Directly choosing Γ and Fair Policy Learning

19331053 纪传宇

2023 年 7 月 31 日

The doubly robust score (Robins and Rotnitzky, 1995).

$$\Gamma_{d,s,i} = \frac{1\{S_i = s\}}{p_s} \left[\frac{1\{D_i = d\}}{e(X_i, S_i)} (Y_i - m_{d,s}(X_i)) + m_{d,s}(X_i) \right] \tag{1}$$

can be estimated by $\hat{\Gamma}_{d,s,i}$ using $\hat{p}_1$ and $\hat{e}(x,s)$, $\hat{m}_{d,s}(.)$.

```
D <- data$accepted
S <- data$female
propensity2 <- mean(S)
Y <- data$short_run_activity
library(glmnet)
propensity1 <- cross_fitting_propensity(D,as.matrix(data[, c(5,
   6,7,8, 9, 10, 11, 12, 13, 14)]), seeds = 123, K = 5)
#propensity1 is e(x,s)
#propensity2 is p(s=1)
X <- data[,c(5,8,9,10,11, 12, 13, 14)]
X_int <- X[, c(3,4,5,6, 7, 8)]
X_reg <- cbind(X, D, D * X_int * S, D * X_int * (1 - S), S * X_
   int, D * X_int) #选择协变量
conditional_means <- cross_fitting_mean(Y, X_reg, X, X_int,
   seeds = 123, K = 5) #治疗d下组s的条件平均值
m_hat11 <- conditional_means[,1]#就是定义的m(d,s)
m_hat10 <- conditional_means[,2]
m_hat01 <- conditional_means[,3]
m_hat00 <- conditional_means[,4]
tau_hat1 <- m_hat11 - m_hat10  #d=1 ,s=1,0
tau_hat0 <- m_hat01 - m_hat00 #d=0, s=1,0
difference <- abs(m_hat11 - m_hat01)  - abs(m_hat10 - m_hat00)
   #s相同 d不同的绝对差异
difference <-difference * S/propensity2 + difference * (1 - S)
   /(1 - propensity2) #
library(foreach)
library(doParallel)
numcores <- 11
```

```
registerDoParallel(numcores)
cost_treatment=0;
m1 <- S*m_hat11 + (1 - S)*m_hat01
m0 <- S*m_hat10 + (1 - S)*m_hat00

G_i1 <- (Y - m0) * (1-D) * S / ((1-propensity1)*propensity2) +
    m0 * S/propensity2
G_i2 <- (Y - cost_treatment - m1) * D * S/ (propensity1*
    propensity2) + m1 * S/propensity2
g_i_S = G_i2 - G_i1

G_i12 <- (Y - m0) * (1-D) * (1 - S) / ((1-propensity1)*(1 -
    propensity2)) + m0 * (1 - S)/(1 - propensity2)
G_i22 <- (Y - cost_treatment - m1) * D * (1 - S)/ (propensity1
    *(1 - propensity2)) + m1 * (1 - S)/(1 - propensity2)
g_i_S2 = G_i22 - G_i12
```

1 Simulation

In this scenario, We are planning to use simulated data to compare the results of candidate selection using the Fair Policy Targeting approach with the direct application of gamma calculations.

In our data generating process, $\mathbf{X}_i = (X_{i1}, X_{i2}, X_{i3}, X_{i4})^T \in \mathbb{R}^3$ are i.i.d. samples where $X_{i,1} \sim \mathrm{B}(1, 0.2)$, $X_{i,2} \sim \mathrm{unif}(-2, 2)$,$X_{i,3} \sim \mathrm{unif}(-1, 1)$,$X_{i,4} \sim \mathrm{B}(1, 0.5)$ is a scalar for $j \in \{1, 2, 3, 4\}$.W_i is a binary treatment variable that follows:

$$\frac{P(W_i = 1 \mid \mathbf{X}_i)}{P(W_i = 0 \mid \mathbf{X}_i)} = \exp\left(\gamma_c + \gamma_x^T \mathbf{X}_i\right)$$

where $\gamma_c = 0.1$ and $\gamma_x = [0.2, 0.3, 0.4, 0.5]$.$Y_i$ is a binary response variable that follows

$$\frac{P(Y_i = 1 \mid \mathbf{X}_i, W_i)}{P(Y_i = 0 \mid \mathbf{X}_i, W_i)} = \exp\left(\beta_c + \beta_w W_i + \beta_x^T \mathbf{X}_i\right)$$

where $\beta_c = -0.2, \beta_w = -0.5, \beta_x = [0.6, 0.7, -0.6, 0.1]$.

```
N = 200
estimand = "ATE"
gamma <- c(0.1, 0.2, 0.3, 0.4, 0.5)
beta <- c(-0.2, -0.5, 0.6, 0.7, -0.6, 0.1)

expit <- function(x) {
    return(1/(1 + exp(-x)))
}
set.seed(123)
```

```
X1 <- rbinom(n=N,size=1,prob=1/5)
X2 <- runif(N, min = -2, max = 2)
X3 <- runif(N, min = -1, max = 1)
X4 <- rbinom(n=N,size=1,prob=1/2)
treat <- rbinom(N, 1, expit(gamma[1]+gamma[2]*X1+gamma[3]*X2+gamma[4]*X3+
    gamma[5]*X4))
y0 <- rbinom(N, 1, expit(beta[1]+beta[2]*0+beta[3]*X1+beta[4]*X2+beta[5]*X3
    +beta[6]*X4))
y1 <- rbinom(N, 1, expit(beta[1]+beta[2]*1+beta[3]*X1+beta[4]*X2+beta[5]*X3
    +beta[6]*X4))
y <- y1*treat + y0*(1-treat)
intercept = rep(1, N)
data <- data.frame(intercept, y, y1, y0, treat, X1, X2, X3, X4)
```

1.1 Using Γ to choose people

First, we rank individuals based on their Γ_i values, and then select the top 30 individuals with the highest Γ_i values (since the maximum number of treatments is 30).

```
D <- data$treat
S <- data$X1
propensity2 <- mean(S)
Y <- data$y
library(glmnet)
propensity1 <- cross_fitting_propensity(D,as.matrix(data[, c(7,8)]), seeds
    = 123, K = 5)
#propensity1 is e(x,s)
#propensity2 is p(s=1)
X <- data[,c(7,8,9)]
X_int <- data[,c(6,9)]
X_reg <- cbind(X, D, D * X_int * S, D * X_int * (1 - S), S * X_int, D * X_
    int) #选择协变量
conditional_means <- cross_fitting_mean(Y, X_reg, X, X_int, seeds = 123, K
    = 5) #治疗d下组s的条件平均值
m_hat11 <- conditional_means[,1]#就是定义的m(d,s)
m_hat10 <- conditional_means[,2]
m_hat01 <- conditional_means[,3]
m_hat00 <- conditional_means[,4]
tau_hat1 <- m_hat11 - m_hat10  #d=1 ,s=1,0
tau_hat0 <- m_hat01 - m_hat00 #d=0, s=1,0
difference <- abs(m_hat11 - m_hat01)  - abs(m_hat10 - m_hat00) #s相同 d不同
    的绝对差异
difference <-difference * S/propensity2 + difference * (1 - S)/(1 -
    propensity2) #
```

```
cost_treatment=0;
m1 <- S*m_hat11 + (1 - S)*m_hat01
m0 <- S*m_hat10 + (1 - S)*m_hat00

G_i1 <- (Y - m0) * (1-D) * S / ((1-propensity1)*propensity2) + m0 * S/
   propensity2
G_i2 <- (Y - cost_treatment - m1) * D * S/ (propensity1*propensity2) + m1 *
    S/propensity2
g_i_S = G_i2 - G_i1

G_i12 <- (Y - m0) * (1-D) * (1 - S) / ((1-propensity1)*(1 - propensity2)) +
    m0 * (1 - S)/(1 - propensity2)
G_i22 <- (Y - cost_treatment - m1) * D * (1 - S)/ (propensity1*(1 -
   propensity2)) + m1 * (1 - S)/(1 - propensity2)
g_i_S2 = G_i22 - G_i12

total=vector()
for(i in 1:length(D)){
    if (g_i_S[i]!=0){
            total[i]=g_i_S[i]
    }
    else{
            total[i]=g_i_S2[i]
    }
}
```

1.2 Using Fair Policy Learning Method

We consider three notions of UnFairness: **Counterfactual Envy** Let the conditional welfare, for the policy function being assigned to the opposite attribute, i.e., the effect of $\pi(x, s_1)$, on the group s_2 , conditional on covariates, be

$$V_{\pi(x,s_1)}(x, s_2) = \mathbb{E}\left[\pi(x, s_1) Y_i(1, s_2) + (1 - \pi(x, s_1)) Y_i(0, s_2) \mid X_i(s_2) = x\right] \tag{2}$$

We say that the agent with attribute s_2 envies the agent with attribute s_1, if her welfare (on the right-hand side of Equation (3)) exceeds the welfare she would have received had her covariate and policy been assigned the opposite attribute (left-hand side of Equation (3)), namely

$$\mathbb{E}_{X(s_1)}\left[V_{\pi(X(s_1),s_1)}(X(s_1), s_2)\right] > \mathbb{E}_{X(s_2)}\left[V_{\pi(X(s_2),s_2)}(X(s_2), s_2)\right] \tag{3}$$

We then measure the unfairness towards an individual with attribute s_2 as

$$\mathcal{A}(s_1, s_2; \pi) = \mathbb{E}_{X(s_1)}\left[V_{\pi(X(s_1),s_1)}(X(s_1), s_2)\right] - \mathbb{E}_{X(s_2)}\left[V_{\pi(X(s_2),s_2)}(X(s_2), s_2)\right]. \tag{4}$$

Whenever we aim not to discriminate in either direction, we take the sum of the effects $\mathcal{A}(s_1, s_2; \pi)$ and $\mathcal{A}(s_2, s_1; \pi)$ it connects to previous notions of counterfactual fairness (Kilbertus et al., 2017).

Predictive Disparity Prediction disparity and its empirical counterpart take the following form

$$C(\pi) = \mathbb{E}[\pi(X,S) \mid S=0] - \mathbb{E}[\pi(X,S) \mid S=1], \quad \hat{C}(\pi) = \frac{\sum_{i=1}^{n} \pi(X_i)(1-S_i)}{n(1-\hat{p}_1)} - \frac{\sum_{i=1}^{n} \pi(X_i) S_i}{n\hat{p}_1}, \tag{5}$$

Prediction disparity captures disparity in the treatment probability between groups.

(Welfare disparity). Define the welfare disparity and its empirical counterpart as

$$D(\pi) = W_0(\pi) - W_1(\pi), \quad \widehat{D}(\pi) = \widehat{W}_0(\pi) - \widehat{W}_1(\pi). \tag{6}$$

```
numcores = 4
discretization = 9
alpha_seq = seq(from = 0.05, to = 0.95, length = discretization)
## Estimate on both sides the Pareto frontier

Y = Y
X = cbind(S, X[, c(1:3)])
D = D
S = S
propensity1 = propensity1
propensity2 = propensity2
scale_Y = F
discretization = discretization
cost_treatment = 0
params=NA
mu_hat11 = m_hat11
mu_hat01 = m_hat01
mu_hat00 = m_hat00
mu_hat10 = m_hat10
maxtime1 = 5000
maxtime2 = 5000
max_treated_units = treatnum
alpha_seq = seq(from = 0.05, to = 0.95, length = discretization)
m1 = m1
m0 = m0
quick_run = F
no_parity_constraint = T
additional_fairness_constraint = F
noparity_constraint = F
parity_constraint = '>='
distance = 'envy'
```

```
model_only = F
quick_run = F
no_parity_constraint = F
additional_fairness_constraint = F
parity_constraint = '>='
frontier = NA
unique_values = 1 - no_parity_constraint
distance = 'envy'
probabilistic = F
numcores = 4
threshold_probabilistic = F
two_directions = T
parallel = T
tolerance_frontier = 10**(-3)
tolerance_optimization = 10**(-6)
return_frontier = F
library(foreach)

start <- Sys.time()    ## 记录时间

maxtime = 300
tolerance = 10**(-3)
res <- foreach(i = alpha_seq, .combine = rbind, .export = c('Y', 'D', 'X',
    'S', 'propensity1',
'propensity2', 'params', 'scale_Y',
'cost_treatment', 'max_treated_units', 'method',
'maxtime', 'm1', 'm0', 'additional_fairness_constraint',
'parity_constraint'))%do%{
    source('./library/helpers.R')
    library(gurobi)
    if(probabilistic == F & threshold_probabilistic == F){
            result <- Est_max_score(Y, X, D, S, propensity1, propensity2,B=
                1, params = params, tolerance_constraint = tolerance,
            scale_Y = scale_Y, additional_fairness_constraint = additional_
                fairness_constraint,
            parity_constraint = parity_constraint, cost_treatment  = cost_
                treatment, alpha = i,
            max_treated_units = max_treated_units, maxtime = maxtime, m1 =
                m1, m0 = m0, cores = numcores) } else if (probabilistic == T
                 & threshold_probabilistic == F) {
            result <- Est_max_score_probabilistic(Y, X, D, S, propensity1,
                propensity2,B=1, params = params, tolerance_constraint =
                tolerance,
```

```
            scale_Y = scale_Y, additional_fairness_constraint = additional_
               fairness_constraint,
            parity_constraint = parity_constraint,
            cost_treatment  = cost_treatment, alpha = i, max_treated_units
               = max_treated_units, maxtime = maxtime, m1 = m1, m0 = m0,
            cores = numcores)
    } else {   result <- Est_threshold_probabilistic(Y, X, D, S, propensity
        1, propensity2,B=1, params = params, tolerance_constraint =tolerance
        ,
            scale_Y = scale_Y, additional_fairness_constraint = additional_
               fairness_constraint,
            parity_constraint = parity_constraint,
            cost_treatment  = cost_treatment, alpha = i, max_treated_units
               = max_treated_units, maxtime = maxtime, m1 = m1, m0 = m0,
               cores = numcores)

    }
    c(result[[2]], result[[1]],
    result[[3]], result[[4]], length(result[[2]]),
    length(result[[3]]))



}



nn <- res[1, dim(res)[2] - 1]
nn2 <- res[1, dim(res)[2]]
res <- res[, -dim(res)[2]]
aa1 <- res[, 1:nn]
aa2 <- res[, nn + 1]

frontier <- list(g_i = res[, 1:nn], objective = res[, nn + 1], results =
   res[, c((nn + 2):(nn + 1 + nn2))],
policies = res[, c((nn + 2 + nn2):(dim(res)[2]-1))], beta = res[, c((nn + 2
     + nn):(nn + 1 + nn2))])

frontier_objective <- frontier[[2]]
results_frontier <- frontier[[4]] ## Store the policies
results_frontier_collapsed <- frontier[[3]]

G_i1 <- (Y - m0) * (1-D) * S / ((1-propensity1)*propensity2) + m0 * S/
   propensity2
```

```
G_i2 <- (Y - cost_treatment - m1) * D * S/ (propensity1*propensity2) + m1 *
     S/propensity2
g_i_S = G_i2 - G_i1

G_i12 <- (Y - m0) * (1-D) * (1 - S) / ((1-propensity1)*(1 - propensity2)) +
     m0 * (1 - S)/(1 - propensity2)
G_i22 <- (Y - cost_treatment - m1) * D * (1 - S)/ (propensity1*(1 -
    propensity2)) + m1 * (1 - S)/(1 - propensity2)
g_i_S2 = G_i22 - G_i12


all_g_i = g_i_S + g_i_S2 ## Save the welfare criterion

if(threshold_probabilistic == F){
    beta <- frontier[[5]]
    if(unique_values == F){
            XX1 <- as.matrix(cbind(1, 1, X[,-1]))
            XX0 <- as.matrix(cbind(1, 0, X[,-1]))
    } else {
            XX1 <- as.matrix(cbind(1, X))
            XX0 <- as.matrix(cbind(1, X))
    }

    ## Compute a warm-start for the MILP
    policy1 <- t(apply(beta, 1, function(x) sapply(XX1%*%x, function(y)
       ifelse(y > 0, 1, 0))))
    policy0 <- t(apply(beta, 1, function(x) sapply(XX0%*%x, function(y)
       ifelse(y > 0, 1, 0))))


    ## Compute the distance for the welfare-based fairness

    if(distance == 'welfare'){
            welfare1 <- apply(policy1, 1, function(x) sum(g_i_S*x) + sum(G_
               i1))
            welfare0 <- apply(policy0, 1, function(x) sum(g_i_S2*x) + sum(G
               _i12))
            objective_warm_starts_welfare <-  welfare0 - welfare1
            if(two_directions) objective_warm_starts_welfare <- abs(welfare
               1 - welfare0)
            least_unfair = which(objective_warm_starts_welfare == min(
               objective_warm_starts_welfare))
            least_unfair = least_unfair[which.min(abs(least_unfair -
```

```
            discretization/2))]
        indicator <- ifelse(welfare1[least_unfair] -  welfare0[least_unfair] > 0, 1, 0)
        warm_start = c(policy1[least_unfair,], policy0[least_unfair,], beta[least_unfair,], indicator, 1 - indicator, least_unfair)
}

if(distance == 'relative_welfare'){
        welfare1 <- apply(policy1, 1, function(x) sum(g_i_S*x))
        welfare0 <- apply(policy0, 1, function(x) sum(g_i_S2*x))
        objective_warm_starts_welfare <-  welfare0 - welfare1
        if(two_directions) objective_warm_starts_welfare <- abs(welfare1 - welfare0)
        least_unfair = which(objective_warm_starts_welfare == min(objective_warm_starts_welfare))
        least_unfair = least_unfair[which.min(abs(least_unfair - discretization/2))]
        indicator <- ifelse(welfare1[least_unfair] -  welfare0[least_unfair] > 0, 1, 0)
        warm_start = c(policy1[least_unfair,], policy0[least_unfair,], beta[least_unfair,], indicator, 1 - indicator, least_unfair)
}

if(distance == 'parity'){
        w1 <- apply(policy1, 1, function(x) sum(S*x/mean(S)))
        w0 <- apply(policy0, 1, function(x) sum((1 - S)*x/(1 - mean(S))))
        objective_warm_starts_welfare <- w0 - w1
        if(two_directions)  objective_warm_starts_welfare <- abs(w1 - w0)
        least_unfair = which(objective_warm_starts_welfare == min(objective_warm_starts_welfare))
        least_unfair = least_unfair[which.min(abs(least_unfair - discretization/2))]
        indicator <- ifelse(w1[least_unfair] -  w0[least_unfair] > 0, 1, 0)
        warm_start = c(policy1[least_unfair,], policy0[least_unfair,], beta[least_unfair,], indicator, 1 - indicator, least_unfair)

}

## Compute the distance for the envy-based fairness
```

```
    if(distance == 'envy'){
            welfare1 <- apply(policy1, 1, function(x) sum(g_i_S*x))
            welfare0 <- apply(policy0, 1, function(x) sum(g_i_S2*x))
            objective_warm_starts1 <- apply(policy1, 1, function(x) sum(mu_
               hat01*x*S)/propensity2 +  sum(mu_hat00*(1 - x)*S)/propensity
               2) -  welfare0
            objective_warm_starts2 <- apply(policy0, 1, function(x) sum(mu_
               hat11*x*(1 - S))/(1 - propensity2) +  sum(mu_hat10*(1 - x)*(
               1 - S))/(1 - propensity2)) - welfare1
            objective_warm_starts_envy <- objective_warm_starts1 +
               objective_warm_starts2
            least_unfair = which(objective_warm_starts_envy == min(
               objective_warm_starts_envy))
            least_unfair = least_unfair[which.min(abs(least_unfair -
               discretization/2))]
            warm_start = c(policy1[least_unfair,], policy0[least_unfair,],
               beta[least_unfair,], least_unfair)
    }
  } else {
    ## Compute warm start for the probabilistic with threshold (note: warm
       start only imporves computational time but not performance)

    beta <- frontier[[5]]
    probs <- frontier[[6]]
    if(unique_values == F){
            XX1 <- as.matrix(cbind(1, 1, X[,-1]))
            XX0 <- as.matrix(cbind(1, 0, X[,-1]))
    } else {
            XX1 <- as.matrix(cbind(1, X))
            XX0 <- as.matrix(cbind(1, X))
    }

    ## Compute a warm-start for the binary indicator and the probabilistic
       assignments
    xi1 <- t(apply(cbind(beta,probs),
    1, function(x) sapply(XX1%*%(x[1:dim(beta)[2]]), function(y) ifelse(y >
        0, 1, 0))))
    policy1 <- t(apply(cbind(beta,probs),
    1, function(x) sapply(XX1%*%(x[1:dim(beta)[2]]), function(y) ifelse(y >
        0, x[dim(beta)[2] + 1] +
    x[dim(beta)[2] + 2], x[dim(beta)[2] + 2]
    ))))
    xi0 <- t(apply(cbind(beta,probs),
```

```
1, function(x) sapply(XX0%*%(x[1:dim(beta)[2]]), function(y) ifelse(y >
     0, 1, 0))))
policy0 <- t(apply(cbind(beta,probs),
1, function(x) sapply(XX0%*%(x[1:dim(beta)[2]]), function(y) ifelse(y >
     0, x[dim(beta)[2] + 1] +
x[dim(beta)[2] + 2], x[dim(beta)[2] + 2]))))


## Compute the distance for the welfare-based fairness

if(distance == 'welfare'){
        welfare1 <- apply(policy1, 1, function(x) sum(g_i_S*x) + sum(G_
           i1))
        welfare0 <- apply(policy0, 1, function(x) sum(g_i_S2*x) + sum(G
           _i12))
        objective_warm_starts_welfare <- welfare0 - welfare1
        if(two_directions) objective_warm_starts_welfare <- abs(welfare
           1 - welfare0)
        least_unfair = which(objective_warm_starts_welfare == min(
           objective_warm_starts_welfare))
        least_unfair = least_unfair[which.min(abs(least_unfair -
           discretization/2))]
        indicator <- ifelse(welfare1[least_unfair] -  welfare0[least_
           unfair] > 0, 1, 0)
        warm_start = c(xi1[least_unfair,], xi0[least_unfair,], beta[
           least_unfair,], indicator, 1 - indicator, probs[least_unfair
           ,],
        policy1[least_unfair,], policy0[least_unfair,], least_unfair)
}

if(distance == 'relative_welfare'){
        welfare1 <- apply(policy1, 1, function(x) sum(g_i_S*x))
        welfare0 <- apply(policy0, 1, function(x) sum(g_i_S2*x))
        objective_warm_starts_welfare <- welfare0 - welfare1
        if(two_directions) objective_warm_starts_welfare <- abs(welfare
           1 - welfare0)
        least_unfair = which(objective_warm_starts_welfare == min(
           objective_warm_starts_welfare))
        least_unfair = least_unfair[which.min(abs(least_unfair -
           discretization/2))]
        indicator <- ifelse(welfare1[least_unfair] -  welfare0[least_
           unfair] > 0, 1, 0)
        warm_start = c(xi1[least_unfair,], xi0[least_unfair,], beta[
```

```
            least_unfair,], indicator, 1 - indicator, probs[least_unfair
            ,],
        policy1[least_unfair,], policy0[least_unfair,], least_unfair)
    }

    if(distance == 'parity'){
        w1 <- apply(policy1, 1, function(x) sum(S*x/mean(S)))
        w0 <- apply(policy0, 1, function(x) sum((1 - S)*x/(1 - mean(S))
            ))
        objective_warm_starts_welfare <- w0 - w1
        if(two_directions) objective_warm_starts_welfare <- abs(w1 - w0
            )
        least_unfair = which(objective_warm_starts_welfare == min(
            objective_warm_starts_welfare))
        least_unfair = least_unfair[which.min(abs(least_unfair -
            discretization/2))]
        indicator <- ifelse(w1[least_unfair] -  w0[least_unfair] > 0, 1
            , 0)
        warm_start = c(xi1[least_unfair,], xi0[least_unfair,], beta[
            least_unfair,], indicator, 1 - indicator, probs[least_unfair
            ,],
        policy1[least_unfair,], policy0[least_unfair,], least_unfair)

    }

    ## Compute the distance for the envy-based fairness

    if(distance == 'envy'){
        welfare1 <- apply(policy1, 1, function(x) sum(g_i_S*x))
        welfare0 <- apply(policy0, 1, function(x) sum(g_i_S2*x))
        objective_warm_starts1 <- apply(policy1, 1, function(x) sum(mu_
            hat01*x*S)/propensity2 +  sum(mu_hat00*(1 - x)*S)/propensity
            2) -  welfare0
        objective_warm_starts2 <- apply(policy0, 1, function(x) sum(mu_
            hat11*x*(1 - S))/(1 - propensity2) +  sum(mu_hat10*(1 - x)*(
            1 - S))/(1 - propensity2)) - welfare1
        objective_warm_starts_envy <- objective_warm_starts1 +
            objective_warm_starts2
        least_unfair = which(objective_warm_starts_envy == min(
            objective_warm_starts_envy))
        least_unfair = least_unfair[which.min(abs(least_unfair -
            discretization/2))]
        warm_start = c(xi1[least_unfair,], xi0[least_unfair,], beta[
```

```
                   least_unfair,], probs[least_unfair,],
               policy1[least_unfair,], policy0[least_unfair,], least_unfair)
        }

    }

    result <- Est_fairnessMaxScore(Y, X, D, S, propensity1 = propensity1, p_s =
        propensity2,  scale_Y,
    discretization, cost_treatment,
    params, frontier_objective = frontier_objective, mu_hat11 = mu_hat11,
    mu_hat01 = mu_hat01, mu_hat00 = mu_hat00, mu_hat10 = mu_hat10,
    all_g_i, max_treated_units = max_treated_units, maxtime = maxtime1,
    warm_start = warm_start, alpha_seq = alpha_seq, noparity_constraint = no_
       parity_constraint,
    additional_fairness_constraint = additional_fairness_constraint,
    unique_values = unique_values,
    distance = distance, m0 = m0, m1 = m1, probabilistic = probabilistic,
    numcores = numcores, two_directions = two_directions, tolerance = tolerance
       _optimization)
```

1.3 Result

The final simulation results are based on 90 simulated samples, with a limitation of treating 30 individuals. The outcomes using Γ ranking and Fair Policy Learning are as follows:

```
        S    total      run_result
67   26 1  6.4546074          1
123  53 0  2.1783413          1
51   21 0  2.1041590          1
21    8 1  2.0367638          1
192  89 0  1.9740650          1
158  71 0  1.9098849          1
78   28 0  1.4439546          0
81   30 0  1.2206355          0
41   16 0  1.1186445          0
13    4 0  0.8926483          1
142  63 0  0.7267271          1
83   31 0  0.7009540          1
169  76 0  0.6770816          1
178  82 0  0.6643738          1
187  86 0  0.6445900          1
198  90 0  0.5938085          1
126  56 1  0.5774249          1
171  78 0  0.5620174          1
```

```
147 65 0  0.5056116          0
165 74 0  0.4414425          1
43  17 0  0.3590679          1
39  15 0  0.2685100          1
161 72 0  0.2623537          1
128 57 0  0.2558366          1
102 40 0  0.2541770          1
93  35 0  0.2132467          1
125 55 0  0.1775280          1
79  29 0 -0.3208828          1
132 59 1 -0.3286518          0
84  32 0 -0.4660383          1
170 77 0 -0.4720545          0
122 52 0 -0.5234971          1
144 64 0 -0.5365482          1
190 88 1 -0.5387106          1
119 49 0 -0.6388459          1
59  23 1 -0.6692207          0
174 80 0 -0.7609004          0
28  11 0 -0.8148685          0
60  24 0 -0.9341979          0
124 54 0 -0.9448499          0
55  22 0 -0.9699466          0
135 60 0 -0.9750350          0
3    1 0 -0.9876089          0
90  34 0 -1.0113886          0
166 75 0 -1.0566913          0
96  37 0 -1.0693433          0
33  14 0 -1.0883606          0
156 69 0 -1.1067955          0
23   9 0 -1.1151077          0
66  25 0 -1.1304355          0
155 68 0 -1.1391529          0
16   6 1 -1.1687825          0
121 51 0 -1.1759404          0
116 47 0 -1.1908054          0
130 58 0 -1.2193716          0
103 41 0 -1.2451331          0
46  19 0 -1.2938570          0
94  36 0 -1.3242485          0
120 50 0 -1.3266961          0
30  12 0 -1.3336529          0
182 84 0 -1.3374894          0
```

```
117 48 0 -1.3551021          0
100 39 0 -1.3991934          0
176 81 0 -1.4160677          0
44   18 0 -1.5299900          0
14    5 0 -1.5442195          0
110 44 0 -1.5736361          0
77   27 0 -1.6306413          0
97   38 0 -1.6546056          0
8     3 1 -1.6676735          1
157 70 0 -1.7491183          0
163 73 0 -1.7821480          0
179 83 1 -1.9011760          0
153 67 0 -1.9211134          0
113 46 0 -1.9324580          0
141 62 0 -2.0765180          0
109 43 0 -2.2109398          0
185 85 0 -2.2684866          0
188 87 0 -2.2689825          0
111 45 1 -2.4720366          0
31   13 1 -2.8303764          0
50   20 1 -4.1512649          0
5     2 1 -4.4951738          0
139 61 1 -4.4986305          0
24   10 1 -4.4999155          0
151 66 1 -4.5082117          0
89   33 1 -4.5268525          0
20    7 1 -4.5850563          0
104 42 1 -4.6755428          0
173 79 1 -7.1880471          0
```

Among the top 30 individuals ranked by Γ sorting, 25 of them are also identified as treatment recipients using the Fair Policy Learning method, resulting in an accuracy rate of 83.33%. This indicates that the Fair Policy Learning approach is effective in capturing the majority of the high-ranking individuals identified by the Γ sorting method. The high accuracy rate suggests that Fair Policy Learning successfully targets and allocates treatments to the most deserving candidates, aligning with the prioritization achieved by the Γ sorting.

This level of agreement between the two methods demonstrates the reliability and consistency of the Fair Policy Learning approach in selecting candidates for treatment. Moreover, it highlights the potential for Fair Policy Learning to provide equitable and fair outcomes in the context of treatment allocation or resource distribution.

Overall, these simulation results support the viability of Fair Policy Learning as a promising approach for equitable decision-making in various domains, particularly in cases where ranking individuals based on certain criteria is critical for resource allocation or intervention distribution.

2 Another Simulation

```
#生成函数
N = 111
estimand = "ATE"
gamma <- c(0.1, 0.2, 0.3, 0.4, 0.5)
beta <- c(-0.2, -0.3, 0.6, 0.7, -0.6, 0.1)

expit <- function(x) {
    return(1/(1 + exp(-x)))
}
set.seed(123)

X1 <- rbinom(n=N,size=1,prob=1/5)
X2 <- runif(N, min = -1, max = 1)
X3 <- runif(N, min = -1, max = 1)
X4 <- rbinom(n=N,size=1,prob=1/2)
treat <- rbinom(N, 1, expit(gamma[1]+gamma[2]*X1+gamma[3]*X2+gamma[4]*X3+
   gamma[5]*X4))
y0 <- rbinom(N, 1, expit(beta[1]+beta[2]*0+beta[3]*X1+beta[4]*X2+beta[5]*X3
   +beta[6]*X4))
y1 <- rbinom(N, 1, expit(beta[1]+beta[2]*1+beta[3]*X1+beta[4]*X2+beta[5]*X3
   +beta[6]*X4))
y <- y1*treat + y0*(1-treat)
intercept = rep(1, N)
data <- data.frame(intercept, y, y1, y0, treat, X1, X2, X3, X4)
```

There is a tricky aspect here. We have limited the treatment slots to 40 individuals, but in the final calculation, we are only considering whether the top 25 individuals are included in the treatment group.

```
 S      total partity
142  95 0  1.8017467      1
158 106 0  1.6366122      1
51   31 0  1.5423089      1
123  82 0  1.5096470      1
18    9 0  1.3834970      1
54   33 0  1.3824140      1
101  64 0  1.3067750      1
39   26 0  1.2995976      1
150 100 0  1.2971860      1
161 109 0  1.2144727      1
```

```
41     27 0  1.2056227       1
92     59 0  1.1996909       1
133    89 0  1.1830182       1
37     24 0  1.1796614       1
112    72 0  1.1322265       1
119    78 0  1.1102610       1
81     52 0  1.1068963       1
64     40 0  0.9817271       1
76     48 0  0.9576977       1
83     53 0  0.9423477       1
147    98 0  0.8687710       1
15      7 0  0.8645039       1
125    84 0  0.5865378       1
93     60 0  0.5530478       1
72     44 0 -0.2293424       0
46     29 0 -0.3857517       0
127    85 0 -0.3973064       0
61     37 0 -0.4241628       0
96     61 0 -0.5395551       0
22     12 0 -0.5883734       0
23     13 0 -0.6117807       0
69     43 0 -0.7057877       1
122    81 0 -0.7578728       1
155 104 0 -0.7615240       0
128    86 0 -0.7795715       1
144    97 0 -0.7923502       1
77     49 0 -0.8475241       0
28     17 0 -0.8501366       1
55     34 0 -0.8642962       1
38     25 0 -0.8915140       0
62     38 0 -0.9733592       0
154 103 0 -1.0564223       0
143    96 0 -1.0592918       0
162 110 0 -1.0695226       0
136    91 0 -1.0933734       0
148    99 0 -1.1137329       0
108    69 0 -1.1140726       0
120    79 0 -1.1144669       0
26     16 0 -1.1207854       0
3       2 0 -1.1269888       0
58     35 0 -1.1352057       0
159 107 0 -1.1427198       0
166 111 0 -1.1779966       0
```

```
79    51 0 -1.1852879       0
35    23 0 -1.1933083       0
78    50 0 -1.2009454       0
135   90 0 -1.2200911       0
124   83 0 -1.2451100       1
115   74 0 -1.2464804       0
90    58 0 -1.2465306       0
7      5 0 -1.2465471       0
103   66 0 -1.2468070       0
73    45 0 -1.2468397       0
117   76 0 -1.2471173       0
156  105 0 -1.2472696       0
75    47 0 -1.2476119       0
97    62 0 -1.2476403       0
30    19 0 -1.2479588       0
160  108 0 -1.2664938       0
116   75 0 -1.2862239       0
44    28 0 -1.2862747       0
2      1 0 -1.2908750       1
130   87 0 -1.2916142       0
29    18 0 -1.3118718       0
33    22 0 -1.3130693       0
121   80 0 -1.3168953       0
63    39 0 -1.3347038       0
98    63 0 -1.3359454       1
110   71 0 -1.3452301       0
153  102 0 -1.3835483       0
102   65 0 -1.3939040       0
74    46 0 -1.4256337       0
109   70 0 -1.4714498       0
25    15 0 -1.6979296       0
86    54 0 -1.7543307       1
6      4 0 -1.9753303       0
52    32 0 -2.1219163       1
14     6 0 -2.1754460       0
141   94 0 -2.6715217       0
20    10 1 -5.0436516       0
118   77 1 -5.0447138       0
31    20 1 -5.0448630       0
132   88 1 -5.0448849       0
24    14 1 -5.0449122       0
88    56 1 -5.0451504       0
114   73 1 -5.0451968       0
```

```
139   93 1 -5.0452038        0
50    30 1 -5.0452827        0
4      3 1 -5.0455343        0
87    55 1 -5.0455445        0
67    42 1 -5.0455694        0
137   92 1 -5.0456088        0
65    41 1 -5.0456515        0
59    36 1 -5.0456683        0
32    21 1 -5.0457240        0
16     8 1 -5.0457879        0
107   68 1 -5.0458418        0
89    57 1 -5.0458556        0
104   67 1 -5.0458733        0
151  101 1 -5.0459120        0
21    11 1 -5.0459387        0
```

Among the top 25 individuals ranked by Γ, 24 of them are assigned to the treatment group using Fair Policy Learning.