Homework 5 Solutions

Question 1

Part (a)

To start with recall that we are going to estimate the parameters in the probit model by solving the following optimization problem:

$$\hat{\beta} = \underset{b}{\text{argmax}} \frac{1}{n} \sum_{i=1}^{n} Y_i \log(\Phi(X_i'b)) + (1 - Y_i) \log(1 - \Phi(X_i'b))$$

and where it is also helpful to recall that we defined the score as the derivative of this objective function taken with respect to the parameters:

$$S_n(b) = \frac{1}{n} \sum_{i=1}^{n} \frac{(Y_i - \Phi(X_i'b))\phi(X_i'b)}{\Phi(X_i'b)(1 - \Phi(X_i'b))} X_i$$

```
library(haven)
data <- read_dta("cps09mar.dta")
data <- subset(data, female==0)
data$black <- 1*data$race==2
Y <- data$union
X <- as.matrix(data[,c("age","education","black","hisp")])
X <- cbind(1,X)
n <- nrow(data)

# log-likelihood as function of parameters
ll <- function(b, X, Y) {
  G <- pnorm(X%*%b)
  mean( Y*log(G) + (1-Y)*log(1-G) )
}

# score function
s <- function(b, X, Y) {
  G <- pnorm(X%*%b)
  g <- dnorm(X%*%b)

  # calculates mean across units (returning k-dim vector)
  apply(as.numeric(Y*(g/G))*X - as.numeric((1-Y)*(g/(1-G)))*X, 2, mean)
}

k <- ncol(X)
start_bet <- rep(0,k)
prob_est <- optim(start_bet, ll, gr=s,
                  X=X, Y=Y,
                  method="BFGS",
                  control=list(fnscale=-1))
bet <- prob_est$par
```

In order to calculate standard errors, recall that we showed in class that

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} N(0, \mathbf{\Omega}^{-1})$$

where $\mathbf{\Omega} = \mathbb{E}\left[\frac{\phi(X'\beta)^2}{\Phi(X'\beta)(1-\Phi(X'\beta))} XX'\right]$, and that we could estimate $\mathbf{\Omega}$ by

$$\hat{\mathbf{\Omega}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\phi(X_i'\hat{\beta})^2}{\Phi(X_i'\hat{\beta})(1-\Phi(X_i'\hat{\beta}))} X_i X_i'$$

```
idx <- as.numeric(X%*%bet)
O1 <- ( dnorm(idx)^2 / ( pnorm(idx)*(1-pnorm(idx)) ) ) * X
Omeg <- t(O1)%*%X/n
Omeg_inv <- solve(Omeg)
se <- sqrt(diag(Omeg_inv))/sqrt(n)
round(cbind.data.frame(bet=bet, se=se, t=bet/se), 6)
```

```
                bet       se          t
          -1.892941 0.106813 -17.721951
age        0.007384 0.001416   5.215583
education -0.028084 0.006212  -4.521237
black     -0.063187 0.060269  -1.048417
hisp      -0.289155 0.056130  -5.151499
```

Finally, let's compare these results to the ones that we get from R's `glm` command

```
# compare to results from R
R_probit <- glm(union ~ age + education + black + hisp,
                family=binomial(link=probit), data=data)
summary(R_probit)
```

```
Call:
glm(formula = union ~ age + education + black + hisp, family = binomial(link = probit),
    data = data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.953818   0.107176 -18.230  < 2e-16 ***
age          0.007925   0.001419   5.584 2.35e-08 ***
education   -0.025505   0.006221  -4.100 4.14e-05 ***
blackTRUE   -0.054083   0.060004  -0.901    0.367
hisp        -0.297745   0.056865  -5.236 1.64e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 6282.0  on 29139  degrees of freedom
Residual deviance: 6210.7  on 29135  degrees of freedom
AIC: 6220.7

Number of Fisher Scoring iterations: 6
```

These are slightly different from each other. I checked the documentation for `glm` and it uses an "iteratively reweighted least squares" estimation procedure; this is different from the optimization procedure that I used and explains the difference between the estimates.

Part (b)

To start with, let's compute estimates of average partial effects. Given what we have already done, this is fairly easy.

```
# compute average partial effects
pe <- dnorm(X%*%bet) %*% t(bet)
ape <- apply(pe, 2, mean)
```

Part (c)

Next, let's derive the limiting distribution of the the average partial effects and use this to compute standard errors.

Starting from the hint in the problem

$$\begin{aligned}
\sqrt{n}(\widehat{APE} - APE) &= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(X_i'\hat{\beta})\hat{\beta} - \mathbb{E}[\phi(X'\beta)\beta]\right) \\
&= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(X_i'\hat{\beta})\hat{\beta} - \frac{1}{n}\sum_{i=1}^{n}\phi(X_i'\hat{\beta})\beta\right) \qquad \text{(A)} \\
&+ \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(X_i'\hat{\beta})\beta - \frac{1}{n}\sum_{i=1}^{n}\phi(X_i'\beta)\beta\right) \qquad \text{(B)} \\
&+ \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(X_i'\beta)\beta - \mathbb{E}[\phi(X'\beta)\beta]\right) \qquad \text{(C)}
\end{aligned}$$

It is helpful to recall from class that we defined

$$\begin{aligned}
\mathbf{Q} &= -\mathbb{E}\left[\frac{\phi(X'\beta)^2}{\Phi(X'\beta)(1-\Phi(X'\beta))}XX'\right] \\
\psi(Y,X,b) &= -\frac{(Y-\Phi(X'b))\phi(X'b)}{\Phi(X'b)(1-\Phi(X'b))}X
\end{aligned}$$

and we showed that

$$\sqrt{n}(\hat{\beta}-\beta) = -\mathbf{Q}^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\psi(Y_i,X_i,\beta) + o_p(1)$$

which was the intermediate step before we applied the CLT.

Now, let's consider each term in turn in the expression for $\sqrt{n}(\widehat{APE} - APE)$, starting with the first one.

$$\begin{aligned}
(A) &= \frac{1}{n}\sum_{i=1}^{n} \phi(X_i'\hat{\beta})\sqrt{n}(\hat{\beta} - \beta) \\
&= \mathbb{E}[\phi(X'\beta)]\sqrt{n}(\hat{\beta} - \beta) + o_p(1) \\
&= \mathbb{E}[\phi(X'\beta)]\mathbf{Q}^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n} \psi(Y_i, X_i, \beta) + o_p(1) \\
&= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} \mathbb{E}[\phi(X'\beta)]\mathbf{Q}^{-1}\psi(Y_i, X_i, \beta) + o_p(1) \\
&:= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} A_i + o_p(1)
\end{aligned}$$

where the second equality holds by the law of large numbers and CMT, the third line holds by what we showed in class for $\sqrt{n}(\hat{\beta} - \beta)$, the fourth line just rearranges in a way that will be convenient below, and the fifth equality just introduces a more concise notation.

Next, consider the second term in the hint (this is hardest term to deal with). Notice that we can write

$$\begin{aligned}
(B) &= \beta\frac{1}{\sqrt{n}}\sum_{i=1}^{n} \phi(X_i'\hat{\beta}) - \phi(X_i'\beta) \\
&= \beta\frac{1}{n}\sum_{i=1}^{n} \phi'(X_i'\beta)X_i'\sqrt{n}(\hat{\beta} - \beta) + o_p(1) \\
&= -\beta\frac{1}{n}\sum_{i=1}^{n} X_i'\beta\phi(X_i'\beta)X_i'\sqrt{n}(\hat{\beta} - \beta) + o_p(1) \\
&= -\beta\mathbb{E}[X'\beta\phi(X'\beta)X']\sqrt{n}(\hat{\beta} - \beta) + o_p(1) \\
&= \beta\mathbb{E}[X'\beta\phi(X'\beta)X']\mathbf{Q}^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n} \psi(Y_i, X_i, \beta) + o_p(1) \\
&= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} \beta\mathbb{E}[X'\beta\phi(X'\beta)X']\mathbf{Q}^{-1}\psi(Y_i, X_i, \beta) + o_p(1) \\
&:= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} B_i + o_p(1)
\end{aligned}$$

where the first equality holds just by re-arranging, the second equality holds using delta method / mean value theorem type of argument, the third equality holds because $\phi'(z) = -z\phi(z)$ (this is a property of a standard normal distribution), the fourth equality holds by the law of large numbers and CMT, the fifth equality holds by what we showed in class for the asymptotically linear representation of $\sqrt{n}(\hat{\beta} - \beta)$, the sixth equality just re-arranges by pushing inside the sum the expectation terms, and the last line defines B_i.

Finally, the third term is immediately equal to

$$\begin{aligned}(C) &= \frac{1}{\sqrt{n}}\sum_{i=1}^{\beta}(\phi(X_i'\beta) - \mathbb{E}[\phi(X'\beta)]) \\ &:= \frac{1}{\sqrt{n}}\sum_{i=1}^{n} C_i\end{aligned}$$

Thus, we can write

$$\sqrt{n}(\widehat{APE} - APE) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}(A_i + B_i + C_i) + o_p(1)$$

If you plug in the expressions for A_i, B_i, C_i, this expression will look very complicated, but it is mean 0, and we can apply the CLT to it. In fact, it immediately follows that

$$\sqrt{n}(\widehat{APE} - APE) \xrightarrow{d} N(0, \mathbf{V})$$

where

$$\mathbf{V} = \mathbb{E}[(A + B + C)(A + B + C)']$$

Estimating $\mathbf{V}$ just involves plugging in sample quantities for population quantities — again: these expressions will be long, but if you do it carefully, everything should work. This is what we will do next.

Part (d)

```
# compute standard errors
idx <- as.numeric(X%*%bet) #nx1
Q <- -Omeg # kxk
G <- pnorm(X%*%bet) # nx1
g <- dnorm(X%*%bet) # nx1
# psi is nxk matrix
psi <- -as.numeric( Y*(g/G))*X - as.numeric((1-Y)*(g/(1-G)))*X

# compute A
a1 <- mean(g)
A <- a1*psi%*%solve(Q)

# compute B
b1 <- t(apply(as.numeric(idx * g) * X, 2, mean))
B <- t(as.matrix(bet)%*%b1%*%solve(Q)%*%t(psi))

# compute C
C <- as.numeric(g - mean(g))%*%t(bet)

# compute variance
```

```
inf_func <- A + B + C
# estimate variance
ape_V <- t(inf_func)%*%inf_func/n
ape_se <- sqrt(diag(ape_V))/sqrt(n)
round(cbind.data.frame(ape, ape_se, t=(ape/ape_se)),4)
```

```
               ape ape_se        t
          -0.1009 0.0065 -15.5874
age        0.0004 0.0001   5.4706
education -0.0015 0.0003  -4.9468
black     -0.0034 0.0033  -1.0344
hisp      -0.0154 0.0032  -4.8905
```

Let's compare these to what you get if you use R to compute average partial effects.

```
library(marginaleffects)
avg_slopes(R_probit)
```

```
      Term     Contrast  Estimate Std. Error      z Pr(>|z|)    S     2.5 %
 age       dY/dX          0.000421   7.63e-05  5.511   <0.001 24.7  0.000271
 black     TRUE - FALSE -0.002746   2.91e-03 -0.943    0.346  1.5 -0.008454
 education dY/dX        -0.001354   3.33e-04 -4.070   <0.001 14.4 -0.002006
 hisp      1 - 0        -0.012818   1.97e-03 -6.512   <0.001 33.7 -0.016676
    97.5 %
  0.000570
  0.002961
 -0.000702
 -0.008960

Type:  response
```

As before, we're getting slightly different results. This is expected though; recall, that our original probit estimates were slightly different from the ones coming from R which would suggest that we'd expect slightly different APEs. That said, these are reasonable close and suggest that we do not have a coding error or anything like that.

Part (e)

This is quite similar to what we have done for the bootstrap before. The code below uses parallel processing to speed up computation using the `pbapply` package. The bootstrap is an example of what's sometimes called an "embarrassingly parallel" problem – this is kind of a strange name, but it just means that it's an obvious place to use parallel processing. The reason is that each bootstrap iteration is fully independent of other bootstrap iterations, so you can run lots of these at the same time and then compute standard errors (or whatever you want) after you have carried out all of the bootstrap iterations.

```r
# finally, compute standard errors using the bootstrap
biters <- 100
library(pbapply) # for computing in parallel
boot_res <- pblapply(1:biters, function(b) {

  # draw new data with replacement
  boot_rows <- sample(1:n, size=n, replace=TRUE)
  boot_data <- data[boot_rows,]
  boot.Y <- boot_data$union
  boot.X <- as.matrix(data[,c("age","education","black","hisp")])
  boot.X <- cbind(1,boot.X)

  # estimate probit using new data
  boot_est <- optim(start_bet, ll, gr=s,
                    X=boot.X,
                    Y=boot.Y,
                    method="BFGS",
                    control=list(fnscale=-1))
  boot_bet <- boot_est$par

  # compute average partial effects
  boot_pe <- dnorm(boot.X%*%boot_bet) %*% t(boot_bet)
  boot_ape <- apply(boot_pe, 2, mean)

  # return results
  boot_ape
}, cl=2)

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# compare to earlier standard errors
round(cbind.data.frame(ape=ape, ape_se=ape_se, boot_se=boot_se),4)
```

```
               ape ape_se boot_se
          -0.1009 0.0065  0.0070
age        0.0004 0.0001  0.0001
education -0.0015 0.0003  0.0004
black     -0.0034 0.0033  0.0031
hisp      -0.0154 0.0032  0.0025
```

These standard errors are very similar to the ones we calculated using asymptotic theory. Also, notice that, for me, it takes about a minute to compute the bootstrap standard errors, but only a second or two to compute the asymptotic standard errors. However, it only took 5 or 10 minutes for me to write the bootstrap code while it took me close to two hours to figure out the limiting

distribution of the average partial effects and write the code for them (these are also probably more prone to making mistakes here because the arguments are more complicated).

Question 2

In this question, we will estimate the ATT of a job training program using a number of different techniques that we discussed in class.

One thing to note: for regression, regression adjustment, propensity score re-weighting, and AIPW, your answer should be exactly the same as mine, but we may be get different estimates when we use machine learning, due to sample splitting, choosing tuning parameters, etc. The same is true for the bootstrap standard errors—they should be broadly similar, but we would not expect that mine and yours will be literally the same number.

Part (a)

To estimate the ATT using regression adjustment, we first need to calculate $\hat{\beta}$ from the regression of Y on X using untreated observations only. Once we have this estimate, we can compute

$$\widehat{ATT} = \frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i'\hat{\beta}$$

```
# load packages used later
library(pbapply)
library(ranger)

# load data
data <- as.data.frame(haven::read_dta("jtrain_observational.dta"))
Y <- data$re78
D <- data$train
p <- mean(D)
n <- nrow(data)
X <- model.matrix(~age + educ + black + hisp + married + re75 + unem75, data=data)
# run regression using untreated observations only
bet <- solve(t(X)%*%(X*as.numeric((1-D)/p)))%*%t(X)%*%as.numeric(Y*(1-D)/p)

att1 <- mean(D*Y/p)
att2 <- sum(apply(D*X/p,2,mean)*as.numeric(bet))
att <- att1 - att2

# report estimate of att
round(att,3)
```

```
[1] 0.859
```

The outcome is in 1000's of dollars, so this indicates that we are estimating that job training increased yearly earnings by $859.

As a side-comment, it's not immediately clear if this should be interpreted as a large effect or not. One way to think about this is to compute: $ATT/\mathbb{E}[Y(0)|D=1]$ (i.e., the relative size of

the ATT compared to what the average outcome would have been absent the treatment). Further, notice that this is equal to $ATT/(\mathbb{E}[Y|D=1] - ATT)$ (which holds by adding and subtracting $\mathbb{E}[Y(1)|D=1]$ in the denominator). If we compute this, we get that we have estimated that yearly earnings as about 16% higher from job training relative to what they would have been in the absence of job training.

Part (b)

Notice that

$$\begin{aligned}
\sqrt{n}(\widehat{ATT} - ATT) &= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i'\hat{\beta}\right) - \sqrt{n}\left(\mathbb{E}\left[\frac{D}{p}Y\right] - \mathbb{E}\left[\frac{D}{p}X'\right]\beta\right) \\
&= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \mathbb{E}\left[\frac{D}{p}Y\right]\right) - \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i' - \mathbb{E}\left[\frac{D}{p}X'\right]\right)\hat{\beta} \\
&\qquad - \mathbb{E}\left[\frac{D}{p}X'\right]\sqrt{n}(\hat{\beta} - \beta) \\
&= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \mathbb{E}\left[\frac{D}{p}Y\right]\right) - \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i' - \mathbb{E}\left[\frac{D}{p}X'\right]\right)\beta \\
&\qquad - \mathbb{E}\left[\frac{D}{p}X'\right]\sqrt{n}(\hat{\beta} - \beta) + o_p(1)
\end{aligned}$$

where the first line holds by definition, the second line adds and subtracts $\mathbb{E}[(D/p)X']\hat{\beta}$, and the third equality holds because $\hat{\beta} \xrightarrow{p} \beta$ (and by the CMT). Recalling that,

$$\sqrt{n}(\hat{\beta} - \beta) = \mathbb{E}\left[\frac{(1-D)}{(1-p)}XX'\right]^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\frac{(1-D_i)}{(1-p)}X_i e_i + o_p(1)$$

we have that

$$\sqrt{n}(\widehat{ATT} - ATT) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}(A_i - B_i - C_i) + o_p(1)$$

where

$$\begin{aligned}
A_i &= \frac{D_i}{p}Y_i - \mathbb{E}\left[\frac{D}{p}Y\right] \\
B_i &= \left(\frac{D_i}{p}X_i' - \mathbb{E}\left[\frac{D}{p}X'\right]\right)\beta \\
C_i &= \mathbb{E}\left[\frac{D}{p}X'\right]\mathbb{E}\left[\frac{(1-D)}{(1-p)}XX'\right]^{-1}\frac{(1-D_i)}{(1-p)}X_i e_i
\end{aligned}$$

Thus, $\sqrt{n}(\widehat{ATT} - ATT) \xrightarrow{d} N(0, V)$ where $V = \mathbb{E}[(A-B-C)^2]$ (we can square here since ATT is a scalar). We can consistently estimate V by replacing all of the population averages by sample averages and replacing β with its consistent estimate $\hat{\beta}$.

```
A2 <- mean(D/p*Y)
Ai <- D/p*Y - A2

XDp <- X*as.numeric(D/p)
B2 <- colMeans(XDp)
# sweep subtracts a vector from each row of a matrix
Bi <- sweep(XDp, 2, B2) %*% bet

C2 <- t(B2)
XUp <- X*as.numeric((1-D)/(1-p))
C3 <- t(X)%*%XUp/n
ehat <- Y - X%*%bet
XUpe <- XUp*as.numeric(ehat)
Ci <- as.numeric(C2%*%solve(C3)%*%t(XUpe))

V <- mean( (Ai-Bi-Ci)^2 )
se <- sqrt(V)/sqrt(n)
round(se,3)
```

```
[1] 0.903
```

This indicates that we cannot reject that job training had no effect earnings at conventional significance levels.

Part (c)

Let's move to computing standard errors using the bootstrap. Towards, this end let's write a function that takes in some data and computes an estimate of ATT (this is essentially just the same code that we used before).

```
compute.att <- function(data) {
  Y <- data$re78
  D <- data$train
  p <- mean(D)
  X <- model.matrix(~age + educ + black + hisp + married + re75 + unem75, data=data)
  # run regression using untreated observations only
  bet <- solve(t(X)%*%(X*as.numeric((1-D)/p)))%*%t(X)%*%as.numeric(Y*(1-D)/p)

  att1 <- mean(D*Y/p)
  att2 <- sum(apply(D*X/p,2,mean)*as.numeric(bet))
  att <- att1 - att2
  att
}
```

There is a subtle issue about whether we should treat p as being known or estimated. Above I treated it like it was known. And, for this reason, I am going to draw bootstrap samples from the treated group and untreated group separately (it is not a big deal if you didn't do this though, just noting it so you can understand the code).

```
# now bootstrap
biters <- 1000
treated_data <- subset(data, train==1)
untreated_data <- subset(data, train==0)
n1 <- nrow(treated_data)
n0 <- nrow(untreated_data)
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att(boot_data)
})

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

round(boot_se, 3)
```

```
[1] 0.918
```

These standard errors are similar to the ones we computed before.

Part (d)

For this part, we are just going to run a regression of Y on D and X.

```
X2 <- cbind(X,D)
bet2 <- solve(t(X2)%*%X2)%*%t(X2)%*%Y
round(bet2,3)
```

```
              [,1]
(Intercept) -0.061
age         -0.057
educ         0.604
black       -0.597
hisp         2.547
```

```
married      1.530
re75         0.788
unem75      -0.079
D            0.525
```

```
ehat <- Y - X2%*%bet2
X2e <- X2*as.numeric(ehat)
Omeg2 <- t(X2e)%*%X2e/n
Q2 <- t(X2)%*%X2/n
V2 <- solve(Q2)%*%Omeg2%*%solve(Q2)
se2 <- sqrt(diag(V2))/sqrt(n)
round(se2,3)
```

```
(Intercept)         age        educ       black        hisp     married
      1.588       0.025       0.096       0.462       1.271       0.517
       re75      unem75           D
      0.036       0.967       0.884
```

The estimated coefficient on D is somewhat closer to 0 than we computed in the first part while the standard errors are about the same. In some sense, this doesn't appear to matter much, because in both cases we are estimating a small positive (and not statistically significant effect) of job training. However, this is mostly a result of us not being able to precisely estimate effects of job training. That said, our earlier point estimate is about 64% larger than the one from the regression which is arguably meaningfully different even though the identifying assumptions are the same.

Part (e)

Next, we will estimate the ATT using propensity score re-weighting. Since we will use the bootstrap to compute standard errors, let's write a function that computes an estimate of the ATT given some data—we'll use this function to estimate the ATT itself and inside our bootstrap iterations.

```
compute.att_ipw <- function(data) {
  Y <- data$re78
  D <- data$train
  p <- mean(D)

  logit_est <- glm(train ~ age + educ + black + hisp + married + re75 + unem75,
                   data=data, family=binomial(link="logit"))
  pscore <- predict(logit_est, type="response")
  att <- mean( ( (D/p) - (1-D)/p*pscore/(1-pscore) ) * Y )
  att
}
att_ipw <- compute.att_ipw(data)
```

and now code to compute bootstrapped standard errors

```
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att_ipw(boot_data)
})

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# report results
data.frame(att_ipw=round(att_ipw,3), se=round(boot_se, 3))
```

```
  att_ipw    se
1   0.458 1.091
```

This estimate is somewhat lower than regression adjustment and the bootstrap standard errors have a similar magnitude.

Part (f)

Next, we will estimate the *ATT* using the AIPW/doubly robust approach discussed in class.

```
compute.att_dr <- function(data) {
  Y <- data$re78
  D <- data$train
  p <- mean(D)
  X <- model.matrix(~age + educ + black + hisp + married + re75 + unem75, data=data)
  bet <- solve(t(X)%*%(X*as.numeric((1-D)/p)))%*%t(X)%*%as.numeric(Y*(1-D)/p)
  m <- X%*%bet

  logit_est <- glm(train ~ age + educ + black + hisp + married + re75 + unem75,
                   data=data, family=binomial(link="logit"))
  pscore <- predict(logit_est, type="response")
  att <- mean( ( (D/p) - (1-D)/p*pscore/(1-pscore) ) * ( Y - m ) )
  att
```

```
}
att_dr <- compute.att_dr(data)
```

and now code to compute bootstrapped standard errors

```
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att_dr(boot_data)
})

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# report results
data.frame(att_dr=round(att_dr,3), se=round(boot_se, 3))
```

```
  att_dr    se
1   0.62 1.054
```

Here the estimate is in between regression adjustment and propensity score re-weighting. The bootstrap standard errors are similar to the ones we computed before.

Part (g)

Next, we will estimate the ATT using machine learning.

```
set.seed(1234)

compute.att_ml <- function(data) {

  # create two folds
  data$fold <- sample(1:2, n, replace=TRUE)
  fold1 <- subset(data, fold==1)
  fold2 <- subset(data, fold==2)
```

```
  # inner function to compute an att for f2 using f1 to estimate the
  # preliminary models
  ml_att <- function(f1, f2) {
    # use f1 to estimate the first step models
    Dmod <- ranger(as.factor(train) ~ age + educ + black + hisp +
                                      married + re75 + unem75,
                   data=f1)
    Ymod <- ranger(re78 ~ age + educ + black + hisp +
                          married + re75 + unem75,
                   data=f1)

    # get predictions with f2
    pscore <- predict(Dmod, data=f2, probability=TRUE)$predictions
    out_reg <- predict(Ymod, data=f2)$predictions

    # compute att(k) with f2
    D <- f2$train
    p <- mean(D)
    Y <- f2$re78
    att1 <- mean(D/p*(Y-out_reg))
    att2 <- mean((1-D)/p * pscore/(1-pscore) * (Y-out_reg) )
    att <- att1-att2
    att
  }

  # cross splitting
  ml1 <- ml_att(fold1,fold2)
  # reverse roles
  ml2 <- ml_att(fold2,fold1)
  # average
  mean(c(ml1,ml2))
}

att_ml <- compute.att_ml(data)
```

and now code to compute bootstrapped standard errors

```
# note: the bootstrap takes longer here compared to other approaches
# below I used parallel processing to speed things up
# but it is also ok to decrease the number of bootstrap iterations (or
# to just wait a while longer...)
biters <- 1000
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
```

```
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att_ml(boot_data)
}, cl=10)

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# report results
data.frame(att_ml=round(att_ml,3), se=round(boot_se, 3))
```

```
  att_ml    se
1  1.092 1.285
```

This is somewhat larger than our earlier estimates though the standard errors are also somewhat larger.

Question 3

The starting point for this question is what we called Decomposition 2 in class:

$$\begin{aligned}\alpha = \mathbb{E}\Big[w(D,X)\Big(\mathbb{E}[Y|X,D=1] - \mathbb{E}[Y|X,D=0]\Big)\Big] \\ + \mathbb{E}[w(D,X)\Big(\mathbb{E}[Y|X,D=0] - \mathrm{L}_0(Y|X)\Big)]\end{aligned}$$

where

$$w(D,X) = \frac{D(1-\mathrm{L}(D|X))}{\mathbb{E}[(D-\mathrm{L}(D|X))^2]}$$

Under unconfoundedness, $\mathbb{E}[Y|X,D=1] - \mathbb{E}[Y|X,D=0] = CATE(X)$, so we mainly need to show that the second term is equal to 0 if either of the conditions in the problem hold. Condition (ii), that $\mathbb{E}[Y|X,D=0] = \mathrm{L}_0(Y|X)$, immediately implies that it is equal to 0. For condition (i), that

$p(X) = \mathrm{L}(D|X)$, ignoring the denominator of the weights, we have that

$$\begin{aligned}
&\mathbb{E}\Big[D(1-\mathrm{L}(D|X))\Big(\mathbb{E}[Y|X,D=0]-\mathrm{L}_0(Y|X)\Big)\Big] \\
&\qquad = \mathbb{E}\Big[D(1-p(X))\Big(\mathbb{E}[Y|X,D=0]-\mathrm{L}_0(Y|X)\Big)\Big] \\
&\qquad = \mathbb{E}\Big[p(X)(1-p(X))\Big(\mathbb{E}[Y|X,D=0]-\mathrm{L}_0(Y|X)\Big)\Big] \\
&\qquad = \mathbb{E}\Big[p(X)(1-p)\Big(\mathbb{E}[Y|X,D=0]-\mathrm{L}_0(Y|X)\Big)\Big|D=0\Big] \\
&\qquad = \mathbb{E}\Big[p(X)Y\Big|D=0\Big](1-p) - \mathbb{E}\Big[\mathrm{L}(D|X)\mathrm{L}_0(Y|X)\Big)\Big|D=0\Big](1-p) \\
&\qquad = \mathbb{E}\Big[p(X)Y\Big|D=0\Big](1-p) - \mathbb{E}\Big[\mathrm{L}(D|X)Y\Big|D=0\Big](1-p) = 0
\end{aligned}$$

where the first equality holds by condition (i), the second equality uses the law of iterated expectations (and that $\mathbb{E}[D|X] = p(X)$), the third equality uses one of the rules we discussed in class for switching from an unconditional expectation to an expectation conditional on $D = 0$, the fourth equality splits the expectation into two parts and applies the law of iterated expectations on the first term, the fifth equality holds by the property of linear projections that we derived in class in Equation (3) in the course notes, and the last equality holds by condition (i). This shows that the second term is equal to 0 under condition (i).

That the weights are non-negative under condition (i) follows essentially immediately. The denominator is positive since it involves the expectation of a quadratic term. Under condition (i), the numerator must be positive because $\mathrm{L}(D|X) = p(X) \leq 1$, which implies that the numerator is non-negative.