

CAUSAL ADJUSTMENT METHODS USING THE PROPENSITY SCORE

Several different methods based on the propensity score can be used for causal adjustment:

- **Stratification** : The propensity score space is partitioned into J regions or *strata*, and the 'local' causal effect is estimated in each, and then combined in an average to estimate $\mu(z)$
- **Matching** : The outcomes for individuals with different Z values but equal $e(X)$ values (that is, the individuals are *matched* on their propensity score values) are compared.
- **Regression** : The propensity score is conditioned upon using a *regression model*.
- **Inverse Weighting** : Using a *reweighting* computed via the propensity score of the Y values observed under $\circ$, a pseudo-sample is created that can be treated as a sample collected under $\mathcal{E}$ in which exposure is balanced.

Simulation study: Consider the following example where we generate according to the parametric model

- $X \sim \text{Normal}_3(\mu, \Sigma)$ with

$$\mu = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 0.25 & 0.00 & 0.00 \\ 0.00 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.75 \end{bmatrix} \begin{bmatrix} 1.0 & 0.9 & -0.1 \\ 0.9 & 1.0 & -0.2 \\ -0.1 & -0.2 & 1.0 \end{bmatrix} \begin{bmatrix} 0.25 & 0.00 & 0.00 \\ 0.00 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.75 \end{bmatrix}$$

- $Z|X = x \sim \text{Bernoulli}(e(x; \alpha))$, with

$$e(x; \alpha) = \frac{\exp\{\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1 x_2\}}{1 + \exp\{\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1 x_2\}}$$

with $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)^\top = (6.0, -0.2, 0.7, 2)^\top$.

- $Y|X = x, Z = z \sim \text{Normal}(\mu(x, z; \beta, \psi), \sigma^2)$, with

$$\mu(x, z) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + z(\psi_0 + \psi_1 x_1 + \psi_2 x_2 + \psi_3 x_1 x_2) = \mu_0(x; \beta) + z\mu_1(x; \psi)$$

say, with $\sigma^2 = 0.1^2$, and

$$\beta = (\beta_0, \beta_1, \beta_2, \beta_3)^\top = (10, -2.0, 1.2, 0.6)^\top \quad \psi = (\psi_0, \psi_1, \psi_2, \psi_3)^\top = (1, 1, 1, 1)^\top$$

In this model, X_1 and X_2 are confounders, and X_3 is a spurious variable unrelated to Y or Z .

In the following simulation, we assume that $n = 2000$ to make the empirical results clearer.

```
#Data simulation
library(MASS)
library(xtable)
set.seed(231987)
n<-2000
D<-diag(c(0.25,0.5,0.75))
Mu<-c(1,-2,-1)
Sigma<-D %*% (matrix(c(1,0.9,-0.1,0.9,1,-0.2,-0.1,-0.2,1),3,3) %*% D)
X<-mvrnorm(n,mu=Mu,Sigma)
al<-c(6.0,-0.2,0.7,2)
Xal<-cbind(1,X[,1],X[,2],X[,1]*X[,2])
expit<-function(x){return(1/(1+exp(-x)))}
ps.true<-expit(Xal %*% al)
Z<-rbinom(n,1,ps.true)
be<-c(10,-2.0,1.2,0.6)
Xb<-cbind(1,X[,1],X[,2],X[,1]*X[,2])
mu0<- Xb %*% be
psi<-c(1,1,1,1)
Xp<-cbind(1,X[,1],X[,2],X[,1]*X[,2])
mu1<- (Xp %*% psi)
eta<-mu0 + Z * mu1 ; sig<-0.1
Y<-rnorm(n,eta,sig)
X1<-X[,1];X2<-X[,2];X3<-X[,3]
true.vals<-c(be,psi)[c(1,2,3,5,4,6,7,8)]
eX<-as.vector(Z-ps.true)
```

The average treatment effect (ATE) can be computed directly in terms of the model parameters: if we denote by

- $\mathcal{E}$ the target experimental model
- $\mathcal{O}$ the *observational* model

then the ATE is computed assuming that $\mathcal{E}$ is true probability model, whereas in fact $\mathcal{O}$ is the true probability model. We have

$$\mu(z) = \mathbb{E}_{Y|Z}^{\mathcal{E}}[Y|Z = z] \equiv \mathbb{E}_X^{\mathcal{O}} \left[\mathbb{E}_{Y|X,Z}^{\mathcal{O}}[Y|X, Z = z] \right] = \mathbb{E}_X^{\mathcal{O}} [\mu(X, z)]$$

where this expectation is simply an expectation over the X distribution, which is presumed the same in both experimental and observational models. Therefore, in this case

$$\begin{aligned} \mu(z) &= \mathbb{E}_X^{\mathcal{O}} [\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + z(\psi_0 + \psi_1 X_1 + \psi_2 X_2 + \psi_3 X_1 X_2)] \\ &= \beta_0 + \beta_1 \mathbb{E}[X_1] + \beta_2 \mathbb{E}[X_2] + \beta_3 \mathbb{E}[X_1 X_2] + z(\psi_0 + \psi_1 \mathbb{E}[X_1] + \psi_2 \mathbb{E}[X_2] + \psi_3 \mathbb{E}[X_1 X_2]) \end{aligned}$$

Here we have that

$$\mathbb{E}[XX^{\top}] = \begin{bmatrix} \mathbb{E}[X_1^2] & \mathbb{E}[X_1 X_2] & \mathbb{E}[X_1 X_3] \\ \mathbb{E}[X_1 X_2] & \mathbb{E}[X_2^2] & \mathbb{E}[X_2 X_3] \\ \mathbb{E}[X_1 X_3] & \mathbb{E}[X_2 X_3] & \mathbb{E}[X_3^2] \end{bmatrix} = \Sigma + \mu\mu^{\top} = \begin{bmatrix} 0.0625 & 0.1125 & -0.01875 \\ 0.1125 & 0.25 & -0.075 \\ -0.01875 & -0.075 & 0.5625 \end{bmatrix} + \begin{bmatrix} 1 & -2 & -1 \\ -2 & 4 & 2 \\ -1 & 2 & 1 \end{bmatrix}$$

which yields

```
(X.M<-Sigma + Mu %*% t(Mu))

##           [,1]      [,2]      [,3]
## [1,]  1.06250 -1.8875 -1.01875
## [2,] -1.88750  4.2500  1.92500
## [3,] -1.01875  1.9250  1.56250
```

so therefore $\mu(0)$ and $\mu(1)$ are

```
(mu0<-sum(be*c(1,Mu[1],Mu[2],X.M[1,2])))

## [1] 4.4675

(mu1<-mu0 + sum(psi*c(1,Mu[1],Mu[2],X.M[1,2])))

## [1] 2.58
```

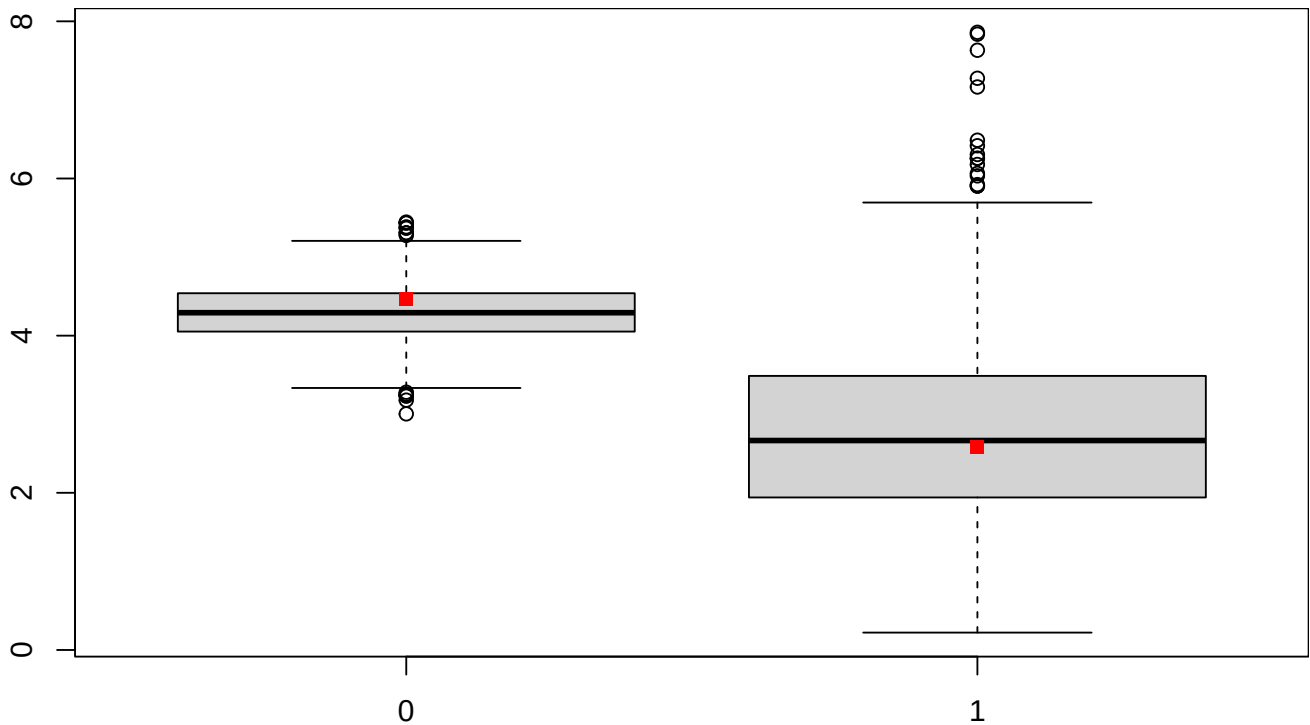
and hence the ‘causal effect’ is

$$\mu(1) - \mu(0) = 2.58 - 4.4675 = -1.8875.$$

1. Unadjusted analysis

If we try to analyze the data as if they arose from an experimental study, as in an unadjusted analysis, it is evident that incorrect inferences are drawn.

```
par(mar=c(2,2,0,0))
boxplot(Y~Z)
points(1:2,c(mu0,mu1),col='red',pch=15)
```



```
c(mean(Y[Z==0]),mean(Y[Z==1]),mean(Y[Z==1])-mean(Y[Z==0]))
+ [1] 4.290417 2.772258 -1.518159
```

Thus the estimated causal effect is -1.518157. We can obtain the same result using regression

```
fit0<-lm(Y~Z)
summary(fit0)$coef

+           Estimate Std. Error  t value    Pr(>|t|)
+ (Intercept)  4.290417 0.03530282 121.53182 0.000000e+00
+ Z           -1.518159 0.04412853 -34.40311 4.182492e-204

beta.hat<-coef(fit0)
```

but note that the standard error reported here is the usual ‘model-based’ standard error, based on the variance estimate

$$\hat{\mathbf{V}} = \hat{\sigma}^2(\mathbf{X}^\top \mathbf{X})^{-1}$$

where $\mathbf{X}$ is the design matrix for the fitted model. Another option is to report the *robust* or *sandwich* standard error which is computed using the formula

$$\hat{\mathbf{V}} = \mathbf{A}^{-1} \mathbf{B} \mathbf{A}^{-\top}$$

where

$$\mathbf{A} = \mathbf{X}^\top \mathbf{X} \quad \mathbf{B} = \mathbf{X}^\top (\mathbf{y} - \mathbf{X} \hat{\boldsymbol{\beta}})$$

```
library(sandwich)
robust.var<-sandwich(fit0)
robust.var

+           (Intercept)           Z
+ (Intercept) 0.000215369 -0.000215369
+ Z          -0.000215369  0.001241501
```

```

Xm<-cbind(1,Z)
A<-t(Xm) %*% Xm
B<-t(Xm) %*% (diag(as.vector(Y-Xm %*% beta.hat)^2) %*% Xm)
solve(A) %*% B %*% t(solve(A))

+           Z
+ 0.000215369 -0.000215369
+ Z -0.000215369 0.001241501

sqrt(diag(robust.var))

+ (Intercept)           Z
+ 0.01467546 0.03523494

```

Thus the robust standard error for the coefficient of Z is 0.0352349, and an approximate 95% confidence interval for the ATE is

$$-1.5181587 \pm 2 \times 0.0352349 = (-1.5886286, -1.4476888)$$

which excludes the true value. This estimator is biased.

2. Outcome Regression

2.1 Correct Specification

A regression analysis using a correctly specified model returns correct parameter estimation.

```

fit0<-lm(Y~X1+X2+X1:X2+Z+Z:X1+Z:X2+Z:X1:X2)
round(cbind(coef(summary(fit0)),true.vals),6)

+           Estimate Std. Error   t value Pr(>|t|) true.vals
+ (Intercept) 10.012666  0.091711 109.176735     0      10.0
+ X1          -1.999370  0.070488 -28.364886     0      -2.0
+ X2           1.196749  0.033121  36.132629     0       1.2
+ Z            0.953441  0.108799   8.763313     0       1.0
+ X1:X2        0.612234  0.029825  20.527421     0       0.6
+ X1:Z         1.006817  0.080260  12.544493     0       1.0
+ X2:Z         0.996127  0.038449  25.907849     0       1.0
+ X1:X2:Z      0.980896  0.033325  29.434509     0       1.0

```

From this fit, it is possible to estimate the causal quantities using an average prediction

$$\hat{\mu}(z) = \frac{1}{n} \sum_{i=1}^n \mu(x_i, z; \hat{\beta}, \hat{\psi}) \quad (1)$$

```

mu0.0<-mean(predict(fit0,newdata=data.frame(X1,X2,Z=0)))
mu1.0<-mean(predict(fit0,newdata=data.frame(X1,X2,Z=1)))
c(mu0.0,mu1.0,mu1.0-mu0.0)

+ [1] 4.456167 2.558785 -1.897382

```

Under correct specification, the sandwich standard errors for the coefficients should be approximately equal to the standard errors computed by `lm`

```

sqrt(diag(sandwich(fit0)))

+ (Intercept)      X1      X2      Z      X1:X2      X1:Z
+ 0.09194223 0.07036956 0.03270877 0.10935666 0.02862122 0.08044854
+      X2:Z      X1:X2:Z
+ 0.03813006 0.03235183

```

However, in this case, to report the standard error for the ATE, we need to compute using the appropriate linear combination formula. We have that

$$\hat{\mu}(1) - \hat{\mu}(0) = \frac{1}{n} \mathbf{1}^\top (\mathbf{X}_1 \hat{\beta}) - \frac{1}{n} \mathbf{1}^\top (\mathbf{X}_0 \hat{\beta}) = \frac{1}{n} \mathbf{1}^\top (\mathbf{X}_1 - \mathbf{X}_0) \hat{\beta} = \mathbf{D} \hat{\beta}$$

say, where $\mathbf{X}_0$ and $\mathbf{X}_1$ are the design matrices from the linear model with Z set to zero and one respectively. The estimated variance of the ATE estimator is then

$$\mathbf{D} \hat{\mathbf{V}} \mathbf{D}^\top$$

where $\hat{\mathbf{V}}$ is the estimated model-based or sandwich estimator.

```
Z0<-rep(0,n);Z1<-Z0+1
Xm0<-cbind(1,X1,X2,Z0,X1*X2,Z0*X1,Z0*X2,Z0*X1*X2)
Xm1<-cbind(1,X1,X2,Z1,X1*X2,Z1*X1,Z1*X2,Z1*X1*X2)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.0<-as.numeric(Dm %*% coef(fit0))

#Model-based standard error
Vm<-summary(fit0)$sigma^2*summary(fit0)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit0)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.005004624 0.004975802

Xm<-cbind(1,X1,X2,Z,X1*X2,Z*X1,Z*X2,Z*X1*X2)
beta.hat<-coef(fit0)
A<-t(Xm) %*% Xm
B<-t(Xm) %*% (diag(as.vector(Y-Xm %*% beta.hat)^2) %*% Xm)
Vs<-solve(A) %*% B %*% t(solve(A))

se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))
se.s[1,1]

+ [1] 0.004975802
```

Therefore the two ATE standard errors are almost identical. The two 95% confidence intervals are

$$(-1.9073915, -1.887373) \quad (-1.9073338, -1.8874306)$$

Note: In this model, what is being computed is **not the average of the predictions for the observed z values** in the two cases for $z_i = 1$ and $z_i = 0$, which would be respectively

$$\frac{\sum_{i=1}^n z_i \mu(x_i, 1; \hat{\beta}, \hat{\psi})}{\sum_{i=1}^n z_i} \quad \frac{\sum_{i=1}^n (1 - z_i) \mu(x_i, 0; \hat{\beta}, \hat{\psi})}{\sum_{i=1}^n (1 - z_i)}.$$

Such a procedure does not estimate the required quantities correctly:

```
m0<-sum((1-Z)*predict(fit0))/sum(1-Z)
m1<-sum(Z*predict(fit0))/sum(Z)
c(m0,m1,m1-m0)

+ [1] 4.290417 2.772258 -1.518159
```

This is due to the confounding present in the structure of the joint distribution,

2.2 Incorrect specification

An incorrectly specified model leads to incorrect estimation, although the estimated causal parameters can still potentially be recovered. For example, if the interaction terms are omitted, although it is not possible to estimate individual coefficients correctly, the causal quantities of interest are estimated reasonably well.

```
fit1<-lm(Y~X1+X2+Z+Z:X1+Z:X2)
round(coef(summary(fit1)),6)

+           Estimate Std. Error   t value Pr(>|t|)
+ (Intercept) 11.250384   0.182438  61.666973    0
+ X1          -3.265528   0.090069 -36.255862    0
+ X2           1.770349   0.046945  37.711503    0
+ Z            2.883118   0.226003  12.757016    0
+ X1:Z         -0.753598   0.113230  -6.655479    0
+ X2:Z          1.981104   0.058201  34.039270    0

mu0.1<-mean(predict(fit1,newdata=data.frame(X1,X2,Z=0)))
mu1.1<-mean(predict(fit1,newdata=data.frame(X1,X2,Z=1)))
c(mu0.1,mu1.1,mu1.1-mu0.1)

+ [1]  4.434850  2.584919 -1.849930
```

The omission of the $X_1:X_2$ and $Z:X_1:X_2$ interactions does not lead to very poor results in this case because the magnitude of the effect of these terms is relatively small. For the standard errors

```
Xm0<-cbind(1,X1,X2,Z0,Z0*X1,Z0*X2)
Xm1<-cbind(1,X1,X2,Z1,Z1*X1,Z1*X2)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.1<-as.numeric(Dm %*% coef(fit1))

#Model-based standard error
Vm<-summary(fit1)$sigma^2*summary(fit1)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit1)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.01290896 0.01050771

Xm<-cbind(1,X1,X2,Z,Z*X1,Z*X2)
beta.hat<-coef(fit1)
A<-t(Xm) %*% Xm
B<-t(Xm) %*% (diag(as.vector(Y-Xm %*% beta.hat)^2) %*% Xm)
Vs<-solve(A) %*% B %*% t(solve(A))

se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))
se.s[1,1]

+ [1] 0.01050771
```

Here we see some difference between the estimated standard errors: the model-based version standard error is 0.012909 whereas the sandwich standard error is 0.0105077. The corresponding confidence intervals are

$$(-1.8757483, -1.8241124) \quad (-1.8709458, -1.8289149)$$

which both exclude the true value.

If either X_1 or X_2 or both is omitted, there is no correct recovery.

```

fit2<-lm(Y~X1+Z+Z:X1)
mu0.2<-mean(predict(fit2,newdata=data.frame(X1,Z=0)))
mu1.2<-mean(predict(fit2,newdata=data.frame(X1,Z=1)))
c(mu0.2,mu1.2,mu1.2-mu0.2)

+ [1] 4.287774 2.757379 -1.530396

fit3<-lm(Y~X2+Z+Z:X2)
mu0.3<-mean(predict(fit3,newdata=data.frame(X1,Z=0)))
mu1.3<-mean(predict(fit3,newdata=data.frame(X1,Z=1)))
c(mu0.3,mu1.3,mu1.3-mu0.3)

+ [1] 4.317704 2.666532 -1.651171

fit4<-lm(Y~Z)
mu0.4<-mean(predict(fit4,newdata=data.frame(X1,Z=0)))
mu1.4<-mean(predict(fit4,newdata=data.frame(X1,Z=1)))
c(mu0.4,mu1.4,mu1.4-mu0.4)

+ [1] 4.290417 2.772258 -1.518159

```

3. Propensity score stratification

For propensity score stratification, we seek to first construct strata of X , $\mathcal{X}_j, j = 1, \dots, J$ defined by propensity score values, then estimate the causal quantities within each stratum, and then combine across strata.

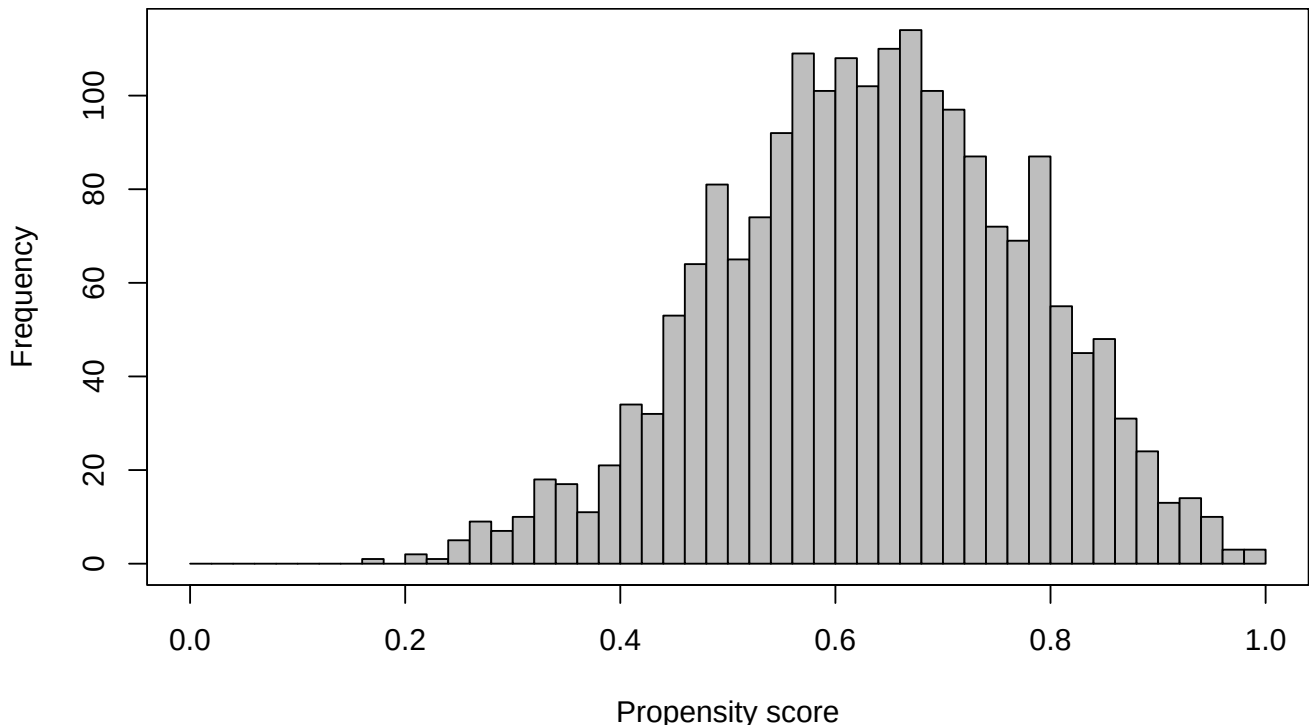
3.1 True propensity score

First, we assume the true propensity score values are known and are given as in the simulation.

```

par(mar=c(4,4,0,0))
hist(ps.true,main='',col='gray',breaks=seq(0,1,by=0.02),xlab='Propensity score');box()

```



We may define the strata by taking percentiles of the propensity score, for example, quintiles.

```

nstrata<-5
ps.quint.true<-quantile(ps.true,probs=seq(0,1,by=1/nstrata))
ps.quint.true

+          0%          20%          40%          60%          80%          100%
+ 0.1622321 0.5138183 0.5993914 0.6725239 0.7608811 0.9901202

ps.strat.true<-as.numeric(cut(ps.true,ps.quint.true,include.lowest=TRUE))
table(Z,ps.strat.true)

+   ps.strat.true
+ Z      1      2      3      4      5
+ 0 220 177 154 106  63
+ 1 180 223 246 294 337

```

Stratification by the quintiles of the propensity score allows a reasonable number of both $Z = 0$ and $Z = 1$ cases in each of the strata. There is good *overlap*.

```

mu.strat.true<-matrix(0,ncol=nstrata,nrow=3)
for(j in 1:nstrata){
  mu.strat.true[1,j]<-mean(Y[Z==0 & ps.strat.true == j])
  mu.strat.true[2,j]<-mean(Y[Z==1 & ps.strat.true == j])
  mu.strat.true[3,j]<-mu.strat.true[2,j]-mu.strat.true[1,j]
}
mu.strat.true

+          [,1]      [,2]      [,3]      [,4]      [,5]
+ [1,]  3.870217  4.239302  4.435440  4.653815  4.935454
+ [2,]  1.403318  1.984441  2.435372  2.927284  4.135429
+ [3,] -2.466899 -2.254860 -2.000068 -1.726532 -0.800025

```

We can then estimate the causal quantities by averaging within the rows of this matrix to compute

$$\hat{\mu}(z) = \sum_{j=1}^J \hat{\mu}^{\mathcal{X}_j}(z) \Pr[X \in \mathcal{X}_j]$$

```

apply(mu.strat.true,1,mean)

+ [1]  4.426846  2.577169 -1.849677

```

We can recover the estimate using `lm`

```

Stratum<-as.factor(ps.strat.true)
fit.strat<-lm(Y~Z*Stratum)
summary(fit.strat)$coef

+      Estimate Std. Error    t value    Pr(>|t|)
+ (Intercept)  3.8702167 0.03618292 106.962539 0.000000e+00
+ Z          -2.4668991 0.05393831 -45.735566 9.223708e-313
+ Stratum2     0.3690850 0.05418913   6.811053 1.278928e-11
+ Stratum3     0.5652236 0.05638700  10.024005 4.180339e-23
+ Stratum4     0.7835987 0.06345406  12.349070 8.244506e-34
+ Stratum5     1.0652372 0.07668784  13.890561 6.133965e-42
+ Z:Stratum2   0.2120386 0.07634259   2.777462 5.529903e-03
+ Z:Stratum3   0.4668310 0.07713925   6.051795 1.707173e-09
+ Z:Stratum4   0.7403673 0.08127873   9.108992 1.968732e-19
+ Z:Stratum5   1.6668741 0.09130080  18.256950 5.673127e-69

df0<-data.frame(Z=0,Stratum)

```

```

df1<-data.frame(Z=1,Stratum)
mu0.strat<-mean(predict(fit.strat,newdata=df0))
mu1.strat<-mean(predict(fit.strat,newdata=df1))
c(mu0.strat,mu1.strat,mu1.strat-mu0.strat)

+ [1] 4.426846 2.577169 -1.849677

Z0<-Z*0;Z1<-Z*1
StratumInd<-cbind(Stratum==2,Stratum==3,Stratum==4,Stratum==5)
Xm0<-cbind(1,Z0,StratumInd,StratumInd*Z0)
Xm1<-cbind(1,Z1,StratumInd,StratumInd*Z1)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.strat<-as.numeric(Dm %*% coef(fit.strat))

#Model-based standard error
Vm<-summary(fit.strat)$sigma^2*summary(fit.strat)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit.strat)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.02682737 0.01856835

```

Again there are differences between the estimated standard errors: the model-based version standard error is 0.0268274 whereas the sandwich standard error is 0.0185684. The corresponding confidence intervals are

$$(-1.9033317, -1.7960222) \quad (-1.8868136, -1.8125402)$$

and in this case the second interval narrowly excludes the true value.

3.2 Fitted propensity score

In practice, we do not know the true values of the propensity score, and so typically we would estimate them from the observed data. If we assume the correct form for the model, but not the values of the parameters, we can estimate them from the observed data using a logistic regression model and the `glm` function.

```
ps.fit.c<-fitted(glm(Z~X1+X2+X1:X2,family=binomial))
```

We can then proceed with an identical analysis, replacing the true PS values with the fitted ones.

```

ps.quint.fit.c<-quantile(ps.fit.c,probs=seq(0,1,by=1/nstrata))
ps.quint.fit.c

+      0%      20%      40%      60%      80%     100%
+ 0.1371910 0.5115502 0.6047622 0.6863503 0.7767072 0.9901116

ps.strat.fit.c<-as.numeric(cut(ps.fit.c,ps.quint.fit.c,include.lowest=TRUE))
table(Z,ps.strat.fit.c)

+   ps.strat.fit.c
+ Z    1    2    3    4    5
+ 0 222 171 157 109  61
+ 1 178 229 243 291 339

mu.strat.fit.c<-matrix(0,ncol=nstrata,nrow=3)
for(j in 1:nstrata){
  mu.strat.fit.c[1,j]<-mean(Y[Z==0 & ps.strat.fit.c == j])
  mu.strat.fit.c[2,j]<-mean(Y[Z==1 & ps.strat.fit.c == j])
  mu.strat.fit.c[3,j]<-mu.strat.fit.c[2,j]-mu.strat.fit.c[1,j]
}
mu.strat.fit.c

```

```

+           [,1]      [,2]      [,3]      [,4]      [,5]
+ [1,]  3.868610  4.232573  4.428262  4.665288  4.9630362
+ [2,]  1.443565  2.026285  2.432638  2.937693  4.0752703
+ [3,] -2.425045 -2.206288 -1.995624 -1.727595 -0.8877659

apply(mu.strat.fit.c,1,mean)

+ [1]  4.431554  2.583090 -1.848463

```

The results are broadly similar to when the true PS values are used. Using `lm`:

```

Stratum<-as.factor(ps.strat.fit.c)
fit.strat.c<-lm(Y~Z*Stratum)
summary(fit.strat.c)$coef

+           Estimate Std. Error   t value    Pr(>|t|)
+ (Intercept)  3.8686095  0.03846545  100.573610  0.000000e+00
+ Z          -2.4250450  0.05766215  -42.056095  4.078512e-277
+ Stratum2     0.3639631  0.05831350   6.241490  5.282126e-10
+ Stratum3     0.5596523  0.05976413   9.364350  1.998503e-20
+ Stratum4     0.7966784  0.06703037  11.885336  1.599788e-31
+ Stratum5     1.0944267  0.08285127  13.209535  2.995311e-38
+ Z:Stratum2   0.2187572  0.08173225   2.676511  7.500365e-03
+ Z:Stratum3   0.4294210  0.08227278   5.219478  1.981198e-07
+ Z:Stratum4   0.6974502  0.08641275   8.071149  1.192350e-15
+ Z:Stratum5   1.5372790  0.09837988  15.625949  5.242430e-52

df0<-data.frame(Z=0,Stratum)
df1<-data.frame(Z=1,Stratum)
mu0.strat<-mean(predict(fit.strat.c,newdata=df0))
mu1.strat<-mean(predict(fit.strat.c,newdata=df1))
c(mu0.strat,mu1.strat,mu1.strat-mu0.strat)

+ [1]  4.431554  2.583090 -1.848463

Z0<-Z*0;Z1<-Z*1
StratumInd<-cbind(Stratum==2,Stratum==3,Stratum==4,Stratum==5)
Xm0<-cbind(1,Z0,StratumInd,StratumInd*Z0)
Xm1<-cbind(1,Z1,StratumInd,StratumInd*Z1)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.strat.fit<-as.numeric(Dm %*% coef(fit.strat.c))

#Model-based standard error
Vm<-summary(fit.strat.c)$sigma^2*summary(fit.strat.c)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit.strat.c)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.02871931 0.01945195

```

The corresponding confidence intervals are

$(-1.9059021, -1.7910249)$ $(-1.8885808, -1.810773).$

However, if the PS model is misspecified, this does not hold; for example if we omit X_2 from the PS model, incorrect results are obtained.

```
ps.fit.i<-fitted(glm(Z~X1,family=binomial))
ps.quint.fit.i<-quantile(ps.fit.i,probs=seq(0,1,by=1/nstrata))
ps.quint.fit.i

+          0%          20%          40%          60%          80%          100%
+ 0.5985442 0.6287867 0.6367798 0.6435404 0.6512389 0.6823522

ps.strat.fit.i<-as.numeric(cut(ps.fit.i,ps.quint.fit.i,include.lowest=TRUE))
table(Z,ps.strat.fit.i)

+   ps.strat.fit.i
+ Z      1      2      3      4      5
+ 0 130 153 169 154 114
+ 1 270 247 231 246 286

mu.strat.fit.i<-matrix(0,ncol=nstrata,nrow=3)
for(j in 1:nstrata){
  mu.strat.fit.i[1,j]<-mean(Y[Z==0 & ps.strat.fit.i == j])
  mu.strat.fit.i[2,j]<-mean(Y[Z==1 & ps.strat.fit.i == j])
  mu.strat.fit.i[3,j]<-mu.strat.fit.i[2,j]-mu.strat.fit.i[1,j]
}
mu.strat.fit.i

+          [,1]      [,2]      [,3]      [,4]      [,5]
+ [1,]  4.427240  4.300523  4.262165  4.254483  4.2112488
+ [2,]  1.975194  2.255472  2.478070  2.985863  4.0249287
+ [3,] -2.452047 -2.045051 -1.784096 -1.268620 -0.1863201

apply(mu.strat.fit.i,1,mean)

+ [1]  4.291132  2.743906 -1.547227
```

With a mis-specified PS model, we no longer have a balancing score for the known confounding variables, that is, Z and X are no longer independent within strata of the propensity score. Lack of balance can be diagnosed numerically (by comparing means of the confounders for $Z = 0$ and $Z = 1$ within strata of the propensity score) or graphically (by comparing boxplots within strata). For the correct model, we have

```
t1<-aggregate(X1,by=list(Z,ps.strat.fit.c),FUN=mean)
t2<-aggregate(X2,by=list(Z,ps.strat.fit.c),FUN=mean)
t3<-aggregate(X3,by=list(Z,ps.strat.fit.c),FUN=mean)
dnames<-list(c("Z=0", "Z=1"),c("Q1", "Q2", "Q3","Q4","Q5"))
matrix(t1$x,nrow=2,ncol=nstrata,byrow=FALSE,dimnames = dnames ) #Summary for X1: mean

+          Q1          Q2          Q3          Q4          Q5
+ Z=0 1.0176148 0.9814824 0.9691275 0.9531705 1.013351
+ Z=1 0.9941112 0.9585973 0.9541224 0.9651071 1.104770

matrix(t2$x,nrow=2,ncol=nstrata,byrow=FALSE,dimnames = dnames ) #Summary for X2: mean

+          Q1          Q2          Q3          Q4          Q5
+ Z=0 -2.275483 -2.147864 -2.072930 -1.976196 -1.716560
+ Z=1 -2.279909 -2.182319 -2.082256 -1.940532 -1.548473

matrix(t3$x,nrow=2,ncol=nstrata,byrow=FALSE,dimnames = dnames ) #Summary for X3: mean

+          Q1          Q2          Q3          Q4          Q5
+ Z=0 -0.7900573 -0.8582284 -0.9541356 -1.023998 -1.406061
+ Z=1 -0.7302646 -0.9780879 -1.0732272 -1.067611 -1.257340
```

and the variables fitted within the PS are balanced. For the incorrect model we have

```

t1<-aggregate(X1,by=list(Z,ps.strat.fit.i),FUN=mean)
t2<-aggregate(X2,by=list(Z,ps.strat.fit.i),FUN=mean)
t3<-aggregate(X3,by=list(Z,ps.strat.fit.i),FUN=mean)
dnames<-list(c("Z=0", "Z=1"),c("Q1", "Q2", "Q3","Q4","Q5"))
matrix(t1$x,nrow=2,ncol=nstrata,byrow=FALSE,dimnames = dnames )    #Summary for X1: mean

+           Q1           Q2           Q3           Q4           Q5
+ Z=0 0.6524630 0.8590105 1.000952 1.132691 1.331253
+ Z=1 0.6389345 0.8625539 0.994136 1.130419 1.365013

matrix(t2$x,nrow=2,ncol=nstrata,byrow=FALSE,dimnames = dnames )    #Summary for X2: mean

+           Q1           Q2           Q3           Q4           Q5
+ Z=0 -2.694549 -2.311095 -2.074785 -1.852525 -1.563079
+ Z=1 -2.616351 -2.203291 -1.988056 -1.719009 -1.288271

matrix(t3$x,nrow=2,ncol=nstrata,byrow=FALSE,dimnames = dnames )    #Summary for X3: mean

+           Q1           Q2           Q3           Q4           Q5
+ Z=0 -0.8630788 -0.8917421 -0.9608001 -0.9818738 -0.9395964
+ Z=1 -0.9113741 -1.0388729 -1.0041257 -1.1170307 -1.1967174

```

and now only X_1 appears balanced. Therefore, the PS mechanism is producing balance amongst those variables that appear in the fitted PS model. However, those variables that remain unbalanced may still have some confounding influence. This analysis informs us that the functional form of the propensity score is correct for the included variables.

4. Matching

Propensity score matching uses the propensity score to find matched (and therefore comparable) individuals in the $Z = 0$ and $Z = 1$ subsets. These matched individuals can then be used to estimate the difference $\mu(1) - \mu(0)$. A statistical analysis using the Matching package is straightforward. In the first analysis, we match 1–1 for observations with $Z = 0$ and $Z = 1$. There are 1280 cases with $Z = 1$ in the simulated data, but the package re-uses the data to produce a sample of size n of matched pairs. In the function Match, the quantity Tr denotes the ‘treatment’ Z , and X is the variable (or variables) used for matching.

```

library(Matching)

+ ##
+ ## Matching (Version 4.10-15, Build Date: 2024-10-14)
+ ## See https://www.jsekhon.com for additional documentation.
+ ## Please cite software as:
+ ## Jasjeet S. Sekhon. 2011. ‘‘Multivariate and Propensity Score Matching
+ ## Software with Automated Balance Optimization: The Matching package for R.’’
+ ## Journal of Statistical Software, 42(7): 1-52.
+ ##

fit.match1 <- Match(Y=Y, Tr=Z, X=ps.true, M=1, estimand='ATE')
summary(fit.match1)

+
+ Estimate... -1.9018
+ AI SE..... 0.033789
+ T-stat..... -56.285
+ p.val..... < 2.22e-16
+
+ Original number of observations..... 2000
+ Original number of treated obs..... 1280
+ Matched number of observations..... 2000
+ Matched number of observations (unweighted). 4141

```

In this case the estimate is -1.9018431, and the estimated standard error is 0.0337893. The corresponding confidence interval is

$$(-1.9694218, -1.8342645).$$

It is possible also to carry out 1- M matching for $M = 5$ and $M = 10$, say. In this analysis, every case with $Z = 1$ is matched to M individuals with $Z = 0$.

```
fit.match2 <- Match(Y=Y, Tr=Z, X=ps.true, M=5, estimand='ATE')
summary(fit.match2)

+
+ Estimate... -1.8945
+ AI SE..... 0.032357
+ T-stat..... -58.549
+ p.val..... < 2.22e-16
+
+ Original number of observations..... 2000
+ Original number of treated obs..... 1280
+ Matched number of observations..... 2000
+ Matched number of observations (unweighted). 10691

fit.match3 <- Match(Y=Y, Tr=Z, X=ps.true, M=10, estimand='ATE')
summary(fit.match3)

+
+ Estimate... -1.8933
+ AI SE..... 0.032232
+ T-stat..... -58.74
+ p.val..... < 2.22e-16
+
+ Original number of observations..... 2000
+ Original number of treated obs..... 1280
+ Matched number of observations..... 2000
+ Matched number of observations (unweighted). 20307
```

Here the estimate is -1.8944654, and the estimated standard error is 0.0323572, with confidence interval

$$(-1.9591797, -1.829751).$$

There are many other options in the Matching package.

5. Propensity score regression

Basic PSR model: In propensity score regression, a regression model is built using the propensity score as a predictor. For example, we might fit

$$\mu(z, x) = \mathbb{E}_{Y|X,Z}^{\mathcal{O}}[Y|X = x, Z = z] = \beta_0 + \psi_0 z + \phi e(x)$$

with the idea that conditioning on the propensity score is achieved by entering the term into the regression model.

```
fit.psr.1<-lm(Y~Z+ps.fit.c)
summary(fit.psr.1)$coef

+           Estimate Std. Error  t value    Pr(>|t|)
+ (Intercept)  1.285470 0.05831581  22.04325 1.461869e-96
+ Z           -2.015132 0.02903293 -69.40850 0.000000e+00
+ ps.fit.c      5.192202 0.09324580  55.68296 0.000000e+00
```

The estimator that follows from this procedure is a regression estimator as in equation (1). In a this model, we can read off the estimator of $\mu(1) - \mu(0)$ as the coefficient of Z , which here is equal to -2.0151318, and model-based estimated standard error 0.0290329. We can also compute the sandwich standard error

```
sqrt(sandwich(fit.psr.1)[2,2])
+ [1] 0.02157739
```

which is only slightly larger.

The basic PSR model suffers the same problems as the regression model from Section 2., which relate to misspecification: if $\mu(x, z)$ is mis-specified, incorrect inference will follow.

Extended PSR model: The regression model may be extended to include other terms. Suppose we specify

$$\mu(x, z) = \beta_0 + z(\psi_0 + \psi_1 x_1 + \psi_2 x_2 + \psi_3 x_1 x_2) + \phi_0 e(x)$$

so that the treatment effect model is correctly specified.

```
fit.psr.2<-lm(Y~Z+Z:X1+Z:X2+Z:X1:X2+ps.fit.c)
round(coef(summary(fit.psr.2)),6)

+           Estimate Std. Error   t value Pr(>|t|)
+ (Intercept)  2.896317   0.019007 152.383379 0.000000
+ Z            2.975352   0.089493  33.246697 0.000000
+ ps.fit.c     2.408844   0.031811  75.723857 0.000000
+ Z:X1        -0.165142   0.050264  -3.285473 0.001036
+ Z:X2         1.572448   0.026272  59.853530 0.000000
+ Z:X1:X2      0.812471   0.021619  37.581122 0.000000

mu0.psr<-mean(predict(fit.psr.2,newdata=data.frame(X1,X2,Z=0,ps.fit.c)))
mu1.psr<-mean(predict(fit.psr.2,newdata=data.frame(X1,X2,Z=1,ps.fit.c)))
c(mu0.psr,mu1.psr,mu1.psr-mu0.psr)

+ [1]  4.437977  2.553379 -1.884598
```

then an approximately correct conclusion is obtained for estimating $\mu(0)$ and $\mu(1)$ – this is despite the fact that the ψ parameters are not estimated correctly, that is, the estimates do not match the ψ parameters in the data generating model. Compare this with the incorrectly specified outcome regression

$$\mu(x, z) = \beta_0 + z(\psi_0 + \psi_1 x_1 + \psi_2 x_2 + \psi_3 x_1 x_2) = m_0(\beta_0) + z m_1(x; \psi)$$

say. Then

```
fit.or2<-lm(Y~Z+Z:X1+Z:X2+Z:X1:X2)
round(coef(summary(fit.or2)),6)

+           Estimate Std. Error   t value Pr(>|t|)
+ (Intercept)  4.290417   0.009298 461.43201    0
+ Z            6.675691   0.147563  45.23971    0
+ Z:X1        -0.992553   0.096563 -10.27877    0
+ Z:X2         2.192876   0.049128  44.63562    0
+ Z:X1:X2      1.593131   0.037401  42.59611    0

mu0.or2<-mean(predict(fit.or2,newdata=data.frame(X1,X2,Z=0)))
mu1.or2<-mean(predict(fit.or2,newdata=data.frame(X1,X2,Z=1)))
c(mu0.or2,mu1.or2,mu1.or2-mu0.or2)

+ [1]  4.290417  2.558785 -1.731632
```

which does not yield the correct result. We have in this model that the *interaction* model $m_1(x; \psi) \equiv \mu_1(x; \psi)$ is correctly specified in that it coincides with the true interaction model. However we have that $m_0(\beta_0)$ does **not** match the true *outcome nuisance model* $\mu_0(x; \beta) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$, and as the parameter β_0 in $m_0(\cdot)$ is estimated, we do not obtain correct inference.

Correct PSR model: Suppose finally we fit the model that includes interactions between $e(x)$ and x_1 and x_2 .

$$\mu(x, z) = \beta_0 + z(\psi_0 + \psi_1 x_1 + \psi_2 x_2 + \psi_3 x_1 x_2) + e(x)(\phi_0 + \phi_1 x_1 + \phi_2 x_2 + \phi_3 x_1 x_2)$$

```
fit.psr.3<-lm(Y~Z+X1+Z:X2+Z:X1:X2+ps.fit.c+ps.fit.c:X1+ps.fit.c:X2 + ps.fit.c:X1:X2)
round(coef(summary(fit.psr.3)),6)

+           Estimate Std. Error   t value Pr(>|t|)
+ (Intercept)  3.293699   0.022397  147.058794  0.00000
+ Z           0.845876   0.127432   6.637850  0.00000
+ ps.fit.c     6.229603   0.155483  40.066130  0.00000
+ Z:X1        1.073735   0.094502  11.362004  0.00000
+ Z:X2        0.962851   0.045039  21.378190  0.00000
+ X1:ps.fit.c -2.126455   0.109876 -19.353144  0.00000
+ X2:ps.fit.c  1.123178   0.054093  20.763898  0.00000
+ Z:X1:X2     0.993968   0.039895  24.914739  0.00000
+ X1:X2:ps.fit.c 0.047344   0.047179   1.003503  0.31574

mu0.psr3<-mean(predict(fit.psr.3,newdata=data.frame(X1,X2,Z=0,ps.fit.c)))
mu1.psr3<-mean(predict(fit.psr.3,newdata=data.frame(X1,X2,Z=1,ps.fit.c)))
c(mu0.psr3,mu1.psr3,mu1.psr3-mu0.psr3)

+ [1]  4.455743  2.559727 -1.896017
```

The resulting estimation of the causal quantities does not change appreciably. However,

```
round(cbind(coef(summary(fit.psr.3))[c(2,4,5,8),],psi),6)

+           Estimate Std. Error   t value Pr(>|t|)  psi
+ Z           0.845876   0.127432   6.63785    0    1
+ Z:X1        1.073735   0.094502  11.36200    0    1
+ Z:X2        0.962851   0.045039  21.37819    0    1
+ Z:X1:X2     0.993968   0.039895  24.91474    0    1
```

we observe that now the ψ values are now also correctly estimated. For the standard errors and confidence intervals:

```
Z0<-Z*0;Z1<-Z*1
Xm0<-cbind(1,Z0,ps.fit.c,Z0*X1,Z0*X2,X1*ps.fit.c,X2*ps.fit.c,Z0*X1*X2,ps.fit.c*X1*X2)
Xm1<-cbind(1,Z1,ps.fit.c,Z1*X1,Z1*X2,X1*ps.fit.c,X2*ps.fit.c,Z1*X1*X2,ps.fit.c*X1*X2)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.psr.fit<-as.numeric(Dm %*% coef(fit.psr.3))

#Model-based standard error
Vm<-summary(fit.psr.3)$sigma^2*summary(fit.psr.3)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit.psr.3)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.005450159 0.005145792
```

The corresponding confidence intervals are

$$(-1.9069169, -1.8851163) \quad (-1.9063082, -1.885725).$$

It can be shown that this is the appropriate PSR model to fit: every term in Z in the treatment effect model must be matched by an equivalent term in $PS_e(X)$.

6. Inverse Weighting

We now study the inverse weighting (or inverse probability weighting, IPW) approach that is based on an importance sampling (or change of measure) procedure. We have that

$$\mu(z) = \mathbb{E}_{Y|Z}^{\varepsilon}[Y|Z=z] = \frac{\mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)Y}{f_{Z|X}^{\circ}(Z|X)} \right]}{\mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)}{f_{Z|X}^{\circ}(Z|X)} \right]}$$

which becomes in the binary treatment case

$$\mu(0) = \frac{\mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{(1-Z)Y}{1-e(X)} \right]}{\mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{(1-Z)}{1-e(X)} \right]} \quad \mu(1) = \frac{\mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{ZY}{e(X)} \right]}{\mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{Z}{e(X)} \right]}$$

which then gives the ATE in the usual way as the difference of these two quantities.

6.1 IPW

We deduce that suitable IPW estimators following from the importance sampling results are given by

$$\hat{\mu}_{\text{IPW}}(0) = \frac{\frac{1}{n} \sum_{i=1}^n \frac{(1-Z_i)Y_i}{1-e(X_i)}}{\frac{1}{n} \sum_{i=1}^n \frac{(1-Z_i)}{1-e(X_i)}} \quad \hat{\mu}_{\text{IPW}}(1) = \frac{\frac{1}{n} \sum_{i=1}^n \frac{Z_i Y_i}{e(X_i)}}{\frac{1}{n} \sum_{i=1}^n \frac{Z_i}{e(X_i)}}$$

which we may write

$$\hat{\mu}_{\text{IPW}}(0) = \frac{\sum_{i=1}^n w_{0i} Y_i}{\sum_{i=1}^n w_{0i}} = \sum_{i=1}^n W_{0i} Y_i \quad \hat{\mu}_{\text{IPW}}(1) = \frac{\sum_{i=1}^n w_{1i} Y_i}{\sum_{i=1}^n w_{1i}} = \sum_{i=1}^n W_{1i} Y_i$$

with

$$w_{0i} = \frac{(1-Z_i)}{1-e(X_i)} \quad w_{1i} = \frac{Z_i}{e(X_i)} \quad W_{zi} = \frac{w_{zi}}{\sum_{j=1}^n w_{zj}} \quad z = 0, 1.$$

Note that by construction,

$$\mathbb{E}_{X,Y,Z}^{\circ}[\hat{\mu}_{\text{IPW}}(z)] = \mu(z) \quad z = 0, 1$$

so the estimators are unbiased, and also that

$$\begin{aligned} \mathbb{E}_{X,Y,Z}^{\circ}[W_{zi}] &= \mathbb{E}_{X,Y,Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)}{f_{Z|X}^{\circ}(Z|X)} \right] = \mathbb{E}_X^{\circ} \left[\mathbb{E}_{Z|X}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)}{f_{Z|X}^{\circ}(Z|X)} \right] \right] \\ &= \mathbb{E}_X^{\circ} \left[\mathbb{E}_{Z|X}^{\circ} \left[\frac{1}{f_{Z|X}^{\circ}(Z|X)} \right] \right] = 1. \end{aligned}$$

This latter result suggests the alternative IPW estimators where we replace the denominator by its expectation, n , that is,

$$\tilde{\mu}_{\text{IPW}}(0) = \frac{1}{n} \sum_{i=1}^n \frac{(1-Z_i)Y_i}{1-e(X_i)} \quad \tilde{\mu}_{\text{IPW}}(1) = \frac{1}{n} \sum_{i=1}^n \frac{Z_i Y_i}{e(X_i)}.$$

Using fitted PS, we can compute as follows: for $\hat{\mu}(z)$, we have

```
w0<-(1-Z)/(1-ps.fit.c)
W0<-w0/sum(w0)
mu.hat0<-sum(W0*Y)
w1<-Z/ps.fit.c
W1<-w1/sum(w1)
mu.hat1<-sum(W1*Y)
c(mu.hat0,mu.hat1,mu.hat1-mu.hat0)

+ [1] 4.445495 2.553335 -1.892160
```

and for $\tilde{\mu}(z)$, we have

```
mu.tilde0<-mean(w0*Y)
mu.tilde1<-mean(w1*Y)
c(mu.tilde0,mu.tilde1,mu.tilde1-mu.tilde0)

+ [1] 4.398303 2.557142 -1.841161
```

Note: We may recover the $\hat{\mu}(z)$ quantities using a weighted least squares (WLS) regression of Y on Z

$$\mathbb{E}_{Y|Z}^{\mathcal{O}}[Y|Z = z] = \beta_0 + \psi_0 z$$

with weights

$$w_{0i} + w_{1i} = \frac{(1 - Z_i)}{1 - e(X_i)} + \frac{Z_i}{e(X_i)} = R_i$$

say.

```
w<-w0+w1
fit.wls<-lm(Y~Z,weights=w)
round(coef(summary(fit.wls)),6)

+           Estimate Std. Error  t value Pr(>|t|)
+ (Intercept) 4.445495  0.026723 166.35186    0
+ Z          -1.892160  0.037678 -50.21886    0
```

The estimated coefficients are precisely $\hat{\mu}(0)$ and $\hat{\mu}(1) - \hat{\mu}(0)$. This result gives us further insight into how the inverse probability weighting succeeds in correcting the confounding; it creates a pseudo-sample from the distribution from the experimental measure $\mathcal{E}$: that is

- in the observed sample, collected under $\mathcal{O}$, each data point carries a weight $1/n$.
- for the pseudo-sample, we wish to have that each data point carries a weight $1/n$ under $\mathcal{E}$, but in fact datum i carries a weight that is dependent on X_i , that is

$$f_{Z|X}^{\mathcal{O}}(z|x_i) = \{e(x_i)\}^z \{1 - e(x_i)\}^{1-z} \quad z = 0, 1.$$

We can again obtain the standard errors using the model-based and sandwich approaches. For example,

```
sqrt(sandwich(fit.wls)[2,2])

+ [1] 0.03821415
```

with confidence interval

$$(-1.9685881, -1.8157315).$$

This approach does not account for the uncertainty potentially incurred due to the estimation of the propensity score and inverse probability weights; however, this is typically quite small.

6.2 Augmented IPW

In augmented IPW (AIPW) estimation, we utilize the fact that if we denote

$$\mu(x, z) = \mathbb{E}_{Y|X, Z}^{\varepsilon}[Y|X = x, Z = z] = \mathbb{E}_{Y|X, Z}^{\circ}[Y|X = x, Z = z]$$

we may write

$$\begin{aligned} \mathbb{E}_{X, Y, Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)Y}{f_{Z|X}^{\circ}(Z|X)} \right] &= \mathbb{E}_{X, Y, Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)(Y - \mu(X, Z) + \mu(X, Z))}{f_{Z|X}^{\circ}(Z|X)} \right] \\ &= \mathbb{E}_{X, Y, Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)(Y - \mu(X, Z))}{f_{Z|X}^{\circ}(Z|X)} \right] + \mathbb{E}_{X, Y, Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)\mu(X, Z)}{f_{Z|X}^{\circ}(Z|X)} \right] \\ &= \mathbb{E}_{X, Y, Z}^{\circ} \left[\frac{\mathbb{1}_{\{z\}}(Z)(Y - \mu(X, Z))}{f_{Z|X}^{\circ}(Z|X)} \right] + \mathbb{E}_X^{\circ} [\mu(X, z)] \end{aligned}$$

with the last simplification following by iterated expectation. This result suggests the alternative estimator

$$\hat{\mu}_{\text{AIPW}}(z) = \sum_{i=1}^n W_{zi}(Y_i - \mu(X_i, z)) + \frac{1}{n} \sum_{i=1}^n \mu(X_i, z) \quad z = 0, 1.$$

By the above construction, $\hat{\mu}_{\text{AIPW}}(z)$ is also an unbiased estimator of $\mu(z)$. We may also utilize the estimator

$$\tilde{\mu}_{\text{AIPW}}(z) = \frac{1}{n} \sum_{i=1}^n w_{zi}(Y_i - \mu(X_i, z)) + \frac{1}{n} \sum_{i=1}^n \mu(X_i, z) \quad z = 0, 1$$

which is also unbiased for $\mu(z)$.

These augmented (AIPW) estimators are often preferred to the original (IPW) estimators as they can have lower variance. For the calculation with known, correctly specified $\mu(x, z)$ we have

```
mu.x0<- Xb %*% be
mu.x1<- Xb %*% be + Xp %*% psi
mu.hat.a0<-sum(W0*(Y-mu.x0))+mean(mu.x0)
mu.hat.a1<-sum(W1*(Y-mu.x1))+mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.454876 2.559186 -1.895690

mu.tilde.a0<-mean(w0*(Y-mu.x0))+mean(mu.x0)
mu.tilde.a1<-mean(w1*(Y-mu.x1))+mean(mu.x1)
c(mu.tilde.a0,mu.tilde.a1,mu.tilde.a1-mu.tilde.a0)

+ [1] 4.454925 2.559188 -1.895737
```

In practice we need to estimate the parameters in the specification $\mu(x, z; \beta, \psi)$: under correct specification

```
fit0<-lm(Y~X1+X2+X1:X2+Z+Z:X1+Z:X2+Z:X1:X2)
mu.x0<-predict(fit0,newdata=data.frame(X1,X2,Z=0))
mu.x1<-predict(fit0,newdata=data.frame(X1,X2,Z=1))
mu.hat.a0<-sum(W0*(Y-mu.x0))+mean(mu.x0)
mu.hat.a1<-sum(W1*(Y-mu.x1))+mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.454933 2.559174 -1.895760

mu.tilde.a0<-mean(w0*(Y-mu.x0))+mean(mu.x0)
mu.tilde.a1<-mean(w1*(Y-mu.x1))+mean(mu.x1)
c(mu.tilde.a0,mu.tilde.a1,mu.tilde.a1-mu.tilde.a0)

+ [1] 4.454946 2.559174 -1.895772
```

This realization highlights the fact that under an assumption of correct specification, we could merely perform outcome regression, which is known to optimal, so that in fact the IPW version is redundant. Specifically, under correct specification of $\mu(X, z; \beta, \psi)$, we have by iterated expectation that

$$\mathbb{E}[W_z(Y - \mu(X, z; \beta, \psi))] = \mathbb{E}[W_z(Y - \mu(x, z; \hat{\beta}, \hat{\psi}))] = 0.$$

However, suppose that we mis-specify the conditional mean model as $m(x, z; \beta, \psi)$, say. Then provided the propensity score model is correctly specified, we still have that

$$\text{Var}[W_z Y + (1 - W_z)m(X, z; \hat{\beta}, \hat{\psi})] < \text{Var}[W_z Y]$$

but also that

$$\begin{aligned} \mathbb{E}[W_z Y + (1 - W_z)\mu(X, z; \hat{\beta}, \hat{\psi})] &= \mathbb{E}[W_z Y + (1 - W_z)(\mu(X, z; \hat{\beta}, \hat{\psi}) - \mu(X, z)) + (1 - W_z)\mu(X, z)] \\ &= \mathbb{E}[W_z \mu(X, z) + (1 - W_z)(\mu(X, z; \hat{\beta}, \hat{\psi}) - \mu(X, z)) + (1 - W_z)\mu(X, z)] \\ &= \mathbb{E}[\mu(X, z)] = \mu(z) \end{aligned}$$

by iterated expectation, as $\mathbb{E}_{Z|X}^\circ[(1 - W_z)|X] = 0$. Therefore, we obtain an unbiased estimator even if the conditional mean model is mis-specified. For example, if we specify

$$m(x, z) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \psi_0 z$$

then the result holds.

```
fit0<-lm(Y~X1+X2+Z)
mu.x0<-predict(fit0,newdata=data.frame(X1,X2,Z=0))
mu.x1<-predict(fit0,newdata=data.frame(X1,X2,Z=1))
mu.hat.a0<-sum(W0*(Y-mu.x0))+mean(mu.x0)
mu.hat.a1<-sum(W1*(Y-mu.x1))+mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.470169 2.556808 -1.913361

mu.tilde.a0<-mean(w0*(Y-mu.x0))+mean(mu.x0)
mu.tilde.a1<-mean(w1*(Y-mu.x1))+mean(mu.x1)
c(mu.tilde.a0,mu.tilde.a1,mu.tilde.a1-mu.tilde.a0)

+ [1] 4.471219 2.556721 -1.914499
```

If the propensity score model is mis-specified, but the conditional mean model is correctly specified, we again obtain an unbiased estimator: suppose that the propensity score model proposed is $f_{Z|X}^\dagger(Z|X)$, and denote

$$w_{zi}^\dagger = \frac{\mathbb{1}_{\{z\}}(Z_i)}{f_{Z|X}^\dagger(Z_i|X_i)}.$$

Then

$$\mathbb{E}[w_{zi}^\dagger] = \mathbb{E}_{Z|X}^\circ \left[\frac{\mathbb{1}_{\{z\}}(Z_i)}{f_{Z|X}^\dagger(Z_i|X_i)} \middle| X_i \right] = \frac{f_{Z|X}^\circ(z|X_i)}{f_{Z|X}^\dagger(z|X_i)} = r^\dagger(X_i, z)$$

say, and we still have an unbiased estimator as

$$\mathbb{E}[(w_{zi}^\dagger Y_i + (1 - w_{zi}^\dagger)\mu(X_i, z))] = \mathbb{E}[(w_{zi}^\dagger \mu(X_i, z) + (1 - w_{zi}^\dagger)\mu(X_i, z))] = \mu(z).$$

Recall that the IPW estimator is not unbiased if the PS model is mis-specified, as we would have

$$\mathbb{E}_{X,Y,Z}^\circ[\tilde{\mu}(z)] = \mathbb{E}_{X,Y,Z}^\circ \left[\frac{\mathbb{1}_{\{z\}}(Z_i)Y_i}{f_{Z|X}^\dagger(Z_i|X_i)} \right] = \mathbb{E}_X^\circ [r^\dagger(X_i, z)\mu(X_i, z)] \neq \mu(z).$$

Thus the AIPW estimator is unbiased even if precisely one of the propensity or conditional mean models is mis-specified. This property is termed *double robustness*.

Note: The double robustness to mis-specification of the AIPW estimator might imply that it is superior to the singly robust outcome regression or IPW estimators. This is not necessarily true in practice, as a badly mis-specified model for the PS or the conditional mean model may yield an AIPW estimator that is not appreciably better than either singly robust estimator.

If both PS and outcome regression model are mis-specified, then a biased estimator results

```
fit0<-lm(Y~X1+Z)
w0.i<-(1-Z)/(1-ps.fit.i)
W0.i<-w0.i/sum(w0.i)
w1.i<-Z/ps.fit.i
W1.i<-w1.i/sum(w1.i)
mu.hat.i0<-sum(W0.i*Y)
mu.hat.i1<-sum(W1.i*Y)
c(mu.hat.i0,mu.hat.i1,mu.hat.i1-mu.hat.i0)

+ [1] 4.288360 2.755926 -1.532434

mu.x0<-predict(fit0,newdata=data.frame(X1,X2,Z=0))
mu.x1<-predict(fit0,newdata=data.frame(X1,X2,Z=1))
mu.hat.a0<-sum(W0.i*(Y-mu.x0))+mean(mu.x0)
mu.hat.a1<-sum(W1.i*(Y-mu.x1))+mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.292073 2.757085 -1.534988

mu.tilde.a0<-mean(w0.i*(Y-mu.x0))+mean(mu.x0)
mu.tilde.a1<-mean(w1.i*(Y-mu.x1))+mean(mu.x1)
c(mu.tilde.a0,mu.tilde.a1,mu.tilde.a1-mu.tilde.a0)

+ [1] 4.292077 2.757085 -1.534992
```

It is possible to construct the AIPW estimate, and its standard error, using the RCAL package in R:

```
library(RCAL)

+ Loading required package: trust

#Correct mean spec
fit0<-lm(Y~X1+X2+X1:X2+Z+Z:X1+Z:X2+Z:X1:X2)
mu.x0<-predict(fit0,newdata=data.frame(X1,X2,Z=0))
mu.x1<-predict(fit0,newdata=data.frame(X1,X2,Z=1))
fit.aipw.correct<-ate.aipw(Y, Z, cbind(w0,w1), cbind(mu.x0,mu.x1))
c(fit.aipw.correct$est,fit.aipw.correct$diff.est[2])

+ [1] 4.456157 2.558751 -1.897406
```

For the standard errors

```
fit.aipw.correct<-ate.aipw(Y, Z, cbind(w0,w1), cbind(mu.x0,mu.x1))
sqrt(c(fit.aipw.correct$var,fit.aipw.correct$diff.var[2]))

+ [1] 0.009248376 0.024616616 0.018058187
```

Here the sandwich calculations are more involved due to the composite form of the estimator. To compute the sandwich variance, we note that the estimating function can be written

$$\mathbf{U}(\mu_0, \mu_1, \beta) = \begin{bmatrix} W_0(Y - \mathbf{X}\beta) + \mathbf{X}_1\beta - \mu_0 \\ W_1(Y - \mathbf{X}\beta) + \mathbf{X}_0\beta - \mu_1 \\ \mathbf{X}^\top(Y - \mathbf{X}\beta) \end{bmatrix}$$

where $\mu_0 = \mu(0)$ and $\mu_1 = \mu(1)$ are the parameters representing the average potential outcomes under no treatment and treatment respectively, and $\mathbf{X}_0$ and $\mathbf{X}_1$ are the corresponding design matrices. Note that $\hat{\mu}_0$ and

$\hat{\mu}_1$ are the solutions to the system of equations

$$\sum_{i=1}^n \mathbf{U}_i(\mu_0, \mu_1, \beta) = \mathbf{0}$$

but due to the structure, we may solve for β in the third component separately from the first two components. The matrix of partial derivatives of $\mathbf{U}$ (multiplied by -1) is

$$-\dot{\mathbf{U}}(\mu_0, \mu_1, \beta) = \begin{bmatrix} 1 & 0 & W_0\mathbf{X} - \mathbf{X}_0 \\ 0 & 1 & W_1\mathbf{X} - \mathbf{X}_1 \\ 0 & 0 & \mathbf{X}^\top \mathbf{X} \end{bmatrix}$$

and then in the sandwich estimator

$$\mathbf{A} = \mathbb{E}[-\dot{\mathbf{U}}] \quad \mathbf{B} = \mathbb{E}[\mathbf{U}\mathbf{U}^\top]$$

and these matrices can be estimated using the empirical versions.

```
#Correct mean spec
fit0<-lm(Y~X1+X2+X1:X2+Z+Z:X1+Z:X2+Z:X1:X2)
mu.x0<-predict(fit0,newdata=data.frame(X1,X2,Z=0))
mu.x1<-predict(fit0,newdata=data.frame(X1,X2,Z=1))
Xm<-cbind(1,X1,X2,Z,X1*X2,Z*X1,Z*X2,Z*X1*X2)
Yfit<- Xm %*% coef(fit0)
Yres<-as.numeric(Y-Yfit)

mu0.est<-mean(W0*Yres + mu.x0)
mu1.est<-mean(W1*Yres + mu.x1)
aipw.est<-mu1.est-mu0.est
c(mu0.est,mu1.est,aipw.est)

+ [1] 4.456167 2.558785 -1.897381

Umat<-matrix(0,nrow=ncol(Xm)+2,ncol=n)
Umat[1,]<-W0*Yres + mu.x0 - mu0.est
Umat[2,]<-W1*Yres + mu.x1 - mu1.est
Umat[3:10,]<-t(Xm*Yres)

Xm0<-cbind(1,X1,X2,0,X1*X2,0,0,0)
Xm1<-cbind(1,X1,X2,1,X1*X2,X1,X2,X1*X2)

B<-(Umat %*% t(Umat))/n
A<-matrix(0,10,10)
A[1,1]<-A[2,2]<-n
A[1,3:10]<-apply(Xm*W0-Xm0,2,sum)
A[2,3:10]<-apply(Xm*W1-Xm1,2,sum)
A[3:10,3:10]<-t(Xm) %*% Xm
A<-A/n
V<-solve(A) %*% (B %*% solve(t(A)))/n
a<-matrix(c(-1,1,rep(0,8)),nrow=1)
se.s<-sqrt(a %*% (V %*% t(a)))
se.s

+ [1]
+ [1,] 0.01873531
```

Therefore the directly computed sandwich standard error is 0.0187353.

6.3 Augmented Outcome Regression (AOR)

A version of the AIPW estimator may be also constructed by using augmented outcome regression (AOR), with

$$\hat{\mu}_{\text{AOR}}(z) = \frac{1}{n} \sum_{i=1}^n \hat{\mu}_A(X_i, z) \quad z = 0, 1.$$

where

$$\mu_A(x, z) \equiv \mu(x, z; \beta) + \phi_0 \frac{(1-z)}{1-e(x)} + \phi_1 \frac{z}{e(x)}$$

and parameters are estimated using ordinary least squares. This approach augments the regression model given by $\mu(x, z; \beta)$ with covariates that depend on the inverse probability weights for untreated and treated respectively, with associated coefficients ϕ_0 and ϕ_1 respectively. This estimator is also doubly robust, and is equivalent to the AIPW estimator when the sample size is large.

```
#Correct mean spec
aug0<-(1-Z)/(1-ps.fit.c)
aug1<-Z/ps.fit.c
fit.aor.0<-lm(Y~X1+X2+X1:X2+Z+Z:X1+Z:X2+Z:X1:X2+aug0+aug1)
mu.x0<-predict(fit.aor.0,newdata=data.frame(X1,X2,Z=0,aug0=1/(1-ps.fit.c),aug1=0))
mu.x1<-predict(fit.aor.0,newdata=data.frame(X1,X2,Z=1,aug0=0,aug1=1/ps.fit.c))
mu.hat.a0<-mean(mu.x0)
mu.hat.a1<-mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.453836 2.559172 -1.894664

#Incorrect mean spec
aug0<-(1-Z)/(1-ps.fit.c)
aug1<-Z/ps.fit.c
fit.aor.1<-lm(Y~X1+X2+Z+aug0+aug1)
mu.x0<-predict(fit.aor.1,newdata=data.frame(X1,X2,Z=0,aug0=1/(1-ps.fit.c),aug1=0))
mu.x1<-predict(fit.aor.1,newdata=data.frame(X1,X2,Z=1,aug0=0,aug1=1/ps.fit.c))
mu.hat.a0<-mean(mu.x0)
mu.hat.a1<-mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.456348 2.557382 -1.898967

#Incorrect PS spec
aug0<-(1-Z)/(1-ps.fit.i)
aug1<-Z/ps.fit.i
fit.aor.2<-lm(Y~X1+X2+X1:X2+Z+Z:X1+Z:X2+Z:X1:X2+aug0+aug1)
mu.x0<-predict(fit.aor.2,newdata=data.frame(X1,X2,Z=0,aug0=1/(1-ps.fit.i),aug1=0))
mu.x1<-predict(fit.aor.2,newdata=data.frame(X1,X2,Z=1,aug0=0,aug1=1/ps.fit.i))
mu.hat.a0<-mean(mu.x0)
mu.hat.a1<-mean(mu.x1)
c(mu.hat.a0,mu.hat.a1,mu.hat.a1-mu.hat.a0)

+ [1] 4.455919 2.558779 -1.897139
```

One advantage of the AOR approach is that the estimated standard errors can be obtained in a more straightforward fashion using the model-based and sandwich approaches, as the fitting is obtained using a linear model. For example

```
#Correct mean spec
Z0<-rep(0,n);Z1<-Z0+1
Xm0<-cbind(1,X1,X2,Z0,(1-Z0)/(1-ps.fit.c), Z0/ps.fit.c,X1*X2,Z0*X1,Z0*X2,Z0*X1*X2)
Xm1<-cbind(1,X1,X2,Z1,(1-Z1)/(1-ps.fit.c), Z1/ps.fit.c,X1*X2,Z1*X1,Z1*X2,Z1*X1*X2)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.0<-as.numeric(Dm %*% coef(fit.aor.0))
ate.est.0
```

```

+ [1] -1.894664

#Model-based standard error
Vm<-summary(fit.aor.0)$sigma^2*summary(fit.aor.0)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit.aor.0)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.005356486 0.004960722

#Incorrect mean spec
Z0<-rep(0,n);Z1<-Z0+1
Xm0<-cbind(1,X1,X2,Z0,(1-Z0)/(1-ps.fit.c), Z0/ps.fit.c)
Xm1<-cbind(1,X1,X2,Z1,(1-Z1)/(1-ps.fit.c), Z1/ps.fit.c)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.1<-as.numeric(Dm %*% coef(fit.aor.1))
ate.est.1

+ [1] -1.898967

#Model-based standard error
Vm<-summary(fit.aor.1)$sigma^2*summary(fit.aor.1)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit.aor.1)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.02055638 0.02518105

#Incorrect PS spec
Z0<-rep(0,n);Z1<-Z0+1
Xm0<-cbind(1,X1,X2,Z0,(1-Z0)/(1-ps.fit.i), Z0/ps.fit.i,X1*X2,Z0*X1,Z0*X2,Z0*X1*X2)
Xm1<-cbind(1,X1,X2,Z1,(1-Z1)/(1-ps.fit.i), Z1/ps.fit.i,X1*X2,Z1*X1,Z1*X2,Z1*X1*X2)
Dm<-matrix(1/n,nrow=1,ncol=n) %*% (Xm1-Xm0)
ate.est.2<-as.numeric(Dm %*% coef(fit.aor.2))
ate.est.2

+ [1] -1.897139

#Model-based standard error
Vm<-summary(fit.aor.2)$sigma^2*summary(fit.aor.2)$cov.unscaled
se.m<-sqrt(Dm %*% (Vm %*% t(Dm)))

#Sandwich standard error
Vs<-sandwich(fit.aor.2)
se.s<-sqrt(Dm %*% (Vs %*% t(Dm)))

c(se.m[1,1],se.s[1,1])

+ [1] 0.005010136 0.004959904

```

7. G-estimation

7.1 Construction

G-estimation can be considered a modified form of outcome regression estimation that introduces a robustness to mis-specification via PS adjustment. Consider first the data generating model

$$\mathbb{E}_{Y|X,Z}^o[Y|X = x, Z = z] = z\psi_0.$$

Then the OLS estimating equation for the ψ_0 is

$$\sum_{i=1}^n z_i(y_i - z_i\psi_0) = 0.$$

Then we have that the resulting estimate for ψ_0 is

$$\hat{\psi}_0 = \frac{\sum_{i=1}^n z_i y_i}{\sum_{i=1}^n z_i^2}.$$

Suppose now that the estimating equation is modified to

$$\sum_{i=1}^n (z_i - e(x_i))(y_i - z_i\psi_0) = 0.$$

Then we have that the new estimate for ψ_0 is

$$\hat{\psi}_0 = \frac{\sum_{i=1}^n (z_i - e(x_i))y_i}{\sum_{i=1}^n z_i(z_i - e(x_i))}$$

Consider now the more general linear model

$$\mathbb{E}_{Y|X,Z}^o[Y|X = x, Z = z] = \mathbf{x}_1\beta + z\mathbf{x}_2\psi = \mu_0(x; \beta) + z\mu_1(x; \psi)$$

and the OLS estimating equation

$$\sum_{i=1}^n \begin{pmatrix} \mathbf{x}_{1i}^\top \\ z\mathbf{x}_{2i}^\top \end{pmatrix} (y_i - \mathbf{x}_{1i}\beta - z\mathbf{x}_{2i}\psi) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

In modified form, this becomes

$$\sum_{i=1}^n \begin{pmatrix} \mathbf{x}_{1i}^\top \\ (z - e(x_i))\mathbf{x}_{2i}^\top \end{pmatrix} (y_i - \mathbf{x}_{1i}\beta - z\mathbf{x}_{2i}\psi) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

These equations are referred to as the *G-estimating equations*. It can be shown that the estimates for ψ obtained by solving these equations are realizations of unbiased estimators of the true values even if the model

$$\mu_0(x; \beta) = \mathbf{x}_1\beta$$

is mis-specified, **provided that the model**

$$\mu_1(x; \psi) = \mathbf{x}_2\psi$$

is correctly specified, and provided that the PS model is correctly specified. Thus this procedure is a *doubly robust* procedure if $\mu_1(x; \psi)$ is correctly specified.

To solve the estimating equations, we consider estimating the model

$$\mathbb{E}_{Y|X,Z}^o[Y|X = x, Z = z] = \mu_0(x; \beta) + z\mu_1(x; \psi) + e(x)\mu_1(x; \phi)$$

via OLS. For example, suppose we have the linear forms

$$\mu_0(x; \beta) = \beta_0$$

$$\mu_1(x; \psi) = \psi_0 + \psi_1 x_1 + \psi_2 x_2 + \psi_3 x_1 x_2$$

$$\mu_1(x; \phi) = \phi_0 + \phi_1 x_1 + \phi_2 x_2 + \phi_3 x_1 x_2$$

Then the OLS estimating equations take the form

$$\sum_{i=1}^n \begin{pmatrix} \mathbf{x}_{1i}^\top \\ z_i \mathbf{x}_{2i}^\top \\ e(x_i) \mathbf{x}_{2i}^\top \end{pmatrix} (y_i - \beta_0 - z \mathbf{x}_{2i} \psi - e(x) \mathbf{x}_{2i} \phi) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

and subtracting the third block from the second block yields

$$\sum_{i=1}^n \begin{pmatrix} \mathbf{x}_{1i}^\top \\ (z_i - e(x_i)) \mathbf{x}_{2i}^\top \\ e(x_i) \mathbf{x}_{2i}^\top \end{pmatrix} (y_i - \beta_0 - z \mathbf{x}_{2i} \psi - e(x) \mathbf{x}_{2i} \phi) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

which recovers the G-estimating equations. Hence, **G-estimation is a form of propensity score regression.**

```
fit.gee<-lm(Y~Z+Z:X1+Z:X2+Z:X1:X2+ps.fit.c+ps.fit.c:X1+ps.fit.c:X2+ps.fit.c:X1:X2)
round(coef(summary(fit.gee)),6)

+           Estimate Std. Error   t value Pr(>|t|)
+ (Intercept)   3.293699   0.022397 147.058794 0.00000
+ Z              0.845876   0.127432   6.637850 0.00000
+ ps.fit.c       6.229603   0.155483  40.066130 0.00000
+ Z:X1           1.073735   0.094502  11.362004 0.00000
+ Z:X2           0.962851   0.045039  21.378190 0.00000
+ X1:ps.fit.c    -2.126455   0.109876 -19.353144 0.00000
+ X2:ps.fit.c     1.123178   0.054093  20.763898 0.00000
+ Z:X1:X2        0.993968   0.039895  24.914739 0.00000
+ X1:X2:ps.fit.c 0.047344   0.047179   1.003503 0.31574

mu0.gee<-mean(predict(fit.gee,newdata=data.frame(X1,X2,Z=0,ps.fit.c)))
mu1.gee<-mean(predict(fit.gee,newdata=data.frame(X1,X2,Z=1,ps.fit.c)))
c(mu0.gee,mu1.gee,mu1.gee-mu0.gee)

+ [1]  4.455743  2.559727 -1.896017
```

7.2 The drgee package

The doubly robust version of G-estimation is carried out by the drgee package in R. This package considers the model specification

$$\mathbb{E}_{Y|X,Z}^\circ[Y|X = x, Z = z] = \mu_0(x; \beta) + z\mu_1(x; \psi)$$

for the outcome regression component, and

$$\mathbb{E}_{Z|X}^\circ[Z|X = x] = g(x; \alpha)$$

for the PS component, where $g(\cdot)$ is an appropriate link function. In the drgee function

- $\mu_0(x; \beta)$ is specified via the oformula and olink arguments;
- $\mu_1(x; \beta)$ is specified via the iaformula and olink arguments;
- $g(x; \alpha)$ is specified via the eformula and elink arguments.

The argument estimation.method allows for estimation via three different methods:

- estimation.method = 'dr' performs doubly robust G-estimation;

- `estimation.method = 'e'` performs singly robust G-estimation, that is, it uses the model with

$$\mu_0(x) \equiv 0;$$

- `estimation.method = 'o'` performs outcome regression using the model

$$\mu_0(x; \beta) + z\mu_1(x; \psi).$$

```
mydata<-data.frame(X1,X2,Z,Y)
library(drgee)

#Correct specification
dr.est <- drgee(oformula = formula(Y~X1*X2), eformula = formula(Z~X1*X2),
iaformula = formula(~X1*X2), olink = "identity", elink = "logit",
data = mydata, estimation.method = "dr")
round(coef(summary(dr.est)),6)

+           Estimate Std. Error   z value Pr(>|z|)
+ Z           0.932592   0.114172   8.168324      0
+ Z:X1        1.022372   0.083075  12.306551      0
+ Z:X2        0.990733   0.039580  25.031183      0
+ Z:X1:X2     0.983728   0.034152  28.804130      0

#Mis-specification of PS model, correct specification of the OR nuisance model
dr.est <- drgee(oformula = formula(Y~X1*X2), eformula = formula(Z~X1),
iaformula = formula(~X1*X2), olink = "identity", elink = "logit",
data = mydata, estimation.method = "dr")
round(coef(summary(dr.est)),6)

+           Estimate Std. Error   z value Pr(>|z|)
+ Z           0.954339   0.109542   8.712102      0
+ Z:X1        1.006508   0.080508  12.501997      0
+ Z:X2        0.996251   0.038154  26.111106      0
+ Z:X1:X2     0.981085   0.032338  30.338897      0

#Correct specification of PS model, mis-specification of the OR nuisance model
dr.est <- drgee(oformula = formula(Y~X1), eformula = formula(Z~X1*X2),
iaformula = formula(~X1*X2), olink = "identity", elink = "logit",
data = mydata, estimation.method = "dr")
round(coef(summary(dr.est)),6)

+           Estimate Std. Error   z value Pr(>|z|)
+ Z           0.473928   0.519484   0.912305  0.361608
+ Z:X1        1.581006   0.330642   4.781626  0.000002
+ Z:X2        0.794845   0.163695   4.855637  0.000001
+ Z:X1:X2     1.239469   0.129291   9.586683  0.000000
```

To match the estimation in the earlier section, we note that `drgee` utilizes an alternative formulation and an additional estimating equation. Using bold symbols to denote vectors and matrices, in the linear case `drgee` proceeds by solving the estimating equations

$$\begin{pmatrix} \mathbf{X}_\beta^\top (\mathbf{y} - \mathbf{X}_\beta \beta - \mathbf{z} \odot \mathbf{X}_\psi \psi') \\ \mathbf{X}_\psi^\top \mathbf{z} \odot (\mathbf{y} - \mathbf{X}_\beta \beta - \mathbf{z} \odot \mathbf{X}_\psi \psi') \\ \mathbf{X}_\psi^\top (\mathbf{z} - \mathbf{e}) \odot (\mathbf{y} - \mathbf{X}_\beta \beta - \mathbf{z} \odot \mathbf{X}_\psi \psi) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

where ψ is the parameter of interest, and where the symbol $\odot$ denotes *elementwise* multiplication, that is for two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^k$,

$$\mathbf{x}_1 \odot \mathbf{x}_2 = [x_{1j}x_{2j}]_{j=1}^k \in \mathbb{R}^k.$$

`drgee` solves the first two equations to get $\hat{\beta}$ utilizing the additional nuisance parameter ψ' , then solves

$$\mathbf{X}_\psi^\top (\mathbf{z} - \mathbf{e}) \odot (\mathbf{y} - \mathbf{X}_\beta \hat{\beta} - \mathbf{z} \odot \mathbf{X}_\psi \psi) = 0$$

that is, yielding the solution

$$\hat{\psi} = (\mathbf{X}_{\psi}^{\top} \mathbf{D} \mathbf{X}_{\psi})^{-1} \mathbf{X}_{\psi}^{\top} (\mathbf{z} - \mathbf{e}) \odot (\mathbf{y} - \mathbf{X}_{\beta} \hat{\beta})$$

where $\mathbf{D}$ is the $n \times n$ diagonal matrix

$$\mathbf{D} = \text{diag} \{ (\mathbf{z} - \mathbf{e}) \odot \mathbf{z} \}.$$

In contrast, the ordinary G-estimation method uses the reduced system

$$\begin{pmatrix} \mathbf{X}_{\beta}^{\top} (\mathbf{y} - \mathbf{X}_{\beta} \beta - \mathbf{z} \odot \mathbf{X}_{\psi} \psi) \\ \mathbf{X}_{\psi}^{\top} (\mathbf{z} - \mathbf{e}) \odot (\mathbf{y} - \mathbf{X}_{\beta} \beta - \mathbf{z} \odot \mathbf{X}_{\psi} \psi) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

which does not have the extra nuisance parameters ψ' . The estimator $\hat{\psi}$ takes precisely the same form except that the plug-in form of $\hat{\beta}$ is different. Under correct specification of the blip (always assumed in our analysis), the two methods are essentially equivalent.

```
#Correct specification
dr.est <- drgee(oformula = formula(Y~1), eformula = formula(Z~X1*X2),
iaformula = formula(~X1*X2), olink = "identity", elink = "logit",
data = mydata, estimation.method = "dr")
pars<-dr.est$coefficients.all
psi.hat<-pars[1:4] #psi parameters
alpha.hat<-pars[5:8] #ps parameters
psi.prime.hat<-pars[9:12] #psi additional parameters
beta.hat<-pars[13] #beta.parameters
psi.hat

+          Z          Z:X1          Z:X2          Z:X1:X2
+ 0.4686013 1.5419502 0.8141539 1.1963121

a<-matrix(c(1,mean(X1),mean(X2),mean(X1*X2)),nrow=1)
ate.gest<-a %*% psi.hat
ate.gest

+          [,1]
+ [1,] -1.889696

ate.gest.var<-a %*% (dr.est$vcov.all[1:4,1:4] %*% t(a))
se.s<-sqrt(ate.gest.var[1,1])
se.s

+ [1] 0.007467362
```

From this analysis under correct specification, we see that the ATE estimate is -1.8896963, and the sandwich standard error is 0.0074674. Under mis-specification of the propensity model, we obtain the following results:

```
dr.est <- drgee(oformula = formula(Y~X1*X2), eformula = formula(Z~X1),
iaformula = formula(~X1*X2), olink = "identity", elink = "logit",
data = mydata, estimation.method = "dr")

pars<-dr.est$coefficients.all
psi.hat<-pars[1:4] #psi parameters
alpha.hat<-pars[5:8] #ps parameters
psi.prime.hat<-pars[9:12] #psi additional parameters
beta.hat<-pars[13] #beta.parameters
psi.hat

+          Z          Z:X1          Z:X2          Z:X1:X2
+ 0.9543391 1.0065084 0.9962512 0.9810853

a<-matrix(c(1,mean(X1),mean(X2),mean(X1*X2)),nrow=1)
ate.gest<-a %*% psi.hat
ate.gest
```

```

+           [,1]
+ [1,] -1.8974

ate.gest.var<-a %*% (dr.est$vcov.all[1:4,1:4] %*% t(a))
se.s<-sqrt(ate.gest.var[1,1])
se.s

+ [1] 0.00497718

```

We see that the ATE estimate is -1.8973999, and the sandwich standard error is 0.0049772. Finally, under misspecification of the nuisance mean model, we obtain the following results:

```

dr.est <- drgee(oformula = formula(Y~X1), eformula = formula(Z~X1*X2),
iaformula = formula(~X1*X2), olink = "identity", elink = "logit",
data = mydata, estimation.method = "dr")

pars<-dr.est$coefficients.all
psi.hat<-pars[1:4] #psi parameters
alpha.hat<-pars[5:8] #ps parameters
psi.prime.hat<-pars[9:12] #psi additional parameters
beta.hat<-pars[13] #beta.parameters
psi.hat

+           Z           Z:X1           Z:X2           Z:X1:X2
+ 0.473928 1.581006 0.794845 1.239469

a<-matrix(c(1,mean(X1),mean(X2),mean(X1*X2)),nrow=1)
ate.gest<-a %*% psi.hat
ate.gest

+           [,1]
+ [1,] -1.888149

ate.gest.var<-a %*% (dr.est$vcov.all[1:4,1:4] %*% t(a))
se.s<-sqrt(ate.gest.var[1,1])
se.s

+ [1] 0.007663223

```

We see that the ATE estimate is -1.8881495, and the sandwich standard error is 0.0076632.

For a calculation from first principles, using the estimating equations given above.

```

#Computing the ATE
Xm<-cbind(1,X1,X2,X1*X2)
ps.fit.drgee<-1/(1+exp(-Xm %*% alpha.hat))
mu0.drgee<-mean(beta.hat + (0-ps.fit.drgee)*Xm%*%psi.hat+ps.fit.drgee*Xm%*%psi.prime.hat)
mu1.drgee<-mean(beta.hat + (1-ps.fit.drgee)*Xm%*%psi.hat+ps.fit.drgee*Xm%*%psi.prime.hat)
c(mu0.drgee,mu1.drgee,mu1.drgee-mu0.drgee)

+ [1] 4.728757 2.840607 -1.888149

```

```

#Now to check the calculation using the theory above
psi.prime.hat

+           Z           Z:X1           Z:X2           Z:X1:X2
+ 6.3950043 -0.7085561 2.1928755 1.5931307

beta.hat

+ (Intercept)
+ 4.571103

```

```

coef(fit0<-lm(Y~Z+Z:X1+Z:X2+Z:X1:X2))

+ (Intercept)          Z          Z:X1          Z:X2          Z:X1:X2
+  4.2904166    6.6756906  -0.9925529    2.1928755    1.5931307

alpha.hat

+ (Intercept)          X1          X2          X1:X2
+  7.2441995   -0.8720527    0.9531355    2.0153501

coef(fit1<-glm(Z~X1*X2,family=binomial))

+ (Intercept)          X1          X2          X1:X2
+  7.2441995   -0.8720527    0.9531355    2.0153501

eX<-fitted(fit1)
Y1<-(Z-eX)*(Y-beta.hat)
Xp<-cbind(1,X1,X2,X1*X2)
D<-diag(Z*(Z-eX))
psi.ests<-solve((t(Xp) %*% D) %*% Xp) %*% t(Xp) %*% Y1
cbind(psi.hat,psi.ests)  #Results match

+          psi.hat
+ Z          0.473928 0.4686013
+ Z:X1        1.581006 1.5419502
+ Z:X2        0.794845 0.8141539
+ Z:X1:X2     1.239469 1.1963121

```