'
mat colnames stats = coef se tstat CI_low CI_high
outtable using q2_2b, mat(stats) replace nobox

* Q2.3c (MAR)- feasible estimator
* we predicted p before, but did not use t, so do that now:
* logit regression, robust standard errors
logit s dpisofirme S_age S_HHpeople log_SincpcP1, vce(robust)

* predict propensity score
predict p_witht

* now run gmm adding in new term s/p
local vars = "dpisofirme S_age S_HHpeople log_SincpcP1"
gmm ((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3})))*dpisofirme) ///
((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3})))*S_age) ///
((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
```

```
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3}))))*S_HHpeople) ///
((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3}))))*log_SincpcP1), ///
instruments('vars') winitial(identity) vce(boot)

* output for LaTeX
 mata:
        coef = st_matrix("e(b)")'
        se = st_matrix("e(se)")'

        tstat = coef:/se

        CI_low = coef - 1.96:*se
        CI_high = coef + 1.96:*se

        stats = round((coef,se,tstat,CI_low,CI_high),.001)

        st_matrix("stats",stats)
end
mat rownames stats = 'vars'
mat colnames stats = coef se tstat CI_low CI_high
outtable using q2_3c, mat(stats) replace nobox

* Q2.3d (MAR)- feasible estimator, trimmed
* we predicted p before, and have s, so add that before the moment conditions
local vars = "dpisofirme S_age S_HHpeople log_SincpcP1"
gmm ((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3}))))*dpisofirme) ///
((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3}))))*S_age) ///
((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3}))))*S_HHpeople) ///
((s/p_witht)*(danemia - invlogit((dpisofirme*{theta}+S_age*{gamma1}+
S_HHpeople*{gamma2}+log_SincpcP1*{gamma3}))))*log_SincpcP1) ///
if p_witht >= 0.1, instruments('vars') winitial(identity) vce(boot)

* output for LaTeX
 mata:
        coef = st_matrix("e(b)")'
        se = st_matrix("e(se)")'

        tstat = coef:/se

        CI_low = coef - 1.96:*se
        CI_high = coef + 1.96:*se

        stats = round((coef,se,tstat,CI_low,CI_high),.001)

        st_matrix("stats",stats)
end
mat rownames stats = 'vars'
mat colnames stats = coef se tstat CI_low CI_high
outtable using q2_3d, mat(stats) replace nobox
```

```
*******************************************************************************
*** Question 3: When Bootstrap Fails
*******************************************************************************
* Q3.1 - nonparametric bootstrap
clear all

* generate sample
set seed 123
set obs 1000
gen X = runiform()

* save actual max
sum X
local maxX=r(max)

* run nonparametric bootstrap of max
bootstrap stat=r(max), reps(599) saving(nonpar_results, replace): summarize X

* load results
use nonpar_results, clear

* generate statistic
gen nonpar_stat = 1000*(`maxX'-stat)

* plot
hist nonpar_stat, ///
 plot(function exponential = 1-exponential(1,x), range(0 5) color(red))
graph export q3_1_S.png, replace

*******************************************************************************
* Q3.2 - parametric bootstrap
clear all

tempname memhold
tempfile para_results

* generate sample
set seed 123
set obs 1000
gen X = runiform()

* save actual max
sum X
local maxX=r(max)

* parametric bootstrap
postfile `memhold' max using `para_results'
forvalues i = 1/599{
        capture drop sample
        gen sample = runiform(0,`maxX')
        sum sample
        post `memhold' (r(max))
}
postclose `memhold'
```

```
* load results
use `para_results', clear

* generate statistic
gen para_stat = 1000*(`maxX'-max)

* plot
hist para_stat, ///
 plot(function exponential = 1-exponential(1,x), range(0 5) color(red))
graph export q3_2_S.png, replace

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