A1+D2 algorithm, P3 Propensity Score Estimation

Depeng Kong & Zihan Chen

11/25/2020

1. Setup

```
library(Matching)
library(glmnet)
library(tidyverse)
library(ggplot2)
setwd("./")
```

2. Load Data

```
ldim <- read.csv("lowDim_dataset.csv")
hdim <- read.csv("highDim_dataset.csv")

# Low Dimention
ltr <- ldim$A
ly <- ldim$Y
lx <- ldim[,-c(1,2)]

# High Dimention
htr <- hdim$A
hy <- hdim$Y
hx <- hdim[, -c(1,2)]
```

3. Calculate Propensity Score with L2 Ridge regression

3.1 Low Dimention

```
set.seed(0)
glm1 <- cv.glmnet(as.matrix(lx), ltr, family = "binomial", alpha = 0)

glm1.fit <- predict(glm1$glmnet.fit,
                    s = glm1$lambda.min,
                    newx = as.matrix(lx),
                    type = "response")
```

3.2 High Dimention

```
set.seed(0)
glm2 <- cv.glmnet(as.matrix(hx), htr, family = "binomial", alpha = 0)

glm2.fit <- predict(glm2$glmnet.fit,
                    s = glm2$lambda.min,
                    newx = as.matrix(hx),
                    type = "response")
```

4. Distance calculated from Propensity Score Matching

4.1 Low Dimention

```
n1 <- length(glm1.fit)
dt1 <- matrix(nrow = n1, ncol = n1)
for (i in 1:n1){
  for (j in 1:n1){
    dt1[i,j] <- abs(glm1.fit[i] - glm1.fit[j])
  }
}

# dt1
# summary(dt1)
```

4.2 High Dimention

```
n2 <- length(glm2.fit)
dt2 <- matrix(nrow = n2, ncol = n2)
for (i in 1:n2){
  for (j in 1:n2){
    dt2[i,j] <- abs(glm2.fit[i] - glm2.fit[j])
  }
}
```

5. Propensity Score Marching

5.1 Matching function

```
cal_neighbour <- function(index,df,thresh,y,A){
  dt_vec <- df[index,]
  ind_vec <- which(dt_vec<threshold)
  ind_final=ind_vec[A[index]!=A[ind_vec]]

  if (length(ind_final)==0){
    return(NA)
```

```
  }
  else{
    return(list(mean(y[ind_final]),ind_final))
  }

}
```

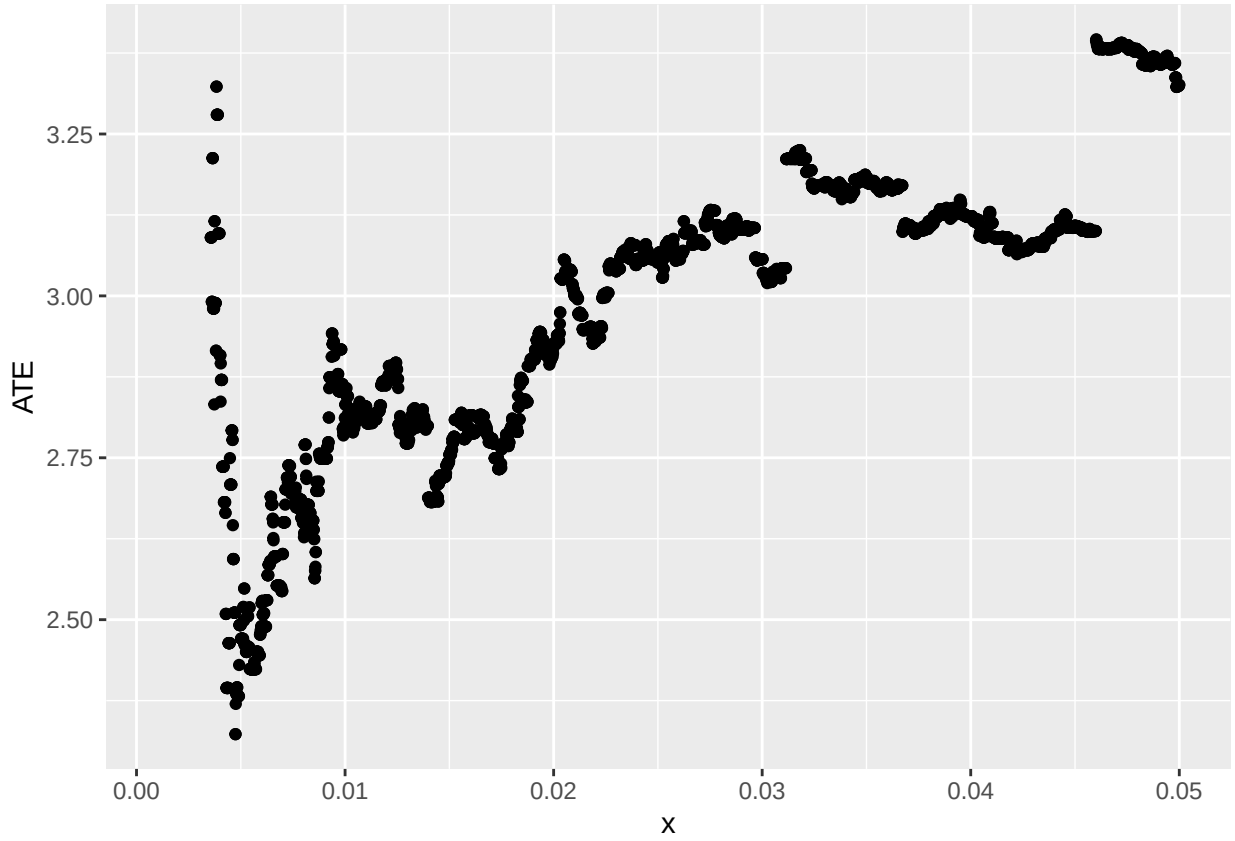
```
####Matching Main Low-Dim


a <- as.vector(dt1)

seq = 100:5000/100000

ATE_low <- vector("double")
pairs_low <- vector("double")
for (percentage in seq){

  threshold <- quantile(a,percentage)


  n1_vec <- 1:n1
  list_1 <- lapply(n1_vec,cal_neighbour,df=dt1,thresh = threshold,y=ly,A=ltr)
  mean_list_1 <- lapply(n1_vec,function(x) unlist(list_1[[x]][1]))
  mean_cal_1 <- unlist(mean_list_1)
  neighbour_list_1 <- lapply(n1_vec,function(x) unlist(list_1[[x]][2]))

  df_1 <- (data.frame(Y=ly,A=ltr)
           %>%mutate(ind = row_number())
           %>%mutate(AAA=neighbour_list_1)
           %>%mutate(mean_cal = mean_cal_1)
           %>%filter(!is.na(mean_cal))
           %>%mutate(ATE = (Y-mean_cal)*ifelse(A==0,-1,1))

  )

  ATE_low <- append(ATE_low,mean(df_1$ATE))
  pairs_low <- append(pairs_low,sum(!is.na(unlist(neighbour_list_1)))/2)
}


####Matching Main high-Dim


a_h <- as.vector(dt2)

seq = 100:5000/100000

ATE_high <- vector("double")
pairs_high <- vector("double")
for (percentage in seq){

  threshold <- quantile(a_h,percentage)
```

```
  n2_vec <- 1:n2
  list_2 <- lapply(n2_vec,cal_neighbour,df=dt2,thresh = threshold,y=hy,A=htr)
  mean_list_2 <- lapply(n2_vec,function(x) unlist(list_2[[x]][1]))
  mean_cal_2 <- unlist(mean_list_2)
  neighbour_list_2 <- lapply(n2_vec,function(x) unlist(list_2[[x]][2]))

  df_2 <- (data.frame(Y=hy,A=htr)
           %>%mutate(ind = row_number())
           %>%mutate(AAA=neighbour_list_2)
           %>%mutate(mean_cal = mean_cal_2)
           %>%filter(!is.na(mean_cal))
           %>%mutate(ATE = (Y-mean_cal)*ifelse(A==0,-1,1))

  )

  ATE_high <- append(ATE_high,mean(df_2$ATE))
  pairs_high <- append(pairs_high,sum(!is.na(unlist(neighbour_list_2)))/2)
}
```

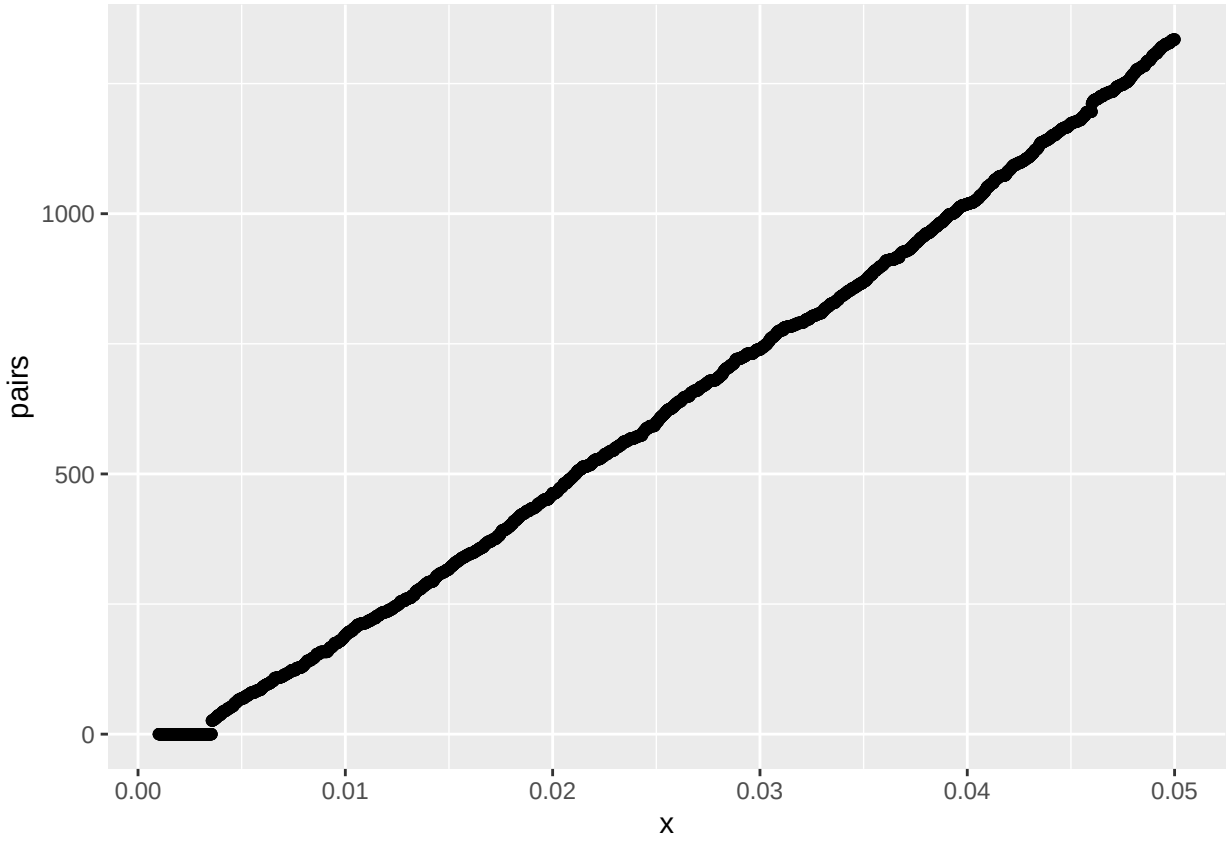
5.2 Plotting Part

```
#########################
plot_low <- data.frame(x=seq,ATE=ATE_low,pairs=pairs_low)
g_low <- ggplot(plot_low)+
  geom_point(aes(x,ATE))
g_low
```

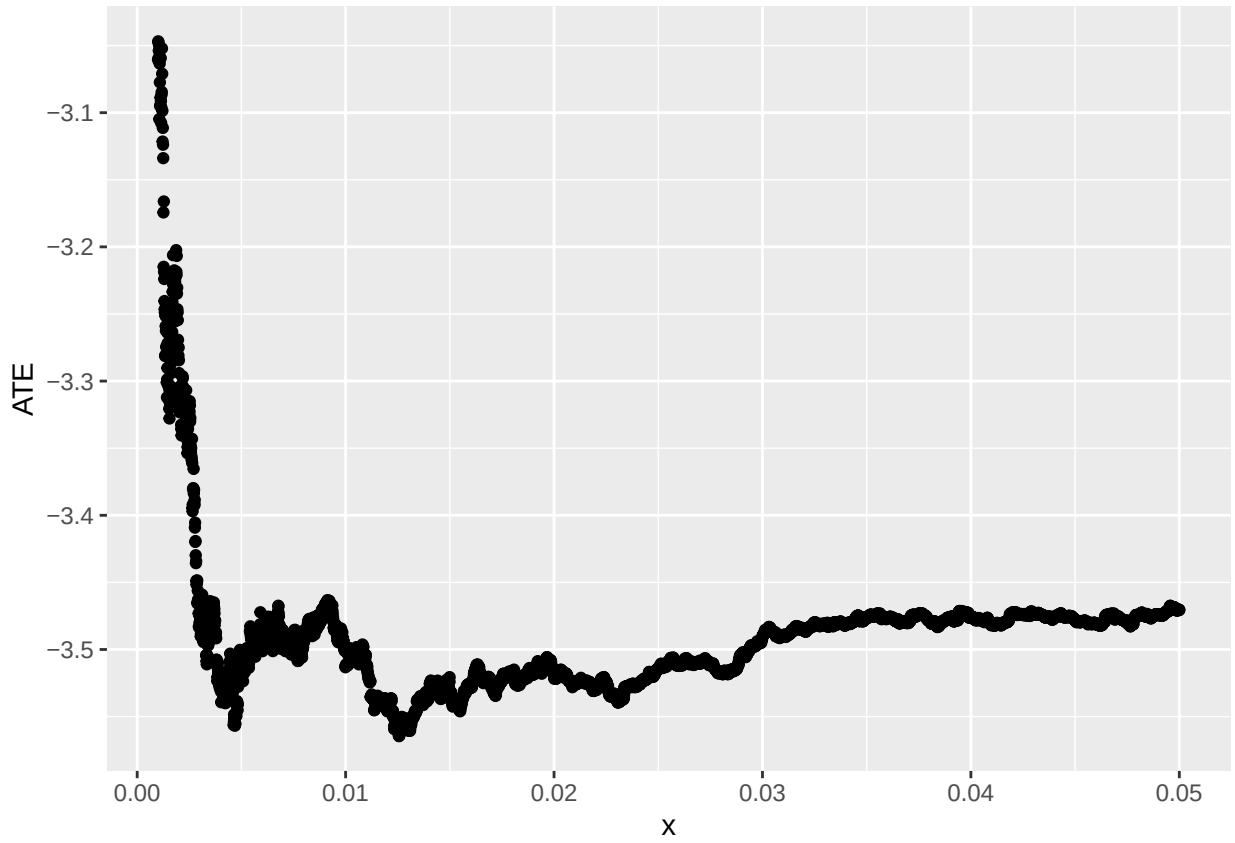
```
## Warning: Removed 256 rows containing missing values (geom_point).
```



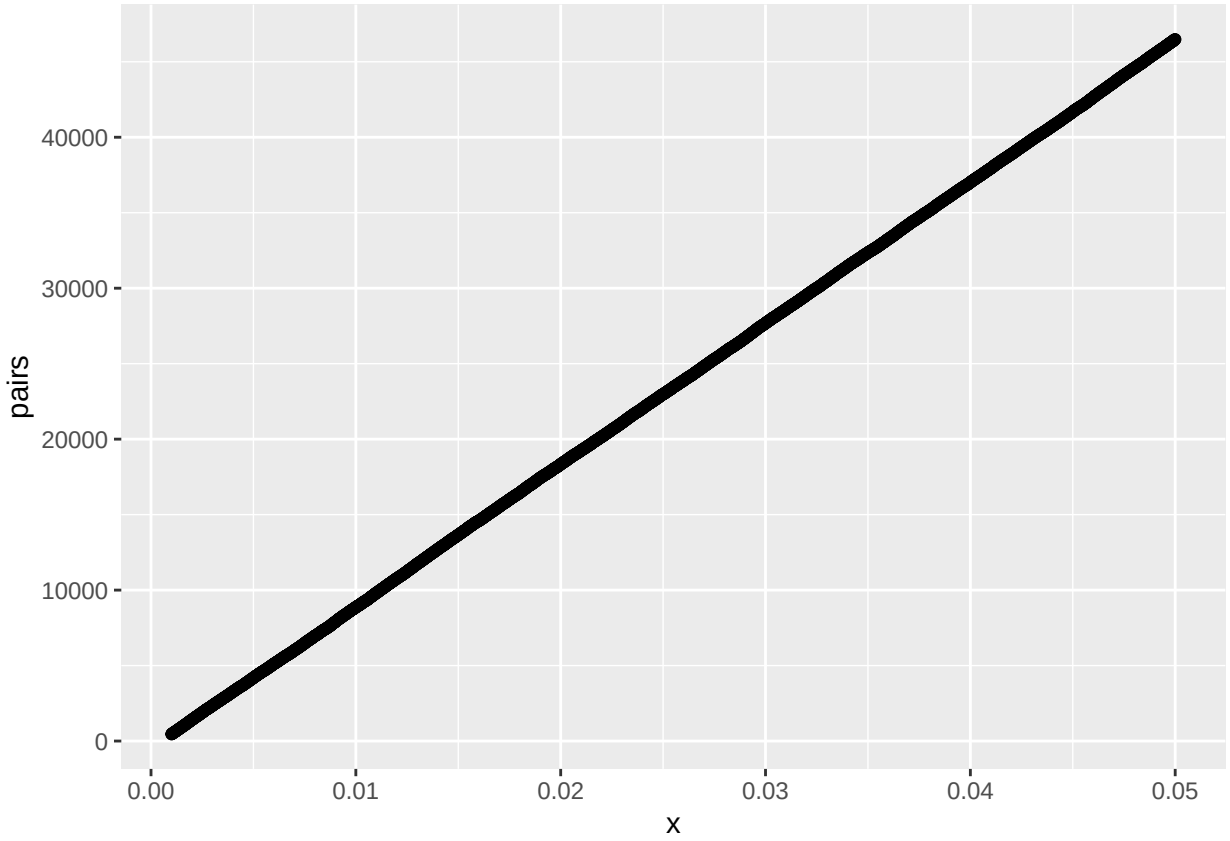

```
match_low <- ggplot(plot_low)+
  geom_point(aes(x,pairs))
match_low
```




```
#########################
plot_high <- data.frame(x=seq,ATE=ATE_high,pairs=pairs_high)
g_high <- ggplot(plot_high)+
  geom_point(aes(x,ATE))
g_high
```




```
match_high <- ggplot(plot_high)+
  geom_point(aes(x,pairs))
match_high
```


40000
30000
20000
10000
0
pairs
0.00
0.01
0.02
0.03
0.04
0.05
x