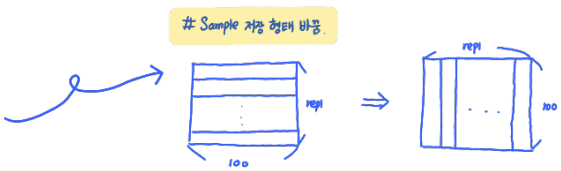
<4월 15일 coding 진행 부분>

```
library(boot)
library(tableone)
library(survey)
library(Hmisc)
library(tidyverse)
library(Matching)

delta<-seq(-2,-1,by=0.01) # delta0 sequence
prevalence<-matrix(0,nrow=length(delta),ncol=1)

## prevalence calculate by changing delta0
for(i in 1:length(delta)){
  delta0<-delta[i]
  delta_b<-0.01
  delta_c<-0.01
  #B,C,U,E generating
  set.seed(123*i)
  exposure_hat<-replicate(100000,expr={
    #set.seed(delta0)
    B<-rbinom(1,1,prob=0.5)
    C<-rnorm(1,0,1)
    U<-runif(1)
    p_z<-inv.logit(delta0+delta_b*B+delta_c*C)
    E<-rbinom(1,1,prob=p_z)
    c(B,C,U,E)
    })
  prevalence[i,]<-mean(exposure_hat[4,])
}
# exposure ratio
prevalence
min(prevalence)
[1] 0.11981

## Data generating
delta_rare<-delta[which(prevalence==min(prevalence))] # -2

delta_rare
[1] -2

# sample store
B_sample<-matrix(0,nrow=100,ncol=1000)
```

```
C_sample<-matrix(0,nrow=100,ncol=1000)
U_sample<-matrix(0,nrow=100,ncol=1000)
E_sample<-matrix(0,nrow=100,ncol=1000)
Y_sample<-matrix(0,nrow=100,ncol=1000)

effect<-c(log(1.2),log(1.5),log(2),log(2))

# 1000th replication data generating
for(repl in 1:1000){
  delta0<-delta_rare
  delta_b<-0.01
  delta_c<-0.01
  set.seed(123*repl)
  sample<-replicate(100,expr={
    B<-rbinom(1,1,prob=0.5)
    C<-rnorm(1,0,1)
    U<-runif(1)
    p_z<-inv.logit(delta0+delta_b*B+delta_c*C)
    E<-rbinom(1,1,prob=p_z)
    c(B,C,U,E)
    })
  B_sample[,repl]<-sample[1,]
  C_sample[,repl]<-sample[2,]
  U_sample[,repl]<-sample[3,]
  E_sample[,repl]<-sample[4,]
  X_sample<-cbind(B_sample[,repl],C_sample[,repl],U_sample[,repl],E_sample[,repl])

  # Y random sampling - continuous
  Y_sample[,repl]<-as.matrix(X_sample)%*%effect+rnorm(nrow(X_sample),0,1)
}
```




```
## Data store ##
write.csv(B_sample,file="B_sample.csv",row.names=FALSE)
write.csv(C_sample,file="C_sample.csv",row.names=FALSE)
#write.csv(U_sample,file="U_sample.csv",row.names=FALSE)
write.csv(E_sample,file="E_sample.csv",row.names=FALSE)
write.csv(Y_sample,file="Y_sample.csv",row.names=FALSE)
```

⇒ 생성한 Data csv file로 저장

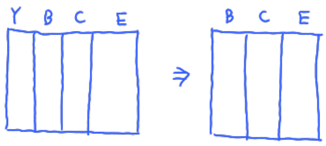
```
## Data import ##
```

; 저장한 Data Import

```
B_sample<-read.csv("B_sample.csv",header=TRUE)
```

```r
C_sample<-read.csv("C_sample.csv",header=TRUE)
#U_sample<-read.csv("U_sample.csv",header=TRUE)
E_sample<-read.csv("E_sample.csv",header=TRUE)
Y_sample<-read.csv("Y_sample.csv",header=TRUE)

## Data export function - treat + covariance ##
#data_export<-function(var_treat,var_cov,data){
#   name<-c(var_treat,var_cov)
#   idx_name<-rep(0,length(name))
#
#   for(n in 1:length(name)){
#     idx_name[n]<-which(colnames(data)==name[n])
#   }
#   mydata<-data[idx_name]
#   return(mydata)
#}
```

문자열로 받음




```r
########################################################################
#############
## weight generation function ##
weight_make<-function(var_treat,var_cov,estimate,data){
  result<-list()
  mydata<-data[c(var_treat,var_cov)]

  trt<-mydata[,var_treat]
  ps<-glm(trt~.,data=mydata,family='binomial')$fitted.values
  #ps<-predict(psmodel,type='response')

  if(estimate=="ATE"){
    weight<-ifelse(trt==1,1/ps,1/(1-ps))
  }
  else if(estimate=='ATT'){
    weight<-ifelse(trt==1,1,ps/(1-ps))
  }
  else{
    weight<-ifelse(trt==1,(1-ps)/ps,1)
  }
  result[[1]]<-weight

result[[2]]<-ifelse(weight>quantile(weight,prob=0.99),quantile(weight,prob=0.99),weight)
  return(result)}
```

Raw Data

String 형태 (treat 변수, confounder 변수명)

원하는 추정치

만 가지고 오도록,,

; 성향점수 get

이전에 잘못 작성한 부분 수정.

truncation 적용한 weight 와 하지 않은 weight 2가지 version list에 저장

```
## weighted data export ##
weighteddata<-function(data,weight){
  require(survey)
  data<-svydesign(ids=~1,data=data,weights=~weight)
  mydata<-data[[7]]
  return(mydata)
}

## balance check function - no function ##
#balance_check<-function(var_treat,var_cov,weight,data){
#  before_table<-matrix(0,nrow=length(var_cov),ncol=2)
#  rownames(before_table)<-var_cov
#
colnames(before_table)<-c("IPTW_before_mean_difference","IPTW_before_variance_rati
o")
#
#  after_table<-matrix(0,nrow=length(var_cov),ncol=2)
#  rownames(after_table)<-var_cov
#
colnames(after_table)<-c("IPTW_after_mean_difference","IPTW_after_variance_ratio")
#
#  ind_treat<-which(colnames(data)==var_treat)
#  treat<-data[,ind_treat]
#  cov<-data[-ind_treat]
#  for(i in 1:length(cov)){
#    var_cov<-cov[,i]
#    std_var_cov = (var_cov - mean(var_cov))/sd(var_cov)
#    simple_M1 = mean(std_var_cov[treat==1])
#    simple_M0 = mean(std_var_cov[treat==0])
#    simple_V1 = var(std_var_cov[treat==1])
#    simple_V0 = var(std_var_cov[treat==0])
#    wgted_M1 = Hmisc::wtd.mean(x=std_var_cov[treat==1],weights=weight[treat==1])
#    wgted_M0 = Hmisc::wtd.mean(x=std_var_cov[treat==0],weights=weight[treat==0])
#    wgted_V1 = Hmisc::wtd.var(x=std_var_cov[treat==1],weights=weight[treat==1])
#    wgted_V0 = Hmisc::wtd.var(x=std_var_cov[treat==0],weights=weight[treat==0])
#    before_table[i,1] = simple_M1 - simple_M0
#    before_table[i,2] = simple_V1/simple_V0
#    after_table[i,1] = wgted_M1 - wgted_M0
#    after_table[i,2] = wgted_V1/wgted_V0
#  }
```

```
#   result<-cbind(before_table,after_table)
#   return(result)
#}

## balance check function - function yes ##
balance_check<-function(var_treat,var_cov,data){
```

(var_treat, var_cov → String 형태; data → raw Data)

```
  require(tableone)
  table<-CreateTableOne(vars=var_cov,strata=var_treat,data=data,test=FALSE)
  print(table,smd=TRUE)
}
```

(CreateTableOne → Balance Check 할 수 있는 table 생성; smd=TRUE → SMD도 같이 제시)

```
## check ##
```

; 함수 결과 확인하는 Example

```
E<-E_sample[,1]
B<-B_sample[,1]
#U<-U_sample[,1]
C<-C_sample[,1]
Y<-Y_sample[,1]
data<-data.frame("E"=E,"B"=B,"C"=C,"Y"=Y)
cov<-c("B","C")
#mydata<-data_export("E",cov,data)
#head(mydata)
tableone_before<-balance_check(var_treat="E",var_cov=cov,data=data)
```

(Weight 부여 전 confounder 간 Balance check)

```
                     Stratified by E
                      0              1             SMD
  n                  88             12
  B (mean (SD))   0.56 (0.50) 0.33 (0.49)   0.451
  C (mean (SD)) -0.04 (0.99) 0.31 (1.10)    0.335

weight_result<-weight_make("E",cov,estimate='ATE',data=data)

weight_result
[[1]]
```

; Question) Weight가 모두 1이라는 것은 PS-score가 매우 작다는 것을 의미 ⇒ how to Solve ?

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [63] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[[2]]
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [63] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
# weighted data #
weighteddata1<-weighteddata(data=data,weight=weight_result[[1]]) #weighteddata[[7]]
weighteddata2<-weighteddata(data=data,weight=weight_result[[2]])
tableone_after1<-balance_check(var_treat="E",var_cov=cov,data=weighteddata1)
                  Stratified by E
                   0              1              SMD
  n                   88             12
  B (mean (SD))   0.56 (0.50) 0.33 (0.49)   0.451
  C (mean (SD)) -0.04 (0.99) 0.31 (1.10)   0.335

tableone_after2<-balance_check(var_treat="E",var_cov=cov,data=weighteddata2)
                  Stratified by E
                   0              1              SMD
  n                   88             12
  B (mean (SD))   0.56 (0.50) 0.33 (0.49)   0.451
  C (mean (SD)) -0.04 (0.99) 0.31 (1.10)   0.335
```

추정한 값 저장 위해 matrix 생성

```
#### IPTW ATE & ATT ####
result_ATE<-matrix(0,nrow=1000,ncol=2)
colnames(result_ATE)<-c("before_truncation_ATE","after_truncation_ATE")
result_ATE<-as.data.frame(result_ATE)

result_ATT<-matrix(0,nrow=1000,ncol=2)
colnames(result_ATT)<-c("before_truncation_ATT","after_truncation_ATT")
result_ATT<-as.data.frame(result_ATT)
```

```
## 1000th simulated ##
for(j in 1:1000){
  E<-E_sample[,j]
  B<-B_sample[,j]
  C<-C_sample[,j]
  #U<-U_sample[,j]
  Y<-Y_sample[,j]
```

각 replication 마다 Data 가져오기

DataFrame 생성

```
  data<-data.frame("E"=E,"B"=B,"C"=C,"Y"=Y)
  cov<-c("B","C")
```

Confounder 변수명 지정

```
  #mydata<-data_export("E",cov,data)
  weight_ATE<-weight_make("E",cov,estimate='ATE',data=data)
  weight_ATT<-weight_make("E",cov,estimate='ATT',data=data)
```

Weight 생성

```
  weighteddata1_ATE<-weighteddata(data=data,weight=weight_ATE[[1]])
```

truncation 안한 weight 이용해 pseudo-population 생성

```
  weighteddata2_ATE<-weighteddata(data=data,weight=weight_ATE[[2]])
  weighteddata1_ATT<-weighteddata(data=data,weight=weight_ATT[[1]])
  weighteddata2_ATT<-weighteddata(data=data,weight=weight_ATT[[2]])

  result_ATE[j,1]<-lm(Y~E,data=weighteddata1_ATE)$coef['E']
  result_ATE[j,2]<-lm(Y~E,data=weighteddata2_ATE)$coef['E']

  result_ATT[j,1]<-lm(Y~E,data=weighteddata1_ATT)$coef['E']
  result_ATT[j,2]<-lm(Y~E,data=weighteddata1_ATT)$coef['E']

}

mean(result_ATE[,1])
[1] 0.6957256
mean(result_ATE[,2])
[1] 0.6957256

mean(result_ATT[,1])
[1] 0.6957256

mean(result_ATT[,2])
[1] 0.6957256
log(2)
[1] 0.6931472

# plotting - ATE #
par(mfrow=c(2,1))
plot(result_ATE[,1],main="ATE estimators using IPTW")
abline(h=log(2),col='red')
plot(result_ATE[,2],main="ATE estimators using IPTW_truncated version")
abline(h=log(2),col='red')

# plotting - ATT #
par(mfrow=c(2,1))
plot(result_ATT[,1],main="ATT estimators using IPTW")
abline(h=log(2),col='red')
plot(result_ATT[,2],main="ATT estimators using IPTW_truncated version")
abline(h=log(2),col='red')

##########################################################################
###########
```

truncation 한 weight 이용해 pseudo-population 생성

생성한 pseudo-population 이용해 ATE, ATT 추정

```
## matched data export ##
matcheddata<-function(var_treat,var_cov,data,PSM,M){
  require(Matching)
  if(PSM==TRUE){
    mydata<-data[c(var_treat,var_cov)]
    trt<-mydata[,var_treat]
    ps<-glm(trt~.,data=mydata,family='binomial')$fitted.values
    m.out<-Match(Tr=data[,var_treat],M=M,X=logit(ps),replace=FALSE)
  }
  else{
    m.out<-Match(Tr=data[,var_treat],M=M,X=data[,var_cov],replace=FALSE)
  }
  matched<-data[unlist(m.out[c("index.treated","index.control")]),]
  return(matched)
}

> ## check ##
E<-E_sample[,1]
B<-B_sample[,1]
#U<-U_sample[,1]
C<-C_sample[,1]
Y<-Y_sample[,1]
data<-data.frame("E"=E,"B"=B,"C"=C,"Y"=Y)
cov<-c("B","C")

# matched data #
matchdata_greedy<-matcheddata("E",cov,data=data,PSM=FALSE,M=1)
matchdata_PSM<-matcheddata("E",cov,data=data,PSM=TRUE,M=1)

# matched data balance check #
balance_PSM<-balance_check("E",cov,matchdata_PSM)
                  Stratified by E
                   0              1              SMD
  n                   12             12
  B (mean (SD))   0.50 (0.52) 0.33 (0.49)   0.328
  C (mean (SD)) -0.48 (0.80) 0.31 (1.10)   0.825

balance_greedy<-balance_check("E",cov,matchdata_greedy)
                  Stratified by E
                   0              1              SMD
  n                   12             12
```

; Matching 후 matching 된 Data return 해주는 function

String 형태

PS Matching 시행여부

한 개의 treated subject에 대해 몇 개의 control subject matching 할 것인지 지정

; 함수 확인 위한 Example

모두 1:1로 지정

```
   B (mean (SD)) 0.33 (0.49) 0.33 (0.49) <0.001
   C (mean (SD)) 0.21 (0.86) 0.31 (1.10) 0.105
```

↳ greedy matching 통해 group간 balance 정도 맞추어진것 확인!

```
#### PSM matching & greedy matching ####
PSM_ATE<-c()
greedy_ATE<-c()
```

↳ 결과 저장 위한 변수 생성

```
## 1000th simulated ##
for(j in 1:1000){
   E<-E_sample[,j]
   B<-B_sample[,j]
   C<-C_sample[,j]
   #U<-U_sample[,j]
   Y<-Y_sample[,j]
   data<-data.frame("E"=E,"B"=B,"C"=C,"Y"=Y)
   cov<-c("B","C")

  # matched data generation #
   matchdata_greedy<-matcheddata("E",cov,data=data,PSM=FALSE,M=1)
   matchdata_PSM<-matcheddata("E",cov,data=data,PSM=TRUE,M=1)


 PSM_ATE[j]<-matchdata_PSM$Y[matchdata_PSM$E==1]-matchdata_PSM$Y[matchdata
 _PSM$E==0]
```

; E[Y|E=1] - E[Y|E=0] 계산

```
 greedy_ATE[j]<-matchdata_greedy$Y[matchdata_greedy$E==1]-matchdata_greedy$Y[
 matchdata_greedy$E==0]

 }

 plot(PSM_ATE,main="ATE estimators using PSM")
 abline(h=log(2),col='red')
 plot(PSM_ATE,main="ATE estimators using greedy matching")
 abline(h=log(2),col='red')

 ###################################################################
 #### Regression ####
 ATE_regression<-c()

 for(j in 1:1000){
   E<-E_sample[,j]
```

```
  B<-B_sample[,j]
  C<-C_sample[,j]
  #U<-U_sample[,j]
  Y<-Y_sample[,j]
  data<-data.frame("E"=E,"B"=B,"C"=C,"Y"=Y)
  cov<-c("B","C")

  #mydata<-data_export("E",cov,data)
  ATE_regression[j]<-lm(Y~.,data=data)$coef['E']
}

par(mfrow=c(1,1))
plot(ATE_regression,main="ATE estimators using regression")
abline(h=log(2),col='red')
```

; Regression 결과 회귀계수 저장.