STA640 Homework 3

Fan Bu

10/12/2020

Problem 1

Load the data first (I also checked that there is no missing values).

```r
debit = read.dta("debitcard199598.dta")
```

Since we have `spending1995`, we can use it as the "lagged outcome" to test for unconfoundedness. Fit a linear model with all the main effects, the treatment variable (`debit_card1998`) and interactions between treatment and all other covariates:

$$Y_{i,\text{lag}} \sim \beta X_{i,\text{no lag}} + \delta Z_i + \gamma X_{i,\text{no lag}} * Z_i,$$

where $X_{i,\text{no lag}}$ consists of all the covariates except `spending1995`. We want to test if $\delta = 0$, which implies

$$\mathbb{E}(Y_{i,\text{lag}}(Z_i = 1) - Y_{i,\text{lag}}(Z_i = 0) \mid X_{i,\text{no lag}} = x) = 0$$

for all configurations x of covariates.

```r
debit_lag = debit %>% select(-spending1998)
lag.ml = lm(spending1995 ~ . + debit_card1998 * (.), data=debit_lag)
summary(lag.ml)$coefficients['debit_card19981 (Yes)',]
```

```
##     Estimate   Std. Error      t value     Pr(>|t|)
## -868.5014493 1178.0534380   -0.7372343    0.4613176
```

Given the summary output, we shouldn't reject the null that $\delta = 0$, which means we can accept the unconfoundedness assumption.

Problem 2

Now check for overlap and balance.

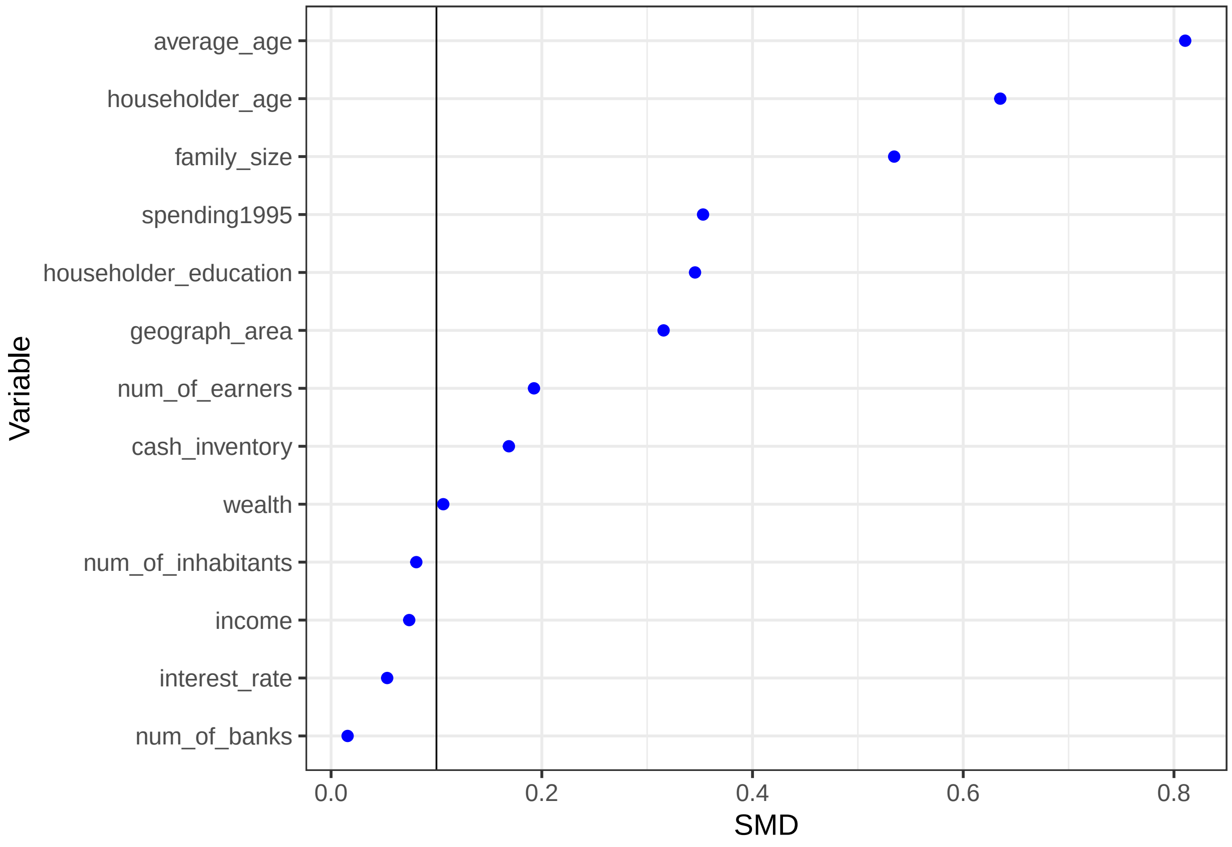
First check for balance by calculating the standard mean difference (i.e., the 2nd version of ASD with heurisic threshold at 0.1) for each covariate, which yields the Love plot below.

We can see that there is severe imbalance on `average_age`, `householder_age` and `family_size`, among quite a few unbalanced covariates overall.

```r
data_lag <- svydesign(ids = ~ 1, data = debit_lag, weights = ~ 1)

vars = names(debit_lag)[2:14]

## Construct a table that checks balance
tabOrig <- svyCreateTableOne(vars = vars, strata = 'debit_card1998',
                             data = data_lag, test = FALSE)
## Show table with SMD
## (no output due to space limit)
#print(tabOrig, smd = TRUE)
```

```r
SMD = data.frame(SMD = as.numeric(ExtractSmd(tabOrig)),
                 Variable = vars)
SMD = SMD %>% arrange(desc(SMD))
```

```r
## Order variable names by magnitude of SMD
varNames <- as.character(SMD$Variable)[order(SMD$SMD)]
## Order factor levels in the same order
SMD$Variable <- factor(SMD$Variable, levels = varNames)

ggplot(data = SMD,
       mapping = aes(x = Variable, y = SMD)) +
    #geom_line() +
    geom_hline(yintercept = 0.1, color = "black", size = 0.3) +
    geom_point(color='blue') +
    coord_flip() +
    theme_bw() + theme(legend.key = element_blank())
```

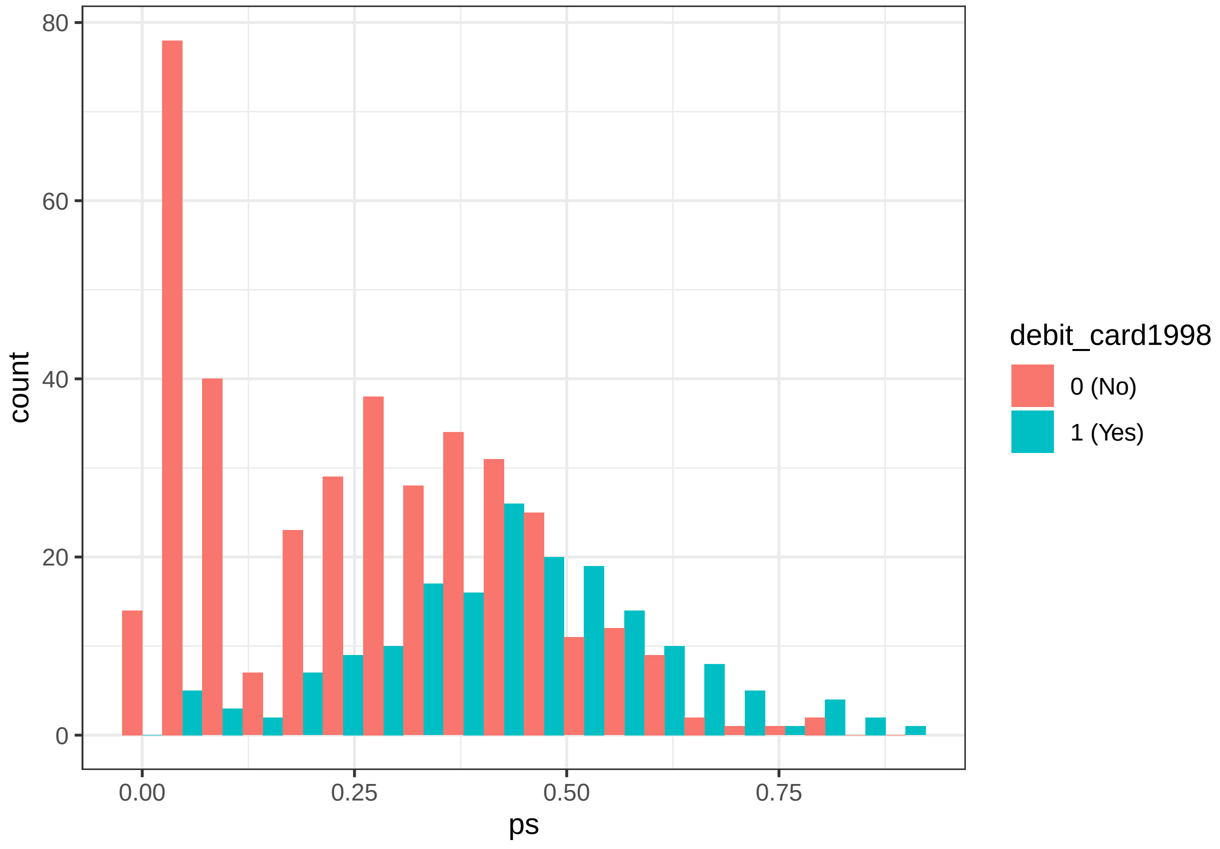


And then estimate the propensity score and plot its histograms for two groups.

From the histogram below, we can see that **there is some overlap between treated and control**, but there are some units with very low propensity scores (indicating not-great balance, which consolidates what we've found through SMD calculation).

```r
# use the data that doesn't contain outcome
# this PS model is chosen by trying out main effects and interactions
# until reasonably good balance is achieved
ps.model = glm(debit_card1998 ~ cash_inventory + interest_rate +
                 wealth + income + spending1995 + num_of_banks +
                 num_of_inhabitants + family_size + num_of_earners +
                 average_age + geograph_area + householder_age +
                 householder_education +
                 householder_age * householder_education,
                data=debit_lag, family=binomial('logit'))
PS = ps.model$fitted.values
```

```
debit_lag$ps = PS
ggplot(data=debit_lag, aes(x=ps,fill=debit_card1998)) +
  geom_histogram(bins = 20, position='dodge') +
  theme_bw()
```



Problem 3

Estimate the ATT.

(a) Outcome regression

Fit a linear model with main effects of all the covariates, the treatment, and all possible interaction terms between treatment and covariates. Then the estimator for ATT would be

$$\hat{\tau} = \frac{1}{N_1}\sum_{i=1}^{N} \mathbf{1}(Z_i = 1)(Y_i - \hat{m}_0(X_i)).$$

I'll use bootstrap to obtain the standard error of this estimate.

```
get_or_estimate <- function(dat){
  or = lm(spending1998 ~ . + debit_card1998 * (.), data=dat)

  dat_trt = dat %>% filter(debit_card1998 == "1 (Yes)")
  spending_trt = dat_trt$spending1998

  #dat_trt = dat_trt %>% dplyr::select(-spending1998)
  dat_trt = dat_trt[,-1]
  dat_trt$debit_card1998 = "0 (No)"

  spending_trt0 = predict(or, dat_trt)

  mean(spending_trt - spending_trt0)
}

get_or_sd <- function(dat, B=2000, seed=42){
  set.seed(seed)
  N = nrow(dat)
  estimates = sapply(1:B, function(i) {
    # bootstrap
    d.samp = dat[sample(1:N, replace = T),]

    # re-adjust the factors
    d.samp[] <- lapply(d.samp, function(x) if(is.factor(x)) factor(x) else x)
    d.samp[] <- lapply(d.samp, function(x) if(is.factor(x) & length(levels(x))<2) as.numeric(x) else x)

    get_or_estimate(d.samp)
  })

  sd(estimates)
}
```

```
Est = get_or_estimate(debit)
SD = get_or_sd(debit)
cat('Estimate:', Est,'\nStandard error:', SD,'\n')
```

```
## Estimate: 189.9409
## Standard error: 69.53933
```

(b) Weighting estimator

Here I'm going to be lazy and use the `PSweight` package. Since the estimand is ATT, the weights on treated units and control units should be

$$\begin{aligned} w_1(x_i) &\propto 1, \\ w_0(x_i) &\propto \frac{\hat{e}(x_i)}{1-\hat{e}(x_i)}. \end{aligned}$$

Here $\hat{e}(x_i)$ will be estimated using the PS model that I have chosen earlier on. The standard error is estimated via bootstrap (with 1000 replicates).

```
ps.formula = debit_card1998 ~ cash_inventory + interest_rate +
                 wealth + income + spending1995 + num_of_banks +
                 num_of_inhabitants + family_size + num_of_earners +
                 average_age + geograph_area + householder_age +
                 householder_education +
                 householder_age * householder_education
PS.fit = PSweight(ps.formula = ps.formula,
                  zname = "debit_card1998",
                  yname = "spending1998",
                  weight = "treated",
                  data = debit,
                  bootstrap = TRUE,
                  R = 1000)
```

Then check out the estimate, standard error, as well as 95% confidence interval.

```
summary(PS.fit)
```

```
## Original group value:  0 (No), 1 (Yes)
## Treatment group value:  1 (Yes)
##
## contrast
##            0 (No) 1 (Yes)
## Contrast 1     -1       1
##
##            Estimate Std.Error Lower.CL Upper.CL p.value
## Contrast 1 189.2617  98.29678 6.925294 391.1966   0.054
```

(c) Double robust estimator

In the case of estimating ATT, a "doubly robust estimator" can be thought of as augmenting the estimator in **(b)** with outcome regression.

Here I'll use the augmented estimator in Mao, H., Li, L., Greene, T. (2019):

$$\hat{\tau}_{\text{aug}} = \frac{\sum_{i=1}^{n} \hat{h}(x_i)(\hat{m}_1(x_i) - \hat{m}_0(x_i))}{\sum_{i=1}^{n} \hat{h}(x_i)} + \frac{\sum_{i=1}^{n} \hat{w}_1(x_i) Z_i (Y_i - \hat{m}_1(x_i))}{\sum_{i=1}^{n} \hat{w}_1(x_i) Z_i} - \frac{\sum_{i=1}^{n} \hat{w}_0(x_i)(1 - Z_i)(Y_i - \hat{m}_0(x_i))}{\sum_{i=1}^{n} \hat{w}_0(x_i)(1 - Z_i)},$$

where $\hat{h}(x_i) = \hat{e}(x_i)$ is the (estimated) tilting function for the target population corresponding to ATT.

The outcome regression model and PS model are the same as those in **(a)** and **(b)**. The standard error is estimated via bootstrap.

```
get_DR_est <- function(dat){
  N = nrow(dat)
  # PS model
  ps.model = glm(debit_card1998 ~ cash_inventory + interest_rate +
                 wealth + income + spending1995 + num_of_banks +
                 num_of_inhabitants + family_size + num_of_earners +
                 average_age + geograph_area + householder_age +
                 householder_education +
                 householder_age * householder_education,
               data=debit_lag, family=binomial('logit'))
  PS = ps.model$fitted.values
  trt_weight = case_when(dat$debit_card1998 == "0 (No)" ~ PS/(1-PS),
                         dat$debit_card1998 == "1 (Yes)" ~ 1)
  # outcome model
  or.model = lm(spending1998 ~ . + debit_card1998 * (.), data=dat)
  POs = matrix(nrow=N, ncol=2)
  new = dat
  new$debit_card1998 = "0 (No)"
  POs[,1] = predict(or.model, new)
  new$debit_card1998 = "1 (Yes)"
  POs[,2] = predict(or.model, new)
  # calculate tau.hat
  control = which(dat$debit_card1998 == "0 (No)")
  treated = which(dat$debit_card1998 == "1 (Yes)")
  tau = sum(PS/sum(PS) * (POs[,2] - POs[,1])) +
    sum(trt_weight[treated]/sum(trt_weight[treated]) *
    (dat$spending1998[treated] - POs[treated,2])) -
    sum(trt_weight[control]/sum(trt_weight[control]) *
    (dat$spending1998[control] - POs[control,1]))
  tau
}

get_DR_sd <- function(dat, B=2000, seed=42){
  set.seed(seed)
  N = nrow(dat)
  estimates = sapply(1:B, function(i) {
    # bootstrap
    d.samp = dat[sample(1:N, replace = T),]

    # re-adjust the factors
    d.samp[] <- lapply(d.samp, function(x) if(is.factor(x)) factor(x) else x)
    d.samp[] <- lapply(d.samp, function(x) if(is.factor(x) & length(levels(x))<2) as.numeric(x) else x)

    get_DR_est(d.samp)
  })

  sd(estimates)
}
```

```
Est = get_DR_est(debit)
SD = get_DR_sd(debit)
cat('Estimate:', Est,'\nStandard error:', SD,'\n')
```

```
## Estimate: 213.5282
## Standard error: 96.37981
```

(d) Bayesian outcome regression

Fit a Bayesian linear model to regress outcomes on covariates in the treated group and control group respectively. Here I'll use the `rstanarm` package to do Bayesian linear regression; as the default setting of this package, a flat prior ($Unif(0,1)$) on R^2 is adopted for the linear model.

```
library(rstan)
library(rstanarm)
```

Split the dataset into the treated arm and control arm, and then fit a Bayesian LM model on each subset. The fitted models can then be used to impute $Y_i(0)$ as well as sample $Y_i(1)$ for each treated unit; the posterior parameter draws can also be directly used to compute estimates for the ATT estimand τ (more details are stated later on).

```
Yes = debit %>% filter(debit_card1998 == "1 (Yes)") %>%
  select(-debit_card1998)

No = debit %>% filter(debit_card1998 == "0 (No)") %>%
  select(-debit_card1998)
```

```r
yes.fit = stan_lm(spending1998 ~ ., data=Yes, prior = R2(0.6), seed=42)
Y1 = Yes$spending1998
Y1_fit = posterior_predict(yes.fit)
sigma1 = as.matrix(yes.fit)[,'sigma']
```

```r
no.fit = stan_lm(spending1998 ~ ., data=No, prior = R2(0.6), seed=42)
Y0 = No$spending1998
#Y0_fit = posterior_predict(no.fit)
Y0_predict = posterior_predict(no.fit, newdata = Yes)
sigma0 = as.matrix(no.fit)[,'sigma']
```

(i) Only draw $Y_i(0)$ from posterior for each treated unit i

At the same time, also vary ρ from 0 to 1 (by an interval of 0.1). It seems that the result isn't very sensitive to the choice of ρ.

```r
# for treated units, draw Y(0)'s and get the mean
get_Y0 <- function(rho=0.5){
  nsamp = length(sigma0)
  N1 = nrow(Yes)
  #samps = numeric(n)
  mu = Y0_predict + rho * sigma0/sigma1 * (Y1 - Y1_fit)
  sigma = sigma0 * sqrt(1-rho^2)
  Y0_means = sapply(1:nsamp, function(i){
    mean(rnorm(N1, mu[i,], sigma[i]))
  })
  Y0_means
}

# method (i)
for(rho in seq(0.0,1.0,by=0.1)){
  Y0_means = get_Y0(rho)
ATT1 = mean(Y1) - Y0_means
cat('For rho =', round(rho,1), 'Estimate:', mean(ATT1),
    '\n\t95% CI: [', paste0(round(quantile(ATT1, c(.025,.975)),4),
                           collapse = ", "), ']\n')
}
```

```
## For rho = 0 Estimate: 201.173
##  95% CI: [ 64.5923, 341.2284 ]
## For rho = 0.1 Estimate: 202.1186
##  95% CI: [ 64.7927, 338.408 ]
## For rho = 0.2 Estimate: 199.134
##  95% CI: [ 60.6377, 335.4593 ]
## For rho = 0.3 Estimate: 200.9591
##  95% CI: [ 64.6828, 344.1249 ]
## For rho = 0.4 Estimate: 200.6888
##  95% CI: [ 58.2918, 344.4529 ]
## For rho = 0.5 Estimate: 199.9349
##  95% CI: [ 56.2411, 343.7977 ]
## For rho = 0.6 Estimate: 201.33
##  95% CI: [ 56.3711, 345.7991 ]
## For rho = 0.7 Estimate: 200.4102
##  95% CI: [ 51.248, 349.372 ]
## For rho = 0.8 Estimate: 200.74
##  95% CI: [ 46.8721, 355.6618 ]
## For rho = 0.9 Estimate: 200.3408
##  95% CI: [ 43.0779, 357.1242 ]
## For rho = 1 Estimate: 200.537
##  95% CI: [ 36.8883, 363.5437 ]
```

(ii) Draw both $Y_i(0)$ and $Y_i(1)$ from posterior for each treated unit i

Same as before, vary the value of ρ from 0 to 1. Still, the result isn't much affected by the choice of ρ.

```r
# for treated units, draw Y(1) as well
Y1_means = apply(Y1_fit, 1, mean)

# method (ii)
for(rho in seq(0.0,1.0,by=0.1)){
  Y0_means = get_Y0(rho)
  ATT2 = Y1_means - Y0_means
  cat('For rho =', round(rho,1), 'Estimate:', mean(ATT2),
      '\n\t95% CI: [', paste0(round(quantile(ATT2, c(.025,.975)),4), collapse = ", "), ']\n')
}
```

```
## For rho = 0 Estimate: 200.4433
##  95% CI: [ 1.9285, 395.2798 ]
## For rho = 0.1 Estimate: 199.9323
##  95% CI: [ -5.6572, 408.692 ]
## For rho = 0.2 Estimate: 201.2748
##  95% CI: [ -13.9315, 410.0267 ]
## For rho = 0.3 Estimate: 200.7998
##  95% CI: [ -20.8685, 417.0382 ]
## For rho = 0.4 Estimate: 201.019
##  95% CI: [ -27.2832, 431.5182 ]
## For rho = 0.5 Estimate: 199.6514
##  95% CI: [ -39.6358, 442.5205 ]
## For rho = 0.6 Estimate: 199.7784
##  95% CI: [ -45.4995, 448.7475 ]
## For rho = 0.7 Estimate: 199.7528
##  95% CI: [ -62.1656, 461.0076 ]
## For rho = 0.8 Estimate: 200.071
##  95% CI: [ -60.3603, 461.9183 ]
## For rho = 0.9 Estimate: 199.9216
##  95% CI: [ -71.7685, 473.3208 ]
## For rho = 1 Estimate: 199.793
##  95% CI: [ -75.9485, 481.8093 ]
```

(iii) Express τ in terms of model parameters θ directly

Given that

$$\begin{pmatrix} Y_i(1) \\ Y_i(0) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \beta_1^T X_i \\ \beta_0^T X_i \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_0 \\ \rho\sigma_1\sigma_0 & \sigma_0^2 \end{pmatrix} \right),$$

we have $\delta_i = Y_i(1) - Y_0(1)$ follows

$$\delta_i \sim \mathcal{N}(\beta_1^T X_i - \beta_0^T X_i, \sigma_1^2 + \sigma_0^2 - 2\rho\sigma_0\sigma_1).$$

Then our estimand is

$$\tau = \frac{1}{N} \sum_{i:Z_i=1} \delta_i.$$

Now draw posterior predictive samples of τ by drawing samples of the parameters from their posterior. Again, vary ρ from 0 to 1; the point estimate isn't affected by ρ but the standard error is (not considerably though).

```
get_delta_mean <- function(rho=0.5){
  nsamp = length(sigma0)
  N1 = nrow(Yes)
  Diffs = Y1_fit - Y0_predict
  diff_sigma = sqrt(sigma0^2 + sigma1^2 - 2*rho*sigma1*sigma0)
  delta_means = sapply(1:nsamp, function(i){
    mean(rnorm(N1, Diffs[i,], diff_sigma[i]))
  })
  delta_means
}

# method (iii)
for(rho in seq(0.0,1.0,by=0.1)){
  delta_means = get_delta_mean(rho)
  cat('For rho =', round(rho,1), 'Estimate:', mean(delta_means),
      '\n\t95% CI: [', paste0(round(quantile(delta_means,
                                              c(.025,.975)),4),
                              collapse = ", "), ']\n')
}
```

```
## For rho = 0 Estimate: 199.4343
##  95% CI: [ -23.6728, 420.5165 ]
## For rho = 0.1 Estimate: 201.7145
##  95% CI: [ -13.0975, 417.9326 ]
## For rho = 0.2 Estimate: 201.1598
##  95% CI: [ -16.4127, 416.2707 ]
## For rho = 0.3 Estimate: 200.4828
##  95% CI: [ -7.3964, 410.0428 ]
## For rho = 0.4 Estimate: 200.6482
##  95% CI: [ -6.1399, 404.2038 ]
## For rho = 0.5 Estimate: 200.0062
##  95% CI: [ -6.691, 400.7662 ]
## For rho = 0.6 Estimate: 200.2996
##  95% CI: [ 6.558, 398.7409 ]
## For rho = 0.7 Estimate: 199.4454
##  95% CI: [ 2.0733, 391.1688 ]
## For rho = 0.8 Estimate: 200.6103
##  95% CI: [ 12.2607, 389.5276 ]
## For rho = 0.9 Estimate: 200.958
##  95% CI: [ 16.2798, 387.0633 ]
## For rho = 1 Estimate: 200.4059
##  95% CI: [ 16.2483, 380.9661 ]
```

Comparing the 3 methods

(i) is estimating the **sample** ATT, while **(ii)** and **(iii)** are closer to estimating the **population** ATT. However here we are doing these through a hybrid approach by conditioning on the sample distribution of covariates X instead of modeling the distribution of X within the treated group.

Also, it is notable that the latter 2 methods yield larger variances (wider credible intervals), since in **(i)** uncertainty only comes from imputing $Y_i(0)$'s, but in **(ii)** and **(iii)** uncertainty comes from "imputing" both potential outcomes.