Assignment4

Kaicheng Luo

2019/10/22

```
# A DIY function of Matching Estimator
# Some preparations: Distance Measure
Edistance <- function(X, Y)
{
  if (length(X) != length(Y)){print("Error: Length Not Matched")}
  else{return(sqrt(sum((X-Y)^2)))}
}
Mdistance <- function(X, Y, cov)
{
  if (length(X) != length(Y)){print("Error: Length Not Matched")}
  else{return(t(X - Y) %*% solve(cov) %*% (X-Y))}
}
kNN <- function(x0, x, y, numberOfMatch = 1, dis = "Euclidean", COV = 0)
{
  # x0 shall be a vector and x shall be a matrix, y shall be the corresponding vector of responding x
  # This function returns a numeric value if x0 is a number, and a vector if x0 is a vector
  if (dis == "Euclidean")
  {
    rankDis = c()
    for (i in 1:nrow(x))
    {
      rankDis = c(rankDis, Edistance(x0, x[i,]))
    }
    rankDis = rank(rankDis)
    Y_hat = mean(y[which(rankDis<=numberOfMatch)])
    X_hat = x[which(rankDis<=numberOfMatch),]
    return(list("Y" = Y_hat, "X" = X_hat))
  }
  else if (dis == "M")
  {
    if (COV == 0){print("Error: covariance matrix not provided.")}
    else{
      rankDis = c()
      for (i in 1:nrow(x))
      {
        rankDis = c(rankDis, Mdistance(x0, x[i,], COV))
      }
      rankDis = rank(rankDis)
      Y_hat = mean(y[which(rankDis<=numberOfMatch)])
      X_hat = x[which(rankDis<=numberOfMatch),]
      return(list("Y" = Y_hat, "X" = X_hat))
    }
  }
}
MyMatching <- function(y, tr, x, numberOfMatch = 1, dis = "Euclidean", COV = 0)
{
  # First deal with the treatment group
```

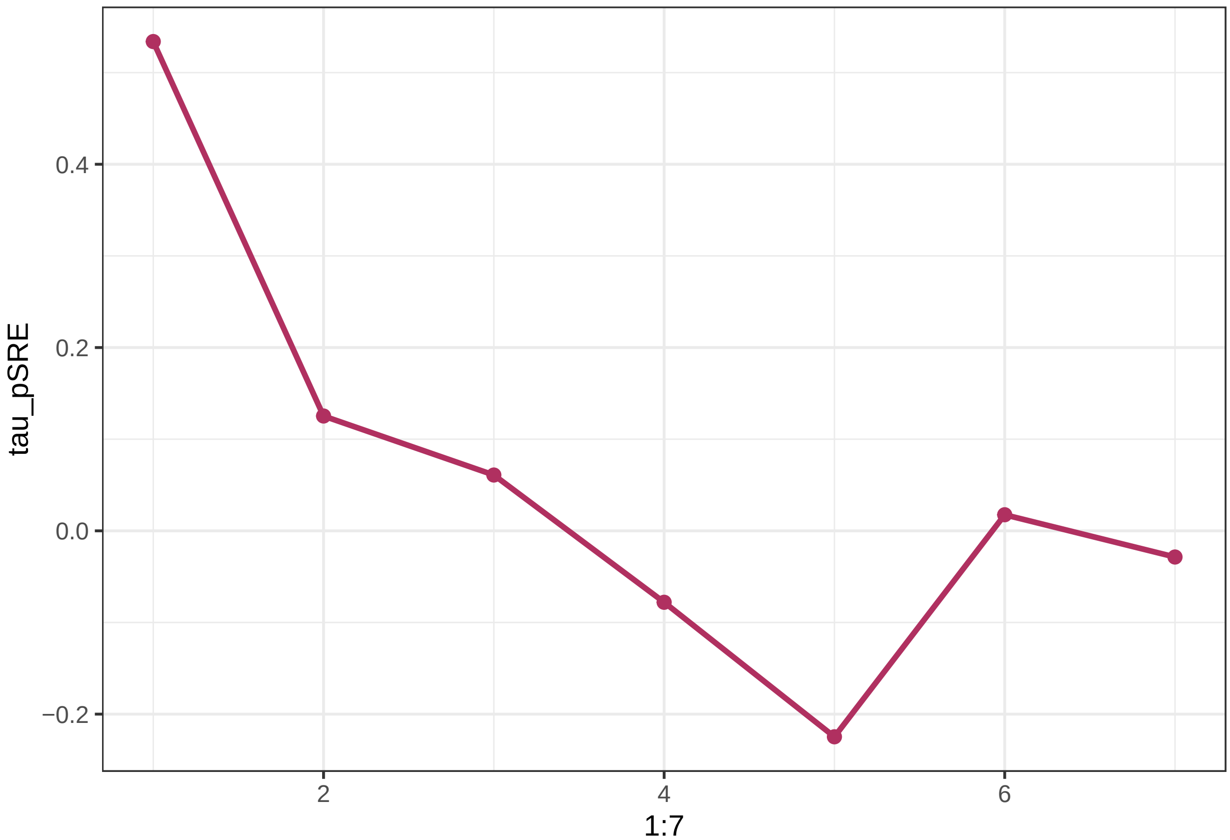
```
  Y1_treat = y[tr == 1]
  Y0_treat = c()
  for (i in 1:length(y[tr == 1]))
  {
    Y0_treat = c(Y0_treat, kNN(x[i,], x[tr == 0,], y[tr == 0], numberOfMatch, dis, COV)$Y)
  }
  # Then deal with the control group
  Y0_control = y[tr == 0]
  Y1_control = c()
  for (i in 1:length(y[tr == 0]))
  {
    Y1_control = c(Y1_control, kNN(x[i,], x[tr == 1,], y[tr == 1], numberOfMatch, dis, COV)$Y)
  }
  # Calculate the test statistics (matching estimator and bias-corrected matching estimator)
  tau_m = mean(c(Y1_treat, Y1_control) - c(Y0_treat, Y0_control))
    # Fit two models for Y0 and Y1, respectively
  model0 <- lm(Y1_treat~x[tr == 1,])$coef
  model1 <- lm(Y0_control~x[tr == 0,])$coef
  bias = 0
  for (i in 1:length(y[tr == 1]))
  {
    predx = c(1, x[i,]) %*% model0
    prednn = cbind(1, kNN(x[i,], x[tr == 0,], y[tr == 0], numberOfMatch, dis)$X) %*% model0
    bias = bias + mean(predx - prednn)
  }
  for (i in 1:length(y[tr == 0]))
  {
    predx = c(1, x[i,]) %*% model1
    prednn = cbind(1, kNN(x[i,], x[tr == 1,], y[tr == 1], numberOfMatch, dis)$X) %*% model1
    bias = bias + mean(prednn - predx)
  }
  bias = bias / length(y)
  tau_mbc = tau_m - bias
  return(list("tau_m" = tau_m, "tau_mbc" = tau_mbc))
}
```

```
data("nhanes_bmi")
z = nhanes_bmi$School_meal
y = nhanes_bmi$BMI
x = as.matrix(nhanes_bmi[, -c(1, 2)])
pscore = glm(z ~ x, family = binomial)$fitted.values

# Propensity Score Estimator
stat_SRE <- function(stratum, treatment, y){
  # Assume in our case that the stratum in arranged and indexed.
  # If not, then re-code it to an index.
  number = length(unique(stratum))
  tau = 0
  wil = 0
  r = 0
  # Calculate the three statistics as defined
  for (i in 1:number){
    tempy = y[stratum == i]
```

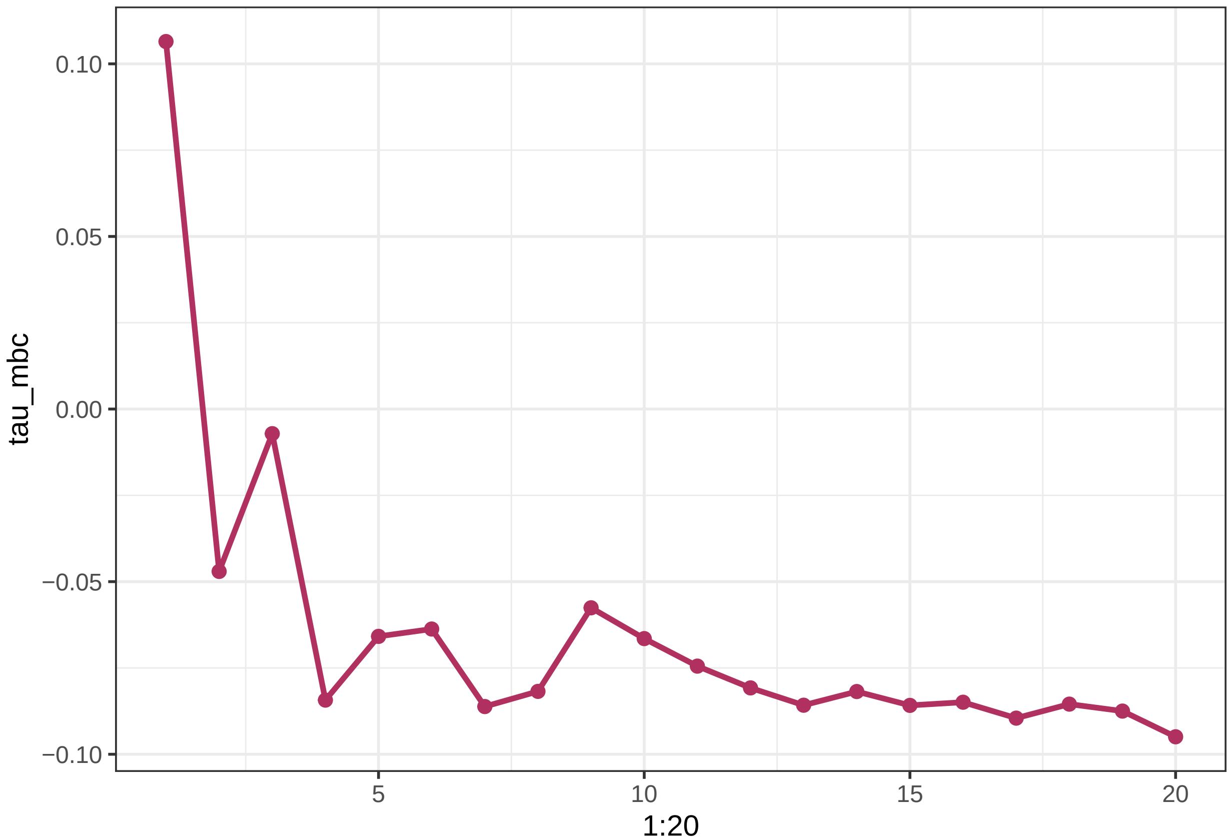
```r
    tempt = treatment[stratum == i]
    n = length(tempy)
    pi = n/length(y)
    tau = tau + pi*(mean(tempy[tempt == 1] - mean(tempy[tempt == 0])))
    wil = wil + wilcox.test(tempy[tempt == 1], tempy[tempt == 0])$statistic / (n+1)
    tempy = tempy - mean(tempy)
  }
  y <- rank(y)
  for (i in 1:length(y)){
    if (treatment[i] == 1){
      r = r + y[i]
    }
  }
  return(c(taus = tau, wilcoxon = wil, alignedRank = r))
}
# Here we obtain the obs. values
tau_pSRE = c()
for (i in 1:7){
  stratum = floor(pscore*i) + 1
  obsValue <- stat_SRE(stratum, z, y)
  tau_pSRE = c(tau_pSRE, obsValue[1])
}
```

```
## Warning in wilcox.test.default(tempy[tempt == 1], tempy[tempt == 0]):
## cannot compute exact p-value with ties
```

```r
ggplot()+
  theme_bw() +
  geom_point(aes(x = 1:7, y = tau_pSRE), color = 'maroon', size = 2) +
  geom_line(aes(x = 1:7, y = tau_pSRE), color = 'maroon', size = 1)
```

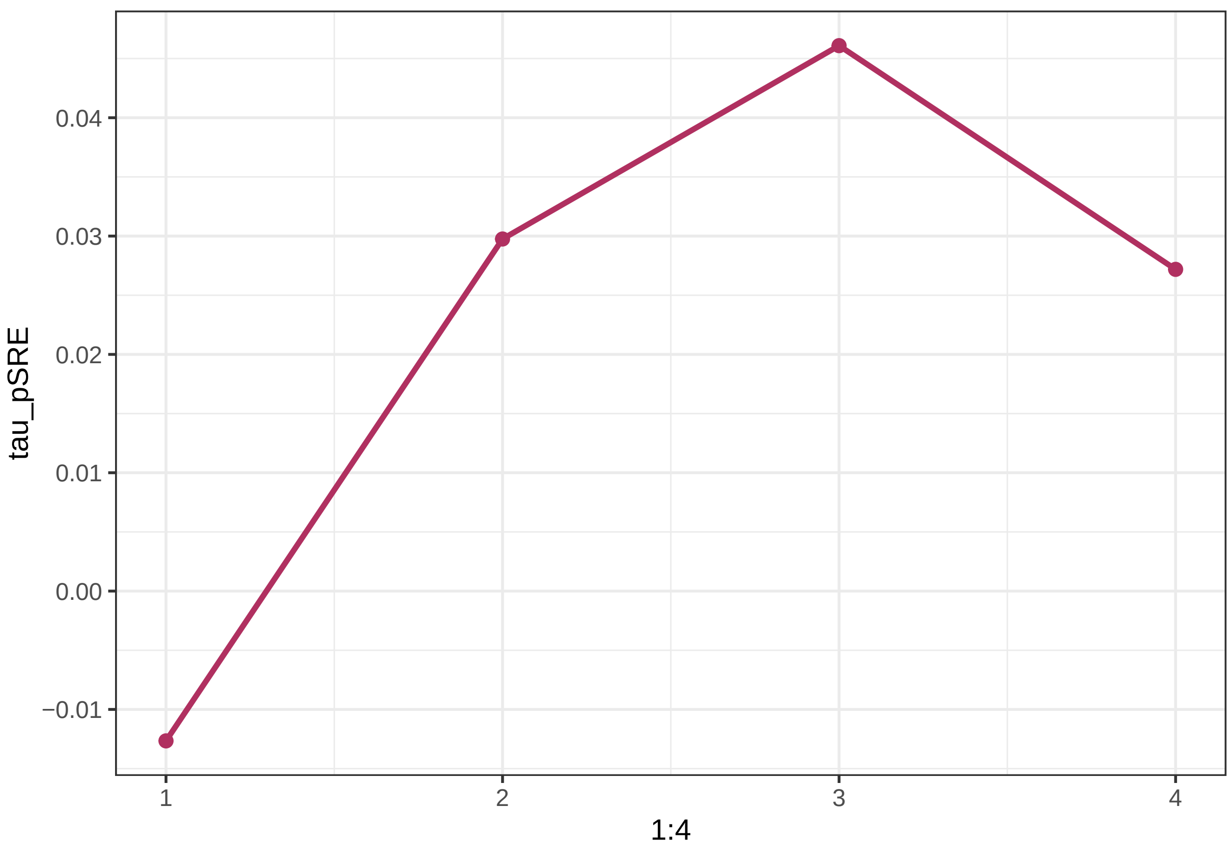



```r
# Matching Estimator (Bias-Corrected)
tau_mbc = c()
se_tau = c()
for (i in 1:20)
{
  model = Match(y, z, x, estimand = "ATE", M = i, BiasAdjust = T)
  tau_mbc = c(tau_mbc, model$est)
  se_tau = c(se_tau, model$se)
}
ggplot()+
  theme_bw() +
  geom_point(aes(x = 1:20, y = tau_mbc), color = 'maroon', size = 2) +
  geom_line(aes(x = 1:20, y = tau_mbc), color = 'maroon', size = 1)
```



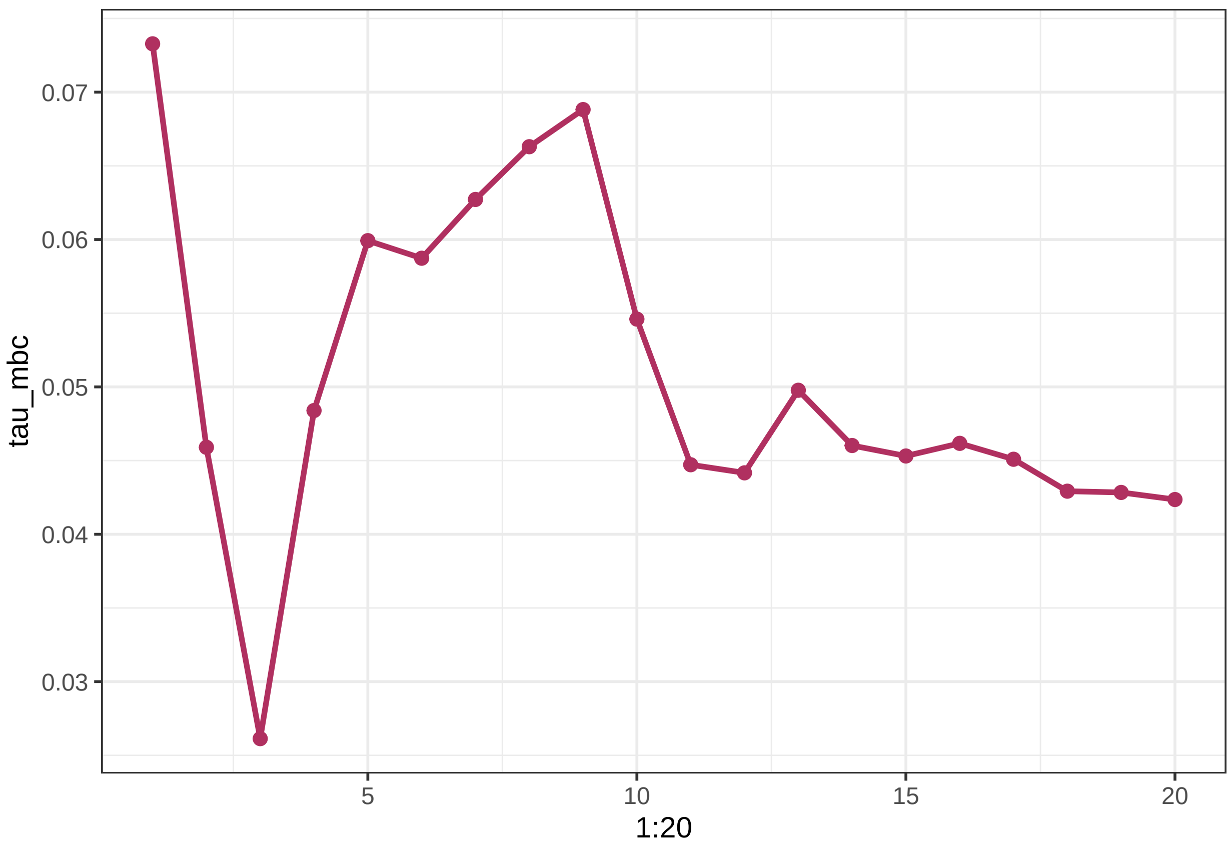

```
karolinska = read.table("karolinska.txt", header = TRUE)

z = karolinska$HighVolDiagHosp
y = 1 - (karolinska$YearsSurvivingAfterDiagnosis == 1)
x = as.matrix(karolinska[, c(3, 4, 5)])

pscore = glm(z ~ x, family = binomial)$fitted.values

# Propensity Score Estimator
stat_SRE <- function(stratum, treatment, y){
  # Assume in our case that the stratum in arranged and indexed.
  # If not, then re-code it to an index.
  number = length(unique(stratum))
  tau = 0
  wil = 0
  r = 0
  # Calculate the three statistics as defined
  for (i in 1:number){
    tempy = y[stratum == i]
    tempt = treatment[stratum == i]
    n = length(tempy)
    pi = n/length(y)
    tau = tau + pi*(mean(tempy[tempt == 1] - mean(tempy[tempt == 0])))
    tempy = tempy - mean(tempy)
  }
  y <- rank(y)
  for (i in 1:length(y)){
```

```r
    if (treatment[i] == 1){
      r = r + y[i]
    }
  }
  return(c(taus = tau, alignedRank = r))
}
# Here we obtain the obs. values
tau_pSRE = c()
for (i in 1:4){
  stratum = floor(pscore*i) + 1
  obsValue <- stat_SRE(stratum, z, y)
  tau_pSRE = c(tau_pSRE, obsValue[1])
}
ggplot()+
  theme_bw() +
  geom_point(aes(x = 1:4, y = tau_pSRE), color = 'maroon', size = 2) +
  geom_line(aes(x = 1:4, y = tau_pSRE), color = 'maroon', size = 1)
```




```r
# Matching Estimator (Bias-Corrected)
tau_mbc = c()
se_tau = c()
for (i in 1:20)
{
  model = Match(y, z, x, estimand = "ATE", M = i, BiasAdjust = T)
  tau_mbc = c(tau_mbc, model$est)
  se_tau = c(se_tau, model$se)
```

```r
}
ggplot()+
  theme_bw() +
  geom_point(aes(x = 1:20, y = tau_mbc), color = 'maroon', size = 2) +
  geom_line(aes(x = 1:20, y = tau_mbc), color = 'maroon', size = 1)
```




```r
library("car")
```

```
## Warning: package 'car' was built under R version 3.5.2
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```r
library("Matching")

## experimental data
lalonde = read.csv("cps1re74.csv", sep = " ")

y = lalonde$re78
z = lalonde$treat
x = as.matrix(lalonde[, c("age", "educ", "black",
                          "hispan", "married", "nodegree",
                          "re74", "re75")])

## analysis the randomized experiment via
# Lin's Estimator
xc = scale(x)
linols = lm(y ~ z + xc + z*xc)
tau_Lin = linols$coefficients[2]
round(summary(linols)$coef[2, ], 4)
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
## -3689.8516  2457.3996    -1.5015     0.1332
```

```r
sqrt(hccm(linols)[2, 2])
```

```
## [1] 3428.439
```

```r
# Regression Imputation Shall return the same results as Lin
model1 = lm(y[z == 1]~x[z == 1,])
model0 = lm(y[z == 0]~x[z == 0,])
tau_reg = mean(cbind(1, x) %*% model1$coefficients - cbind(1, x) %*% model0$coefficients)
tau_Lin - tau_reg
```

```
##            z
## 3.187779e-09
```

```r
# Propensity score stratification
pscore = glm(z ~ x, family = binomial)$fitted.values
tau_pSRE = c()
for (i in 2:5){
  stratum = floor(pscore*i) + 1
  obsValue <- stat_SRE(stratum, z, y)
  tau_pSRE = c(tau_pSRE, obsValue[1])
}
tau_pSRE
```

```
##      taus      taus      taus      taus
## -8506.495 -7804.194 -7706.861 -7625.539
```

```r
# IPW estimator
# With no truncation
truncpscore = c(0,1)
```

```r
pscore = glm(z ~ x, family = binomial)$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_1 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_1 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# With certain truncation
truncpscore = c(0.1,0.9)
pscore = glm(z ~ x, family = binomial)$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_2 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_2 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# Doubly Robust Estimator
outcome1 = glm(y ~ x, weights = z, family = gaussian)$fitted.values
outcome0 = glm(y ~ x, weights = (1 - z), family = gaussian)$fitted.values
res1 = y - outcome1
res0 = y - outcome0
tau_dr = mean(outcome1 - outcome0) + mean(z*res1/pscore - (1 - z)*res0/(1 - pscore))

print(paste("Lin's estimator is ", tau_Lin, sep = ""))
```

```
## [1] "Lin's estimator is -3689.8515880728"
```

```r
print(paste("Regression Imputation gives us an estimate same as Lin's ", tau_reg, sep = ""))
```

```
## [1] "Regression Imputation gives us an estimate same as Lin's -3689.85158807599"
```

```r
print(paste("pscore stratification's estimator is ", tau_pSRE, sep = ""))
```

```
## [1] "pscore stratification's estimator is -8506.49536105039"
## [2] "pscore stratification's estimator is -7804.1941784074"
## [3] "pscore stratification's estimator is -7706.8611458836"
## [4] "pscore stratification's estimator is -7625.53852433987"
```

```r
print(paste("IPW estimator (hovic-thompson) without truncation is ",ipw_1, " and the hajek estimator is
```

```
## [1] "IPW estimator (hovic-thompson) without truncation is -10452.7456142986 and the hajek estimator i
```

```r
print(paste("IPW estimator (hovic-thompson) with truncation is ",ipw_2, " and the hajek estimator is ",i
```

```
## [1] "IPW estimator (hovic-thompson) with truncation is -15955.5388374637 and the hajek estimator is -
```

```r
print(paste("The doubly robust estimator is ", tau_dr))
```

```
## [1] "The doubly robust estimator is  -3688.65001081252"
```

```r
# Propensity score stratification
pscore = glm(z ~ x, family = binomial(link = "probit"))$fitted.values
tau_pSRE = c()
for (i in 2:5){
  stratum = floor(pscore*i) + 1
  obsValue <- stat_SRE(stratum, z, y)
  tau_pSRE = c(tau_pSRE, obsValue[1])
}

# IPW estimator
# With no truncation
truncpscore = c(0,1)
pscore = glm(z ~ x, family = binomial(link = "probit"))$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_1 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_1 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# With certain truncation
truncpscore = c(0.1,0.9)
pscore = glm(z ~ x, family = binomial(link = "probit"))$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_2 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_2 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# Doubly Robust Estimator
outcome1 = glm(y ~ x, weights = z, family = gaussian)$fitted.values
outcome0 = glm(y ~ x, weights = (1 - z), family = gaussian)$fitted.values
res1 = y - outcome1
res0 = y - outcome0
tau_dr = mean(outcome1 - outcome0) + mean(z*res1/pscore - (1 - z)*res0/(1 - pscore))

print("If we fit e(x) by probit model,")
```

```
## [1] "If we fit e(x) by probit model,"
```

```r
print(paste("Lin's estimator is ", tau_Lin, sep = ""))
```

```
## [1] "Lin's estimator is -3689.8515880728"
```

```r
print(paste("Regression Imputation gives us an estimate same as Lin's ", tau_reg, sep = ""))
```

```
## [1] "Regression Imputation gives us an estimate same as Lin's -3689.85158807599"
```

```r
print(paste("pscore stratification's estimator is ", tau_pSRE, sep = ""))
```

```
## [1] "pscore stratification's estimator is -8506.49536105039"
## [2] "pscore stratification's estimator is -7826.07775869297"
## [3] "pscore stratification's estimator is -7199.12284967037"
## [4] "pscore stratification's estimator is -7185.93488163915"
```

```r
print(paste("IPW estimator (hovic-thompson) without truncation is ",ipw_1, " and the hajek estimator is
```

```
## [1] "IPW estimator (hovic-thompson) without truncation is -8738.13952518101 and the hajek estimator i
```

```r
print(paste("IPW estimator (hovic-thompson) with truncation is ",ipw_2, " and the hajek estimator is ",i
```

```
## [1] "IPW estimator (hovic-thompson) with truncation is -15953.1987455341 and the hajek estimator is -
```

```r
print(paste("The doubly robust estimator is ", tau_dr))
```

```
## [1] "The doubly robust estimator is  -3686.27091355767"
```

```r
print("The doubly robust estimator is the one closest to the golden rule estimation.")
```

```
## [1] "The doubly robust estimator is the one closest to the golden rule estimation."
```

```r
library(MatchIt)
```

```
##
## Attaching package: 'MatchIt'

## The following object is masked _by_ '.GlobalEnv':
##
##     lalonde
```

```r
data <- read.csv("FDA-Carpenter.csv")
# We perform similar data treatment
# The democrat is the senate is the treatment indicator, acttime is Y, and others are covariate.

## rescaling
data$hospdisc <- data$hospdisc/100000
data$natreg <- data$natreg/100
data$stafcder <- data$stafcder/100
data$prevgenx <- data$prevgenx/100
data$hhosleng <- data$hhosleng/10
data$condavg3 <- data$condavg3/10
data$orderent <- data$orderent/10
data$vandavg3 <- data$vandavg3/10
data$wpnoavg3 <- data$wpnoavg3/100

z <- data$demsnmaj
y <- data$acttime
x <- as.matrix(data %>%
  dplyr::select(-demsnmaj, -acttime))
## analysis the randomized experiment via
# Lin's Estimator
xc = scale(x)
linols = lm(y ~ z + xc + z*xc)
tau_Lin = linols$coefficients[2]
round(summary(linols)$coef[2, ], 4)
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
##   -13.8625     4.4678    -3.1027     0.0021
```

```
sqrt(hccm(linols)[2, 2])
```

```
## [1] 4.389857
```

```
# Regression Imputation Shall return the same results as Lin
model1 = lm(y[z == 1]~x[z == 1,])
model0 = lm(y[z == 0]~x[z == 0,])
tau_reg = mean(cbind(1, x) %*% model1$coefficients - cbind(1, x) %*% model0$coefficients)

# Propensity score stratification
pscore = glm(z ~ x, family = binomial)$fitted.values
tau_pSRE = c()
for (i in 2:5){
  stratum = floor(pscore*i) + 1
  obsValue <- stat_SRE(stratum, z, y)
  tau_pSRE = c(tau_pSRE, obsValue[1])
}
tau_pSRE
```

```
##      taus      taus      taus      taus
## -11.78213 -13.49131 -14.69429 -16.93254
```

```
# IPW estimator
# With no truncation
truncpscore = c(0,1)
pscore = glm(z ~ x, family = binomial)$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_1 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_1 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# With certain truncation
truncpscore = c(0.1,0.9)
pscore = glm(z ~ x, family = binomial)$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_2 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_2 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# Doubly Robust Estimator
outcome1 = glm(y ~ x, weights = z, family = gaussian)$fitted.values
outcome0 = glm(y ~ x, weights = (1 - z), family = gaussian)$fitted.values
res1 = y - outcome1
res0 = y - outcome0
tau_dr = mean(outcome1 - outcome0) + mean(z*res1/pscore - (1 - z)*res0/(1 - pscore))

print(paste("Lin's estimator is ", tau_Lin, sep = ""))
```

```
## [1] "Lin's estimator is -13.8624685450355"
```

```r
print(paste("Regression Imputation gives us an estimate same as Lin's ", tau_reg, sep = ""))
```

```
## [1] "Regression Imputation gives us an estimate same as Lin's -13.8624685450355"
```

```r
print(paste("pscore stratification's estimator is ", tau_pSRE, sep = ""))
```

```
## [1] "pscore stratification's estimator is -11.7821330501416"
## [2] "pscore stratification's estimator is -13.4913054500575"
## [3] "pscore stratification's estimator is -14.6942924219253"
## [4] "pscore stratification's estimator is -16.9325445893505"
```

```r
print(paste("IPW estimator (hovic-thompson) without truncation is ",ipw_1, " and the hajek estimator is
```

```
## [1] "IPW estimator (hovic-thompson) without truncation is -14.6313842730615 and the hajek estimator i
```

```r
print(paste("IPW estimator (hovic-thompson) with truncation is ",ipw_2, " and the hajek estimator is ",i
```

```
## [1] "IPW estimator (hovic-thompson) with truncation is -14.6646978648039 and the hajek estimator is -
```

```r
print(paste("The doubly robust estimator is ", tau_dr))
```

```
## [1] "The doubly robust estimator is  -12.6943681855002"
```

```r
data <- read.csv("Visibility-Koch.csv")
data <- subset(data, subset=c(repman==1 & voter==1))
data <- na.omit(data)

y = data$prcanid
z = 1 - data$rvisman
x = data %>%
  dplyr::select("repcan1", "goppty", "rideo", "rproj", "repft", "aware") %>%
  as.matrix

## analysis the randomized experiment via
# Lin's Estimator
xc = scale(x)
linols = lm(y ~ z + xc + z*xc)
tau_Lin = linols$coefficients[2]
round(summary(linols)$coef[2, ], 4)
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
##    -0.0251     0.0602    -0.4168     0.6769
```

```r
sqrt(hccm(linols)[2, 2])
```

```
## [1] 0.05782918
```

```r
# Regression Imputation Shall return the same results as Lin
model1 = lm(y[z == 1]~x[z == 1,])
model0 = lm(y[z == 0]~x[z == 0,])
tau_reg = mean(cbind(1, x) %*% model1$coefficients - cbind(1, x) %*% model0$coefficients)

# Propensity score stratification
pscore = glm(z ~ x, family = binomial)$fitted.values
tau_pSRE = c()
for (i in 2:5){
  stratum = floor(pscore*i) + 1
  obsValue <- stat_SRE(stratum, z, y)
  tau_pSRE = c(tau_pSRE, obsValue[1])
}
tau_pSRE
```

```
##         taus         taus         taus         taus
##  0.028850929  0.007541221          NaN -0.006391974
```

```r
# IPW estimator
# With no truncation
truncpscore = c(0,1)
pscore = glm(z ~ x, family = binomial)$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_1 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_1 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# With certain truncation
truncpscore = c(0.1,0.9)
pscore = glm(z ~ x, family = binomial)$fitted.values
pscore = pmax(truncpscore[1], pmin(truncpscore[2], pscore))
ipw_2 = mean(z*y/pscore - (1 - z)*y/(1 - pscore))
ipw_hajek_2 = mean(z*y/pscore)/mean(z/pscore) - mean((1 - z)*y/(1 - pscore))/mean((1 - z)/(1 - pscore))

# Doubly Robust Estimator
outcome1 = glm(y ~ x, weights = z, family = gaussian)$fitted.values
outcome0 = glm(y ~ x, weights = (1 - z), family = gaussian)$fitted.values
res1 = y - outcome1
res0 = y - outcome0
tau_dr = mean(outcome1 - outcome0) + mean(z*res1/pscore - (1 - z)*res0/(1 - pscore))

print(paste("Lin's estimator is ", tau_Lin, sep = ""))
```

```
## [1] "Lin's estimator is -0.0250891102390644"
```

```r
print(paste("Regression Imputation gives us an estimate same as Lin's ", tau_reg, sep = ""))
```

```
## [1] "Regression Imputation gives us an estimate same as Lin's -0.025089110239063"
```

```r
# Note that there could be Na values as stratums of pscores might not always include enough control and
print(paste("pscore stratification's estimator is ", tau_pSRE, sep = ""))
```

```
## [1] "pscore stratification's estimator is 0.028850928914995"
## [2] "pscore stratification's estimator is 0.00754122081333689"
## [3] "pscore stratification's estimator is NaN"
## [4] "pscore stratification's estimator is -0.00639197365510275"
```

```
print(paste("IPW estimator (hovic-thompson) without truncation is ",ipw_1, " and the hajek estimator is
```

```
## [1] "IPW estimator (hovic-thompson) without truncation is -0.142357535224816 and the hajek estimator
```

```
print(paste("IPW estimator (hovic-thompson) with truncation is ",ipw_2, " and the hajek estimator is ",
```

```
## [1] "IPW estimator (hovic-thompson) with truncation is -0.178563422656618 and the hajek estimator is
```

```
print(paste("The doubly robust estimator is ", tau_dr))
```

```
## [1] "The doubly robust estimator is  -0.0394785911682918"
```

```
ATT.est = function(z, y, x, out.family = gaussian, Utruncpscore = 1)
{
  ## sample size
  nn  = length(z)
  nn1 = sum(z)

  ## fitted propensity score
  pscore   = glm(z ~ x, family = binomial)$fitted.values
  pscore   = pmin(Utruncpscore, pscore)
  odds.pscore = pscore/(1 - pscore)

  ## fitted potential outcomes
  outcome0 = glm(y ~ x, weights = (1 - z),
                 family = out.family)$fitted.values

  ## regression imputation estimator
  ace.reg0 = lm(y ~ z + x)$coef[2]
  ace.reg  = mean(y[z==1]) - mean(outcome0[z==1])
  ## propensity score weighting estimator
  ace.ipw0 = mean(y[z==1]) -
                mean(odds.pscore*(1 - z)*y)*nn/nn1
  ace.ipw  = mean(y[z==1]) -
                mean(odds.pscore*(1 - z)*y)/mean(odds.pscore*(1 - z))
  ## doubly robust estimator
  res0     = y - outcome0
  ace.dr   = ace.reg - mean(odds.pscore*(1 - z)*res0)*nn/nn1

  return(c(ace.reg0, ace.reg, ace.ipw0, ace.ipw, ace.dr))
}


ObsCausal.ATT = function(z, y, x, n.boot = 10^2,
                         out.family = gaussian, Utruncpscore = 1)
{
```

```r
  point.est  = ATT.est(z, y, x, out.family, Utruncpscore)

  ## nonparametric bootstrap
  n.sample   = length(z)
  x          = as.matrix(x)
  boot.est   = replicate(n.boot,
                         {id.boot = sample(1:n.sample, n.sample, replace = TRUE)
                          ATT.est(z[id.boot], y[id.boot], x[id.boot, ],
                                  out.family, Utruncpscore)})

  boot.se    = apply(boot.est, 1, sd)

  res        = rbind(point.est, boot.se)
  rownames(res) = c("est", "se")
  colnames(res) = c("reg0", "reg", "HT", "Hajek", "DR")

  return(res)
}



n.sim   = 2000
ATT0    = rep(0, n.sim)
ATT     = matrix(0, 5, n.sim)
SEboot  = matrix(0, 5, n.sim)
n       = 200

## Baseline: simulation with correct models
## nonparallel y1 and y0
for(r in 1:n.sim)
{
  x       = matrix(rnorm(n*2), n, 2)
  x1      = cbind(1, x)
  beta.z  = c(0, 1, 1)
  pscore  = 1/(1 + exp(- as.vector(x1%*%beta.z)))
  z       = rbinom(n, 1, pscore)
  beta.y1 = c(1, 2, 1)
  beta.y0 = c(1, 1, 1)
  y1      = rnorm(n, x1%*%beta.y1)
  y0      = rnorm(n, x1%*%beta.y0)
  y       = z*y1 + (1 - z)*y0
  ATT0[r] = mean(y1[z==1]) - mean(y0[z==1])

  causaleffect = ObsCausal.ATT(z, y, x)
  ATT[, r]     = causaleffect[1, ]
  SEboot[, r]  = causaleffect[2, ]
}

apply(ATT, 1, mean) - mean(ATT0)
```

```
## [1] -0.359320852  0.004192477 -0.011764708  0.038584431  0.001900091
```

```r
apply(ATT, 1, sd)
```

```
## [1] 0.1843905 0.2101216 0.7792105 0.3939605 0.2576940
```

```r
apply(SEboot, 1, mean)
```

```
## [1] 0.1860617 0.2098427 0.5523045 0.3167784 0.2492950
```

```r
## Case 1:
## Wrong Model for y1 and y0 but correct model for propensity score
for(r in 1:n.sim)
{
  x       = matrix(rnorm(n*2), n, 2)
  x1      = cbind(1, x)
  beta.z  = c(0, 1, 1)
  pscore  = 1/(1 + exp(- as.vector(x1%*%beta.z)))
  z       = rbinom(n, 1, pscore)
  beta.y1 = c(1, 2, 1)
  beta.y0 = c(1, 1, 1)
  realbeta.y1 = c(2, 3, 4)
  realbeta.y0 = c(100, 101, 102)
  y1      = rnorm(n, x1%*%realbeta.y1)
  y0      = rnorm(n, x1%*%realbeta.y0)
  y       = z*y1 + (1 - z)*y0
  ATT0[r] = mean(y1[z==1]) - mean(y0[z==1])

  causaleffect = ObsCausal.ATT(z, y, x)
  ATT[, r]     = causaleffect[1, ]
  SEboot[, r]  = causaleffect[2, ]
}

apply(ATT, 1, mean) - mean(ATT0)
```

```
## [1] 71.3687951608  0.0003030757  0.1103855378  4.1287009031 -0.0008410053
```

```r
apply(ATT, 1, sd)
```

```
## [1] 12.00293 12.18383 66.10132 32.79879 12.18419
```

```r
apply(SEboot, 1, mean)
```

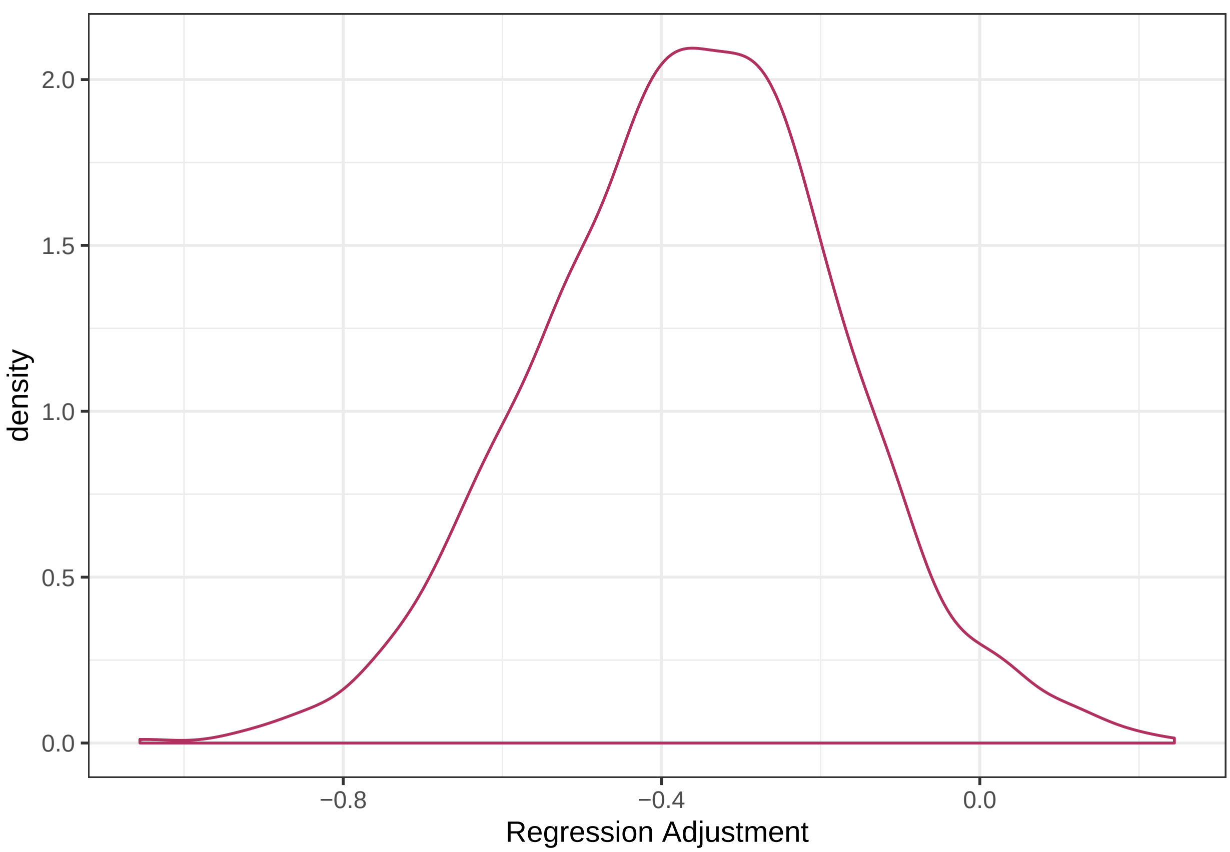
```
## [1] 11.93985 11.82235 48.30555 23.22371 11.82418
```

```r
## Case 2:
## Correct Model for y1 and y0 but wrong model for propensity score
## Note that we're generating a uniform distribution of pscore. // z = rbinom(n, 1, pscore)
for(r in 1:n.sim)
{
  x       = matrix(rnorm(n*2), n, 2)
```

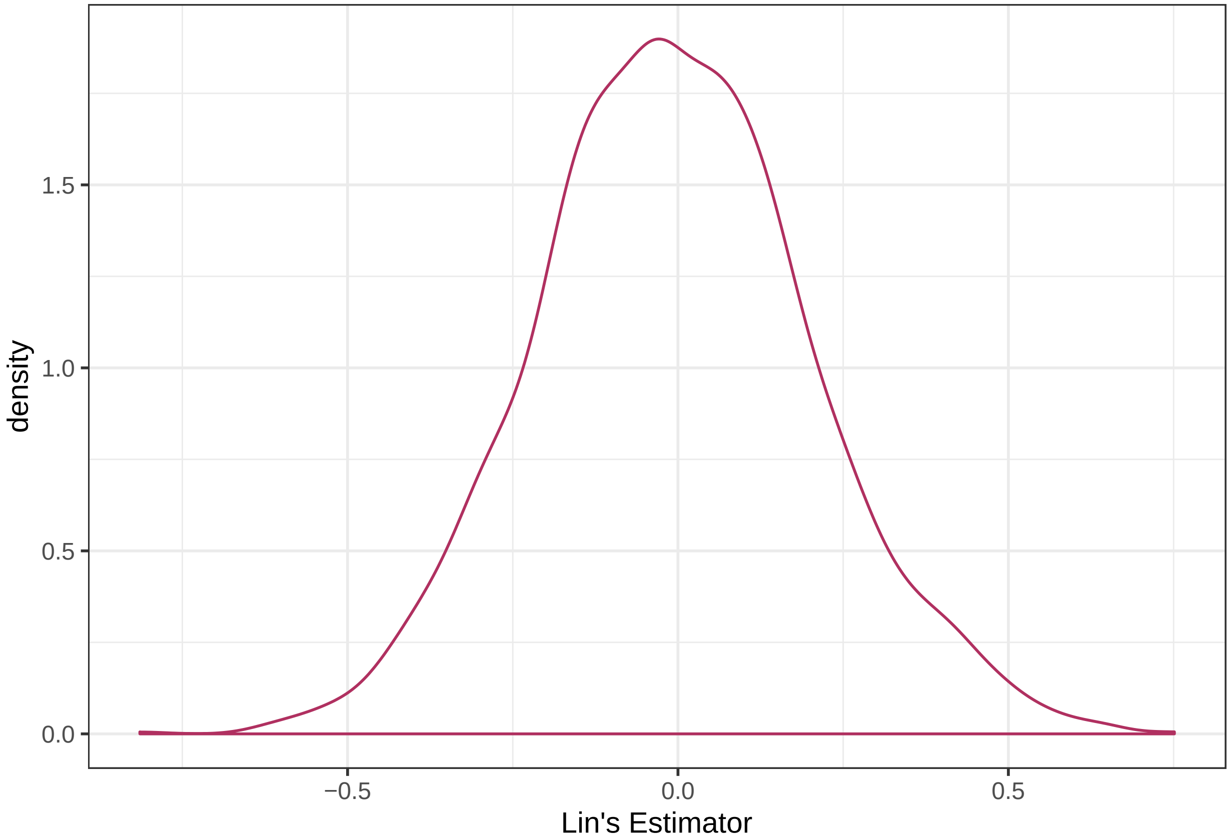
```r
  x1       = cbind(1, x)
  beta.z   = c(0, 1, 1)
  pscore   = 1/(1 + exp(- as.vector(x1%*%beta.z)))
  realpscore = 1/(1 + exp(-as.vector(x1 %*% c(0,0,0))))
  z        = rbinom(n, 1, pscore)
  beta.y1 = c(1, 2, 1)
  beta.y0 = c(1, 1, 1)
  y1       = rnorm(n, x1%*%beta.y1)
  y0       = rnorm(n, x1%*%beta.y0)
  y        = z*y1 + (1 - z)*y0
  ATT0[r] = mean(y1[z==1]) - mean(y0[z==1])

  causaleffect = ObsCausal.ATT(z, y, x)
  ATT[, r]      = causaleffect[1, ]
  SEboot[, r]   = causaleffect[2, ]
}

apply(ATT, 1, mean) - mean(ATT0)
```

```
## [1] -0.363686898 -0.006544096  0.002204782  0.037634792 -0.010464628
```

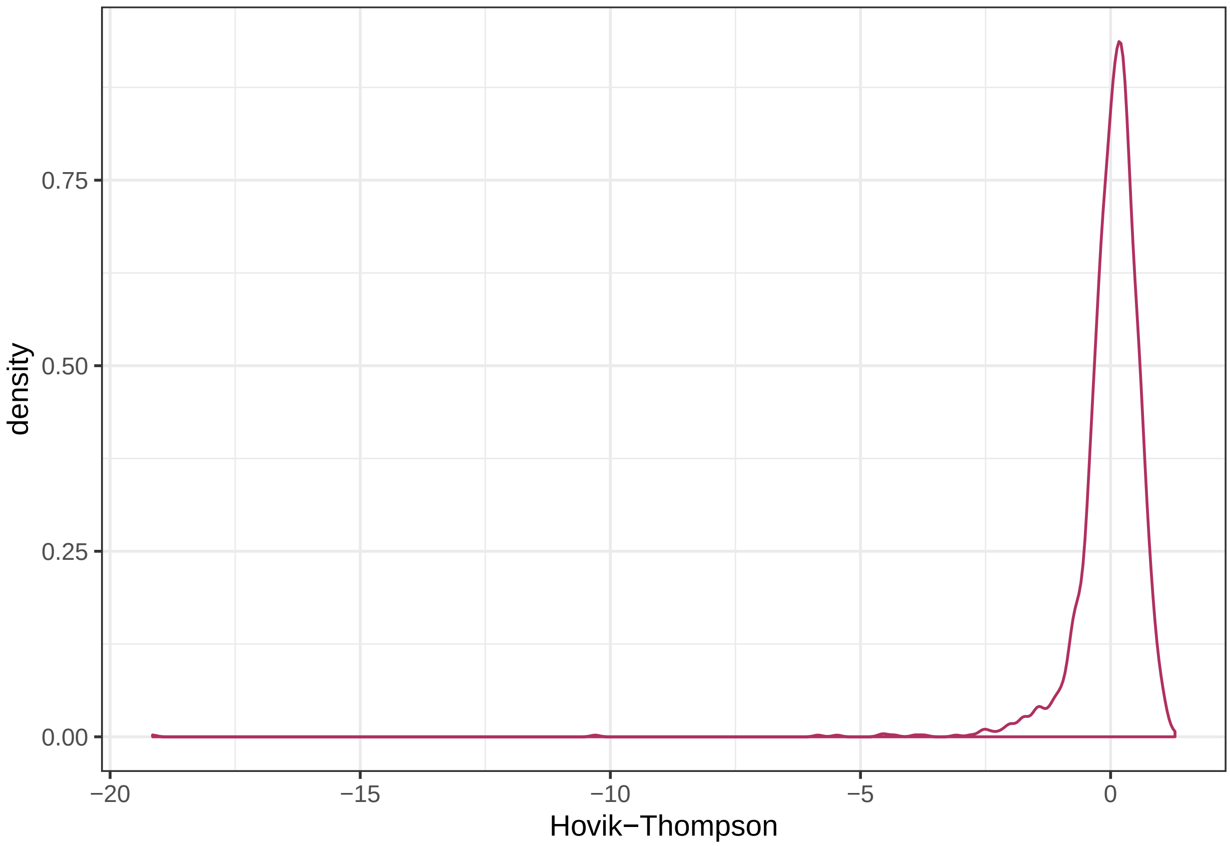
```r
ggplot() + theme_bw() +
  geom_density(aes(x = ATT[1,] - mean(ATT0)), color = "maroon") + labs(x = "Regression Adjustment")
```



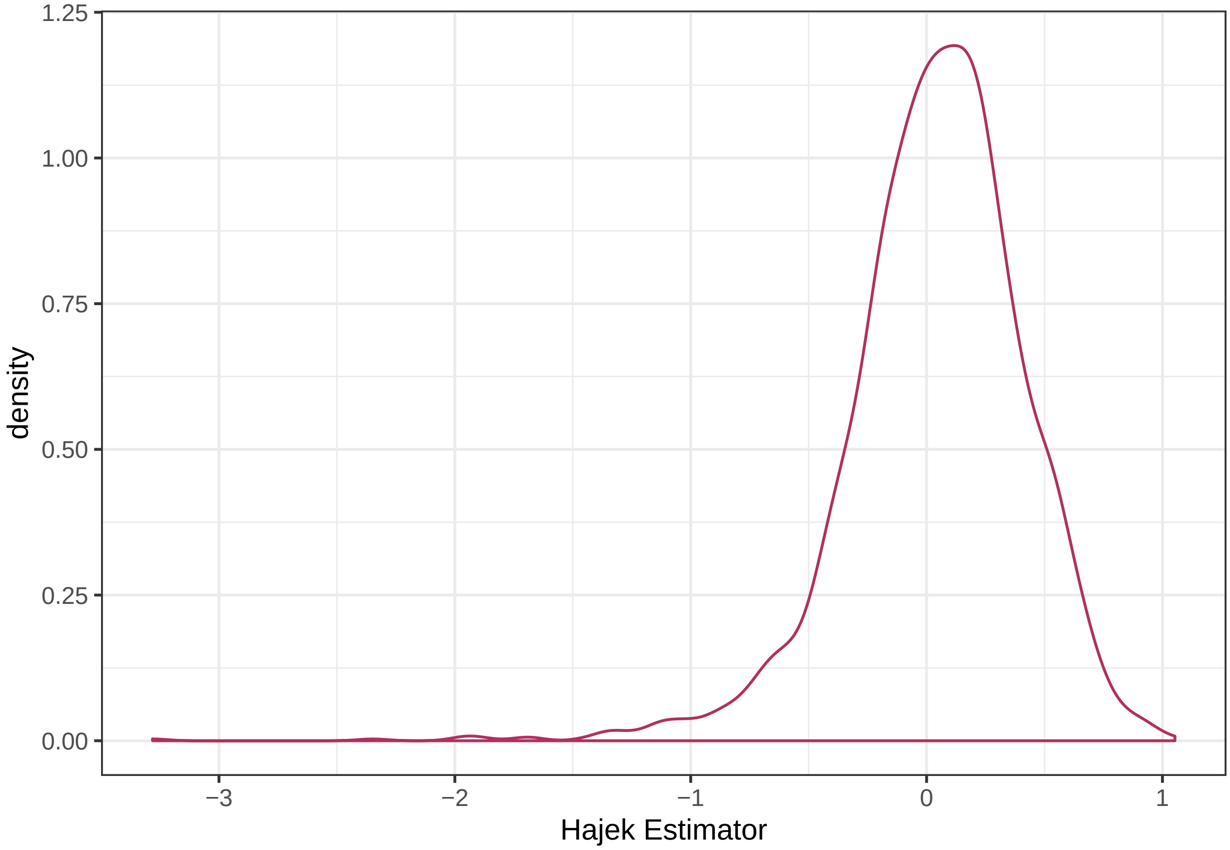
```r
ggplot() + theme_bw() +
  geom_density(aes(x = ATT[2,] - mean(ATT0)), color = "maroon") + labs(x = "Lin's Estimator")
```



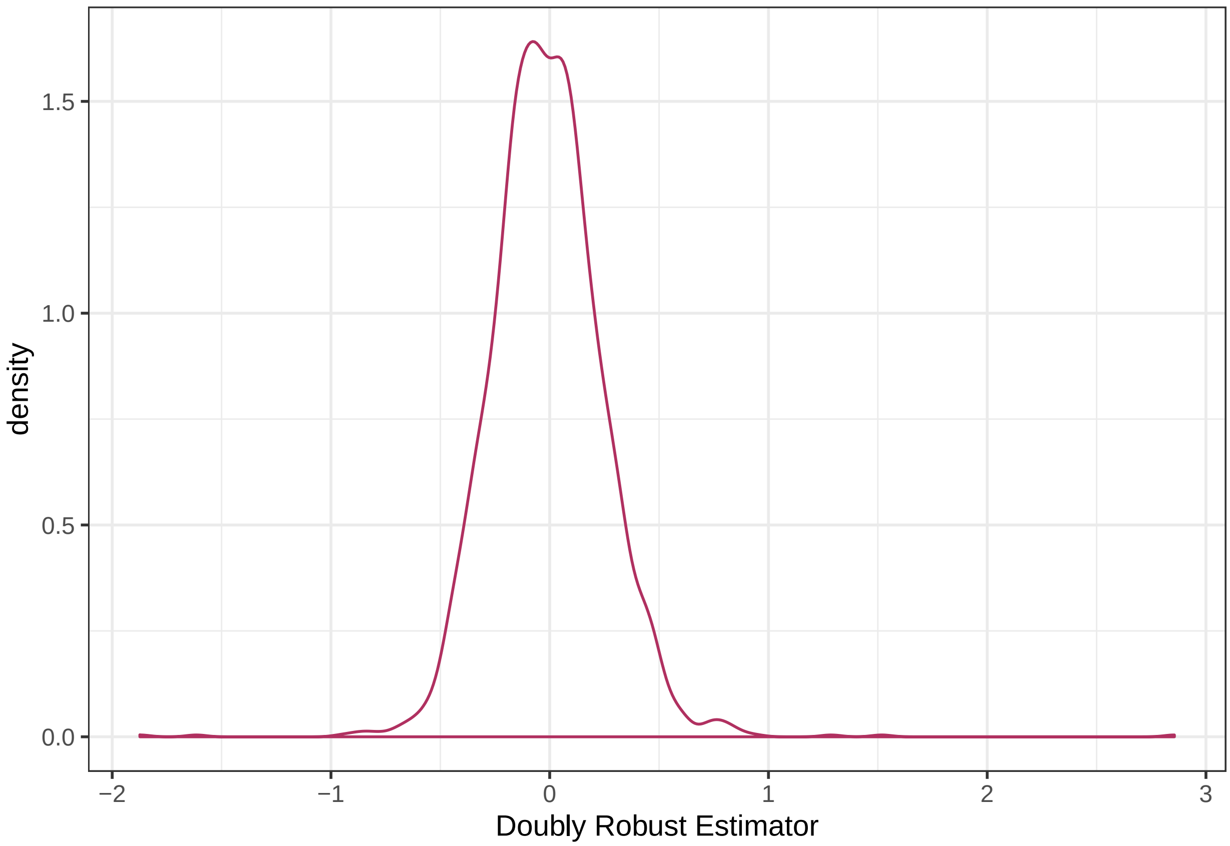

```r
ggplot() + theme_bw() +
  geom_density(aes(x = ATT[3,] - mean(ATT0)), color = "maroon") + labs(x = "Hovik-Thompson")
```




```
ggplot() + theme_bw() +
  geom_density(aes(x = ATT[4,] - mean(ATT0)), color = "maroon") + labs(x = "Hajek Estimator")
```




```
ggplot() + theme_bw() +
  geom_density(aes(x = ATT[5,] - mean(ATT0)), color = "maroon") + labs(x = "Doubly Robust Estimator")
```




```
apply(ATT, 1, sd)
```

```
## [1] 0.1895128 0.2094865 0.7741395 0.3772193 0.2608904
```

```
apply(SEboot, 1, mean)
```

```
## [1] 0.1847593 0.2089299 0.5328994 0.3129829 0.2457885
```

```
apply(ATT, 1, mean) - mean(ATT0) - 1.96*apply(ATT, 1, sd)
```

```
## [1] -0.7351320 -0.4171376 -1.5151086 -0.7017150 -0.5218098
```

```
## Case 3:
## Wrong model for both y0/ y1 and pscore
for(r in 1:n.sim)
{
  x        = matrix(rnorm(n*2), n, 2)
  x1       = cbind(1, x)
  beta.z   = c(0, 1, 1)
  pscore   = 1/(1 + exp(- as.vector(x1%*%beta.z)))
  realpscore = 1/(1 + exp(-as.vector(x1 %*% c(0,0,0))))
  z        = rbinom(n, 1, pscore)
  beta.y1 = c(1, 2, 1)
  beta.y0 = c(1, 1, 1)
```

```r
  realbeta.y1 = c(2, 3, 4)
  realbeta.y0 = c(100, 101, 102)
  y1      = rnorm(n, x1%*%realbeta.y1)
  y0      = rnorm(n, x1%*%realbeta.y0)
  y       = z*y1 + (1 - z)*y0
  ATT0[r] = mean(y1[z==1]) - mean(y0[z==1])

  causaleffect = ObsCausal.ATT(z, y, x)
  ATT[, r]     = causaleffect[1, ]
  SEboot[, r]  = causaleffect[2, ]
}

apply(ATT, 1, mean) - mean(ATT0)
```

```
## [1] 71.292947086 -0.004852229  3.120162985  5.626615117 -0.004806452
```

```r
apply(ATT, 1, sd)
```

```
## [1] 12.08423 12.13165 61.82708 30.81661 12.13796
```

```r
apply(SEboot, 1, mean)
```

```
## [1] 11.91909 11.84323 45.62230 22.54042 11.84496
```