

Project 4: Causal Inference Algorithms Evaluation

Group 4

Jingbin Cao, Pin-Chun Chen, Aurore Gosmant, Weiwei Song, Zikun Zhuang

Basic Setup

```
#Test Branch created
if(!require("readr")){
  install.packages("readr")
}
if(!require("tidyverse")){
  install.packages("tidyverse")
}
if(!require("glmnet")){
  install.packages("glmnet")
}
if(!require("pryr")){
  install.packages("pryr")
}

library(readr)
library(tidyverse)
library(glmnet)
library(pryr)

lowDim_raw <- read_csv('../data/lowDim_dataset.csv')
highDim_raw <- read_csv('../data/highDim_dataset.csv')

lowDim <- lowDim_raw
highDim <- highDim_raw
```

Project Overview

We will evaluate four inference algorithm in this project, Inverse Propensity Weighting (IPW) + Logistic Regression, Regression Estimate, Stratification + Logistic Regression, and Regression Adjustment + Logistic Regression. We will compute the average treatment effect (ATE) using the four algorithms on two distinct datasets (low dimension & high dimension), and we will compare their performance and computational efficiency.

Getting Propensity Scores

By definition, a **propensity score** is the probability of a unit being assigned to a particular treatment given a set of observed covariates, and the purpose to use propensity scores is to reduce selection bias by equating groups based on those covariates. In this project, we estimate the propensity scores using logistic regression:

$$\text{logit}[Pr(T = 1|X)] = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p$$
$$Pr(T = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p)}}$$

In R, we used the `glm` function to build the logistic regression model based on both High-Dimension and Low-Dimension Dataset, and then we estimated propensity scores for both datasets based on the model. Finally, we did some exploratory data analysis for the propensity scores, summarizing the means and the distributions of them.

```
# High Dimensional Data
ps_high_estimate <- glm(data = highDim,
                        formula = A ~ . - Y,
                        family = binomial())

ps_high_data <- data.frame(ps = predict(ps_high_estimate, type = "response"),
                          treatment = ps_high_estimate$model$A)

# Low Dimensional Data
ps_low_estimate <- glm(data = lowDim,
                      formula = A ~ . - Y,
                      family = binomial())

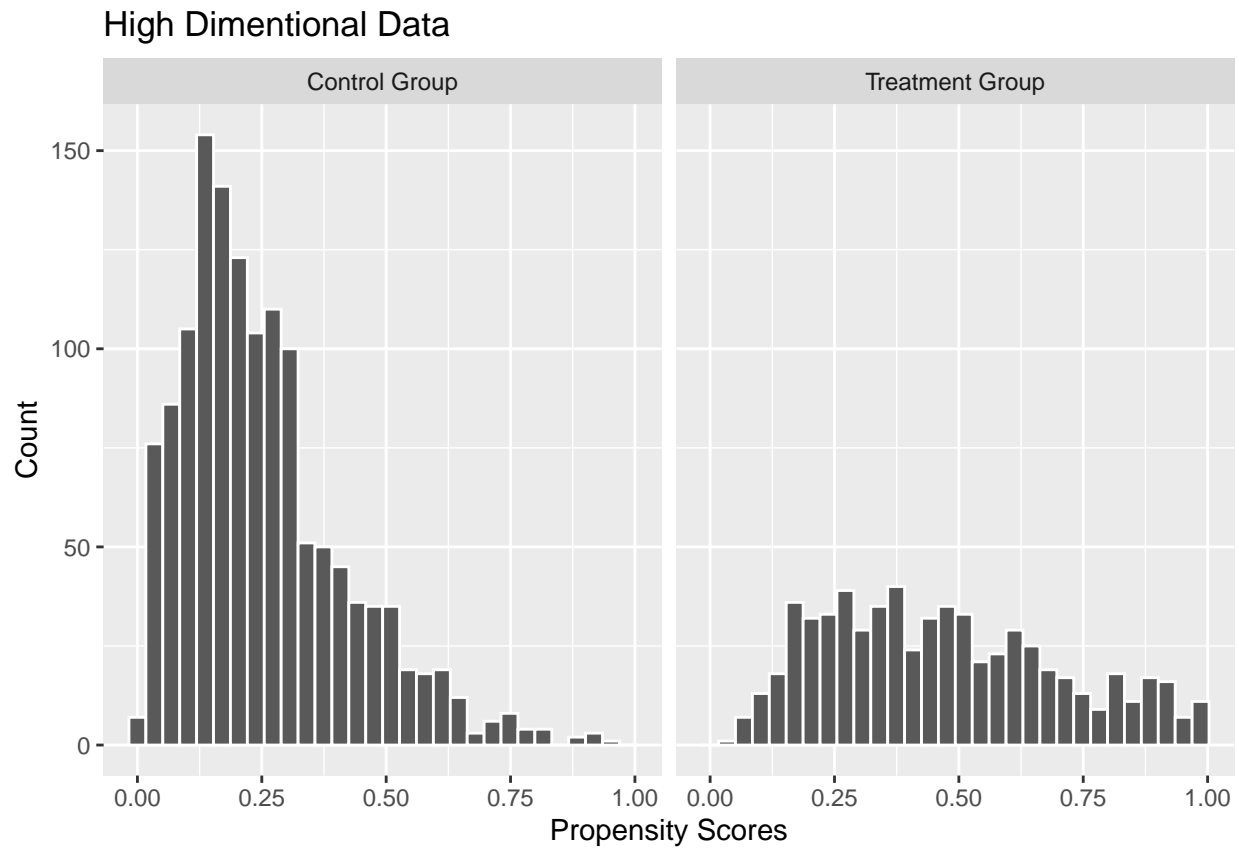
ps_low_data <- data.frame(ps = predict(ps_low_estimate, type = "response"),
                          treatment = ps_low_estimate$model$A)

# Visualize Data
## Show PS means
(ps_means <- data.frame(high_treat_ps_mean = mean(ps_high_data[ps_high_data$treatment == 1,]$ps),
                        high_control_ps_mean = mean(ps_high_data[ps_high_data$treatment == 0,]$ps),
                        low_treat_ps_mean = mean(ps_low_data[ps_low_data$treatment == 1,]$ps),
                        low_control_ps_mean = mean(ps_low_data[ps_low_data$treatment == 0,]$ps)))

##   high_treat_ps_mean high_control_ps_mean low_treat_ps_mean
## 1          0.4663911          0.2528449          0.3290442
##   low_control_ps_mean
## 1          0.1805109

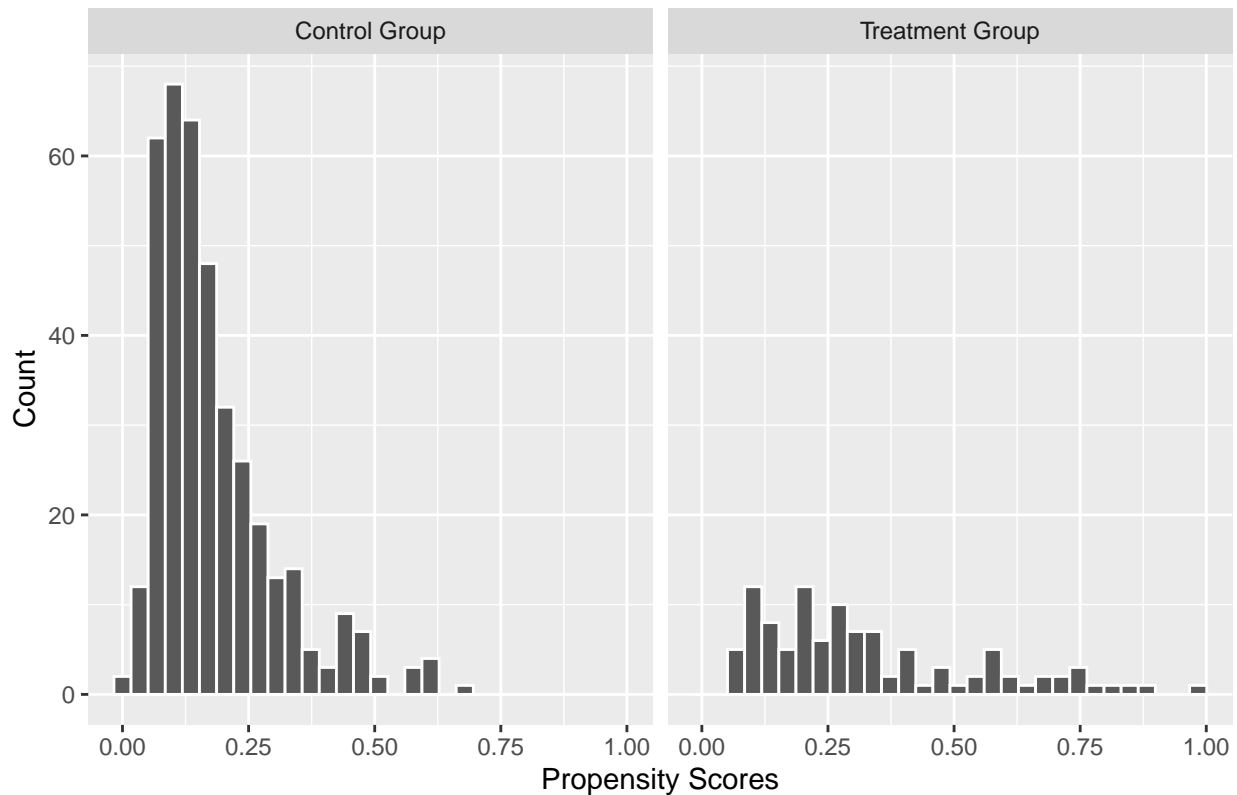
## Plot Counts
ps_high_data %>%
  mutate(treatment = ifelse(treatment == 0, "Control Group", "Treatment Group")) %>%
  ggplot(aes(x = ps)) +
  ggtitle("High Dimensional Data") +
  xlab("Propensity Scores") +
  ylab("Count") +
```

```
geom_histogram(color="white")+
facet_wrap(~treatment)
```



```
ps_low_data %>%
  mutate(treatment = ifelse(treatment == 0, "Control Group", "Treatment Group")) %>%
  ggplot(aes(x=ps))+
  ggtitle("Low Dimentional Data")+
  xlab("Propensity Scores")+
  ylab("Count")+
  geom_histogram(color="white")+
  facet_wrap(~treatment)
```

Low Dimentional Data



Model 1: Inverse Propensity Weighting and Logistic Regression

Inverse Propensity score is one method of propensity score weighting, which is an alternative way to propensity score matching. It can effectively remove the bias by re-weighting the data with weights inversely proportional to the probability of selection. Averaging with known inverse sampling weights was introduced by Horvitz and Thompson in 1952 and has been further studied in recent KDD papers.

$$w_i = \frac{T_i}{\hat{e}_i} + \frac{1 - T_i}{1 - \hat{e}_i}$$

the sum of the weights over the controls equals one. where \hat{e}_i is the estimated propensity score for individual i .

Estimate ATE:

$$\hat{\Delta}_{IPW} = N^{-1} \left(\sum_{i \in Treated} w_i Y_i - \sum_{i \in Controlled} w_i Y_i \right)$$

where the first summation is from the treated group, and the second summation is from the controlled group. However, the IPW will have high variance if $p(X)$ is close to one for some controls. The propensities close to one arise if X nearly separates the controls and exposed or the algorithm used to estimate the propensities is unstable.

```
set.seed(0)
# Define Data

# Write Algorithm
```

```

IPW <- function(df,ps){
  start <- Sys.time()
  ps['weights'] <- df$A/ps$ps+(1-df$A)/(1-ps$ps)
  treatment <- sum(ps[ps$treatment==1,]$weights*df$Y[df$A==1])
  controll <- sum(ps[ps$treatment==0,]$weights*df$Y[df$A==0])
  ATE <- (treatment-controll)/nrow(df)
  end <- Sys.time()
  runtime = end - start
  return(list(ATE=ATE, runtime=runtime))
}
# Output Performance
matrix(c(IPW(highDim,ps_high_data)$ATE,
         IPW(lowDim,ps_low_data)$ATE,
         IPW(highDim,ps_high_data)$runtime,
         IPW(lowDim,ps_low_data)$runtime),
       nrow = 2, byrow = TRUE,
       dimnames = list(c("ATE", "Running Time (secs)"), c("High Dimension", "Low Dimension")))

```

```

##                High Dimension Low Dimension
## ATE            -57.57303      1.689329
## Running Time (secs)      0.00000      0.000000

```

Model 2: Regression Estimate

In this section, we estimated ATE by using a simple regression, and it does not need propensity score. Based on Chan et al. (2010),

$$\hat{\Delta}_{reg_{ATT}} = n_1^{-1} \sum_{exposed} (\hat{m}_1(X_i) - \hat{m}_0(X_i)), n_1 = \sum_{i=1}^n Z_i$$

Here, they estimated ATT, but we are interested in ATE, so the formula that we used is below:

$$\hat{\Delta}_{reg_{ATE}} = N^{-1} \sum_{i=1}^N (\hat{m}_1 - \hat{m}_0(X_i))$$

where \hat{m}_1 is the model that using A=1 (Treatment Group), and \hat{m}_0 is the model that using A=0 (Controlled Group).

```

set.seed(0)
# Write Algorithm
Regression_Estimate <- function(df){
  start <- Sys.time()
  model_0 <- glm(Y ~ ., data = subset(df[df$A==0,], select = -A))
  model_1 <- glm(Y ~ ., data = subset(df[df$A==1,], select = -A))
  ATE = 1/nrow(df) * sum(predict(model_1, newdata = df%>% select(-Y,-A)) -
                        predict(model_0, newdata = df%>% select(-Y,-A)))
  end <- Sys.time()
  runtime = end - start
  return(list(ATE = ATE, runtime = runtime))
}

```

```
# Output Performance
matrix(c(Regression_Estimate(highDim)$ATE,
      Regression_Estimate(lowDim)$ATE,
      Regression_Estimate(highDim)$runtime,
      Regression_Estimate(lowDim)$runtime),
      nrow = 2,
      byrow = TRUE,
      dimnames = list(c("ATE", "Running Time (secs)"), c("High Dimension", "Low Dimension")))
```

```
##                High Dimension Low Dimension
## ATE                -57.426587      2.125138
## Running Time (secs)    0.173368      0.000000
```

Model 3: Stratification and Logistic Regression

$$\hat{\Delta}_S = \sum_{j=1}^K \frac{N_j}{N} (N_{1j}^{-1} \sum_{i=1}^N T_i Y_i I(\hat{e}_i \in \hat{Q}_j) - N_{0j}^{-1} \sum_{i=1}^N (1 - T_i) Y_i I(\hat{e}_i \in \hat{Q}_j))$$

where K is the number of strata ($K=5$).

N_j is the number of individuals in stratum j .

N_{1j} is the number of “treated” individuals in stratum j , and N_{0j} is the number of “controlled” individuals in stratum j .

$\hat{Q}_j = (\hat{q}_{j-1}, \hat{q}_j)$ where \hat{q}_j is the j th sample quantile of the estimated propensity scores.

Estimation of treatment effects with causal interpretation from raw data is hard because exposure to treatment may be confounded with subject characteristics. Therefore, we need to use stratification to adjust for confounding.

In our implementation, we can select k where k represented the number of the strata. With the help of strata, we can first calculate the data which have similar properties from the propensity score aspect. We also tested which K value in the formula above has the best performance. In the last part of this section, from the graph, we can see that $K=8$ should provide the best result (lowest squared error for both high and low dimensional datasets).

```
set.seed(0)
# Define Data

# Write Algorithm
Strat <- function(df, ps, k=8){
  start = Sys.time()
  ATE <- 0
  prev <- 0
  concatdata <- cbind(df, ps)
  for (i in 1:k) {
    ps_str = as.numeric(ps$ps)
    str_q <- quantile(ps_str, (1/k)*i)
    str_data <- concatdata[concatdata$ps>=prev,]
    str_data <- str_data[str_q>=str_data$ps,]
    prev = str_q
    temp_t <- sum(str_data$Y[str_data$A==1])
    temp_t <- 1/nrow(str_data[str_data$A==1,])*temp_t
    temp_c <- sum(str_data$Y[str_data$A==0])
```

```

    temp_c <- 1/nrow(str_data[str_data$A==0,])*temp_c
    str_e <- temp_t - temp_c
    ATE <- ATE + str_e*nrow(str_data)/nrow(df)
  }
  end = Sys.time()
  runtime = end - start
  return(list(ATE = ATE, runtime = runtime))
}

# Output Performance
stra_result_high <- matrix(0, nrow = 10, ncol = 4,
                           dimnames = list(c("1","2","3","4","5","6","7","8","9","10"),
                                             c("K","ATE","Running Time (secs)","Sq_Error")))
stra_result_low <- matrix(0, nrow = 10, ncol = 4,
                           dimnames = list(c("1","2","3","4","5","6","7","8","9","10"),
                                             c("K","ATE","Running Time (secs)","Sq_Error")))

for (i in 1:10){
  stra_result_high[i,] <- c(i,
                           Strat(highDim,ps_high_data, i)$ATE,
                           Strat(highDim,ps_high_data, i)$runtime,
                           sqrt(abs(-54.8558-Strat(highDim,ps_high_data, i)$ATE)))
  stra_result_low[i,] <- c(i,
                           Strat(lowDim,ps_low_data, i)$ATE,
                           Strat(lowDim,ps_low_data, i)$runtime,
                           sqrt(abs(2.0901-Strat(highDim,ps_high_data, i)$ATE)))
}

stra_result_high <- as.data.frame(stra_result_high)
stra_result_low <- as.data.frame(stra_result_low)
matrix(c(Strat(highDim,ps_high_data)$ATE,
          Strat(lowDim,ps_low_data)$ATE,
          Strat(highDim,ps_high_data)$runtime,
          Strat(lowDim,ps_low_data)$runtime),
        nrow = 2,
        byrow = TRUE,
        dimnames = list(c("ATE","Running Time (secs)",
                           c("High Dimension","Low Dimension"))))

```

```

##                High Dimension Low Dimension
## ATE            -58.98685168    2.026394882
## Running Time (secs)    0.07281685    0.008979082

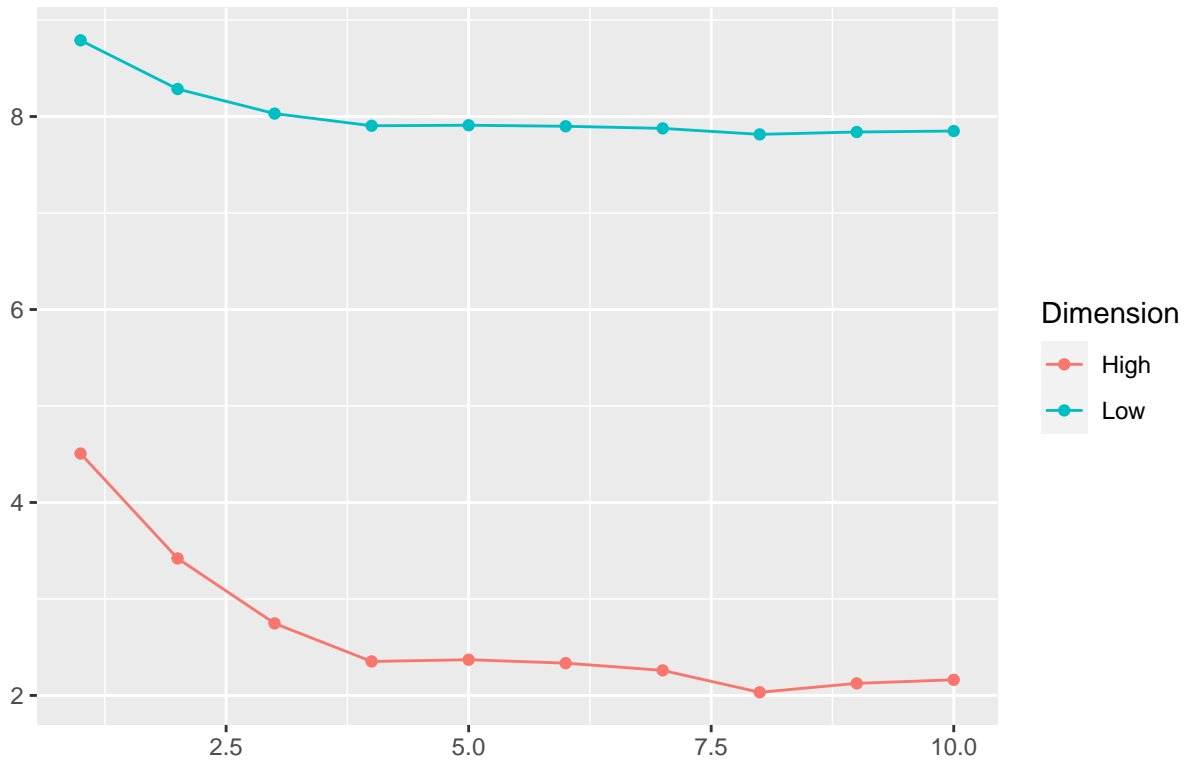
```

```

# Plot Different K
Summary_K <- rbind(K_low <- stra_result_low %>% mutate(Dimension = "Low"),
                   K_high <- stra_result_high %>% mutate(Dimension="High"))
ggplot(Summary_K, aes(K, Sq_Error, group=Dimension, col = Dimension)) +
  geom_point() +
  geom_line()+
  labs(title='Squared Error of Different Ks in Stratification',
        x='',
        y='',
        col='Dimension')

```

Squared Error of Different Ks in Stratification



Model 4: Regression Adjustment and Logistic Regression

Regress the outcome variable Y on treatment indicator variable T and the estimated propensity score. Then, the estimated coefficient on the treatment indicator variable would be an estimate of ATE.

In this method, we regress the response variable (Y) with the treatment variable (A) and the propensity scores estimated using our model above, in this case, logistic regression. The estimated coefficient of the treatment variable (A) is then an estimate of the ATE.

D'Agostino (1998) and Austin (2011) compare regression adjustment with more traditional propensity score methods. One of the main advantages of the regression adjustment is in its simplicity in execution, in which one performs a somewhat basic linear regression model on two covariates and one response variable.

However, depending on the size of the dataset, this may run into computation issues as linear regression involves finding the inverse of a matrix. Additionally, regression adjustment may also not be helpful in cases where there is a strong separation between the two groups.

No such issues were present in this setup given that both datasets had a relatively small number of observations and there is no clear separation between the two groups

```
set.seed(0)
# Define Data

# Write Algorithm
Reg_adj <- function(df,ps_data){
  df<- data.frame(cbind(Y=df$Y,A=df$A,ps=ps_data$ps))
  start <- Sys.time()
```



```

m<- lm(Y ~ A+ ps, data = df)
ATE = m$coefficients[2]
end <- Sys.time()
runtime = end - start
return(list(ATE = ATE, runtime = runtime))
}
# Output Performance
matrix(c(Reg_adj(highDim,ps_high_data)$ATE,
        Reg_adj(lowDim,ps_low_data)$ATE,
        Reg_adj(highDim,ps_high_data)$runtime,
        Reg_adj(lowDim,ps_high_data)$runtime),
      nrow = 2,
      byrow = TRUE,
      dimnames = list(c("ATE","Running Time (secs)",
                        c("High Dimension","Low Dimension"))))

```

```

##                High Dimension Low Dimension
## ATE            -59.416700238  2.4180864541
## Running Time (secs)    0.001040936  0.0009970665

```

Model Comparisons

Performance is measured by the squared difference of true ATE and estimated ATE Run times are in seconds

Low Dimension Dataset

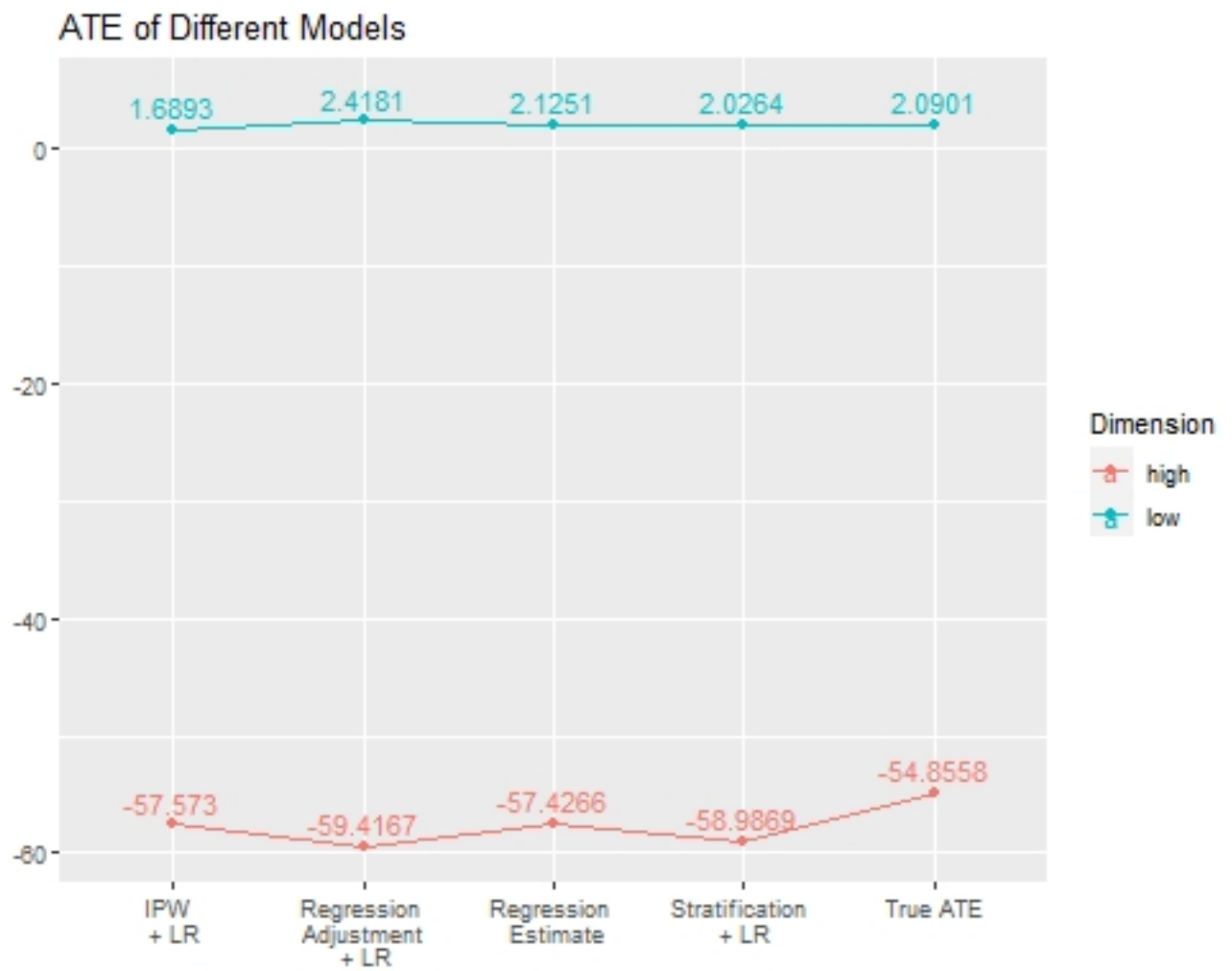
Model	ATE	Run_Time	Squared_Error
True ATE	2.0901	N.A	N.A.
IPW + LR	1.6893	0	0.6331
Regression Estimate	2.1251	0.01197004	0.1872
Stratification + LR	2.0264	0.00697899	0.2524
Regression Adjustment + LR	2.4181	0.00099993	0.5727

High Dimension Dataset

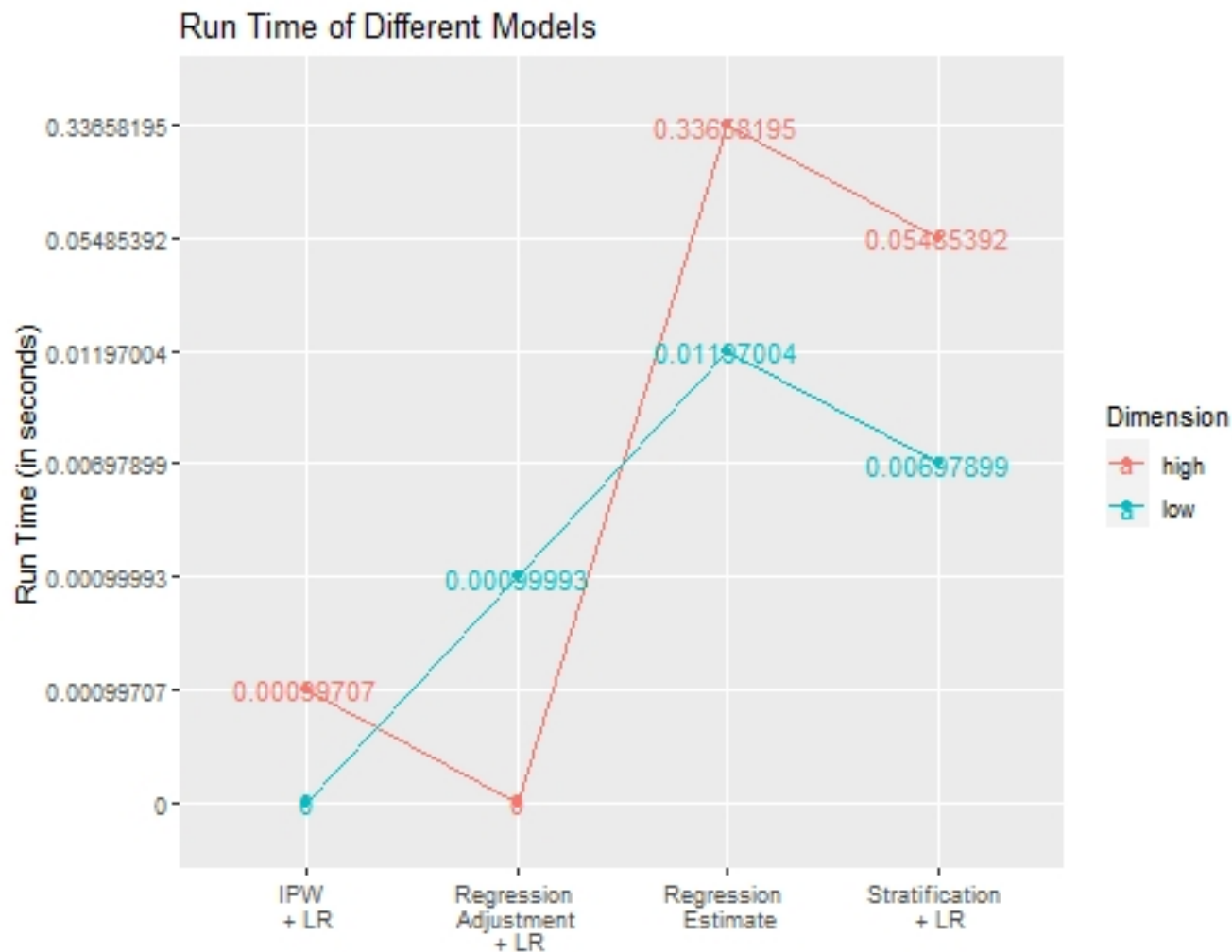
Model	ATE	Run_Time	Squared_Error
True ATE	-54.8558	N.A	N.A.
IPW + LR	-57.5730	0.00099707	1.6484
Regression Estimate	-57.4266	0.33658195	1.6034
Stratification + LR	-58.9869	0.05485392	2.0325
Regression Adjustment + LR	-59.4167	0	2.1356

Regression Estimate has the best performance on Low Dimension Dataset; Regression Estimate has the best performance on High Dimension Dataset.

Plotting the Result







Reference

- Austin, Peter C. 2011. “An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies.” *Multivariate Behavioral Research* 46 (3): 399–424.
- Chan, David, Rong Ge, Ori Gershony, Tim Hesterberg, and Diane Lambert. 2010. “Evaluating Online Ad Campaigns in a Pipeline: Causal Models at Scale.” In *Proceedings of the 16th Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 7–16.
- Lunceford, Jared K, and Marie Davidian. 2004. “Stratification and Weighting via the Propensity Score in Estimation of Causal Treatment Effects a Comparative Study.” *Statistics in Medicine* 23 (19): 2937–60.
- Stuart, Elizabeth A. 2010. “Matching Methods for Causal Inference: A Review and a Look Forward.” *Statistical Science: A Review Journal of the Institute of Mathematical Statistics* 25 (1): 1.