CAUSAL ADJUSTMENT FOR THE LOG-LINEAR MODEL USING REGRESSION

In the following,

- X is the random vector of confounders, and x its observed values,
- $\mathbf{x}$ is the vector of functions of x that form the linear predictor,
- $\mathbf{X}$ is the random variable version of $\mathbf{x}$.

Suppose that a *log-linear* model is deemed appropriate:

$$\log \mu(x, z; \beta, \psi) = \mathbf{x}_\beta \beta + z \mathbf{x}_\psi \psi$$

Regarding this as the structural model, we then have that

$$\mathbb{E}[Y(\mathsf{z})] \equiv \mathbb{E}^{\mathcal{E}}_{Y|Z}[Y|Z = \mathsf{z}] = \mathbb{E}^{\mathcal{E}}_{X}[\exp\{\mathbf{X}_\beta \beta + \mathsf{z}\mathbf{X}_\psi \psi\}]$$

which would then invoke the estimator

$$\widetilde{\mu}(\mathsf{z}) = \frac{1}{n} \sum_{i=1}^{n} \exp\{\mathbf{x}_{\beta i} \widehat{\beta} + \mathsf{z}\mathbf{x}_{\psi i} \widehat{\psi}\}.$$

where $(\widehat{\beta}, \widehat{\psi})$ are estimated from a correctly specified model. The above model assumes that

$$\mu(x, 1; \beta, \psi) = \mu(x, 0; \beta, \psi) \exp\{\mathbf{x}_\psi \psi\}.$$

with the effect of Z represented *conditional* on X; *marginally*, ψ alone does not capture the effect of treatment. However, if we can estimate β and ψ consistently, then we can recover APO and ATE estimators from them. By standard regression arguments, we know that *correct specification* of the outcome model is needed.

```
set.seed(1293)
library(mvnfast)
expit<-function(x){return(1/(1+exp(-x)))}
muX<-c(-1,0.5,1)
Sig<-0.1*0.9^abs(outer(1:3,1:3,'-'))
al<-c(2,1,-2,0.75)
N<-1000000
X<-rmvn(N,muX,Sig)
Xa<-cbind(1,X[,1],X[,2],X[,1]*X[,2])
eX<-expit(Xa %*% al)
Z<-rbinom(N,1,eX)
Xb<-cbind(1,X[,1],X[,2],Z)
X1<-X[,1];X2<-X[,2]
be<-c(3,2,-2); psi<-1
muY<-exp(Xb %*% c(be,psi))
Y<-rpois(N,muY)
mu0<-exp(cbind(1,X[,1],X[,2],0) %*% c(be,psi))
mu1<-exp(cbind(1,X[,1],X[,2],1) %*% c(be,psi))
#Estimated effects (via a large sample Monte Carlo estimate)
true.ate1<-mean(mu1)-mean(mu0)   #Additive ATE
true.ate2<-mean(mu1)/mean(mu0)   #Multiplicative ATE
c(true.ate1,true.ate2)

+ [1] 1.788359 2.718282
```

```
fit0<-glm(Y~X1+X2+Z,family=poisson)
fit.mu0<-predict(fit0,newdata=data.frame(X1=X1,X2=X2,Z=0),type='response')
fit.mu1<-predict(fit0,newdata=data.frame(X1=X1,X2=X2,Z=1),type='response')
ate1.est<-mean(fit.mu1)-mean(fit.mu0)
ate2.est<-mean(fit.mu1)/mean(fit.mu0)
c(ate1.est,ate2.est)

+ [1] 1.791005 2.720701
```

```r
#Mis-specified model
fit1<-glm(Y~X1+Z,family=poisson)
fit.mu0<-predict(fit1,newdata=data.frame(X1=X1,Z=0),type='response')
fit.mu1<-predict(fit1,newdata=data.frame(X1=X1,Z=1),type='response')
ate1.est<-mean(fit.mu1)-mean(fit.mu0)
ate2.est<-mean(fit.mu1)/mean(fit.mu0)
c(ate1.est,ate2.est)

+ [1] 1.997818 3.003571
```

For the log-linear model,

$$\log \mu(x, z; \beta, \psi) = \mathbf{x}_\beta \beta + z \mathbf{x}_\psi \psi$$

the standard estimating equation takes the form

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

We might try to make inference robust to mis-specification using a *G-estimation*-like strategy, and modify the estimating equation to be

$$\sum_{i=1}^{n} \begin{pmatrix} \mathbf{x}_{\beta i}^\top \\ (z_i - e(x_i))\mathbf{x}_{\psi_i}^\top \end{pmatrix} (y_i - \exp\{\mathbf{x}_{\beta i}\beta + z_i \mathbf{x}_{\psi i}\psi\}) = \mathbf{0}$$

where $e(x_i)$ is the propensity score. In the *log-linear* case, we must modify the second estimating function to be

$$\varphi(X, Y, Z) = (Z - e(X))\mathbf{X}_\psi^\top \exp\{-Z\mathbf{X}_\psi\psi\}(Y - \exp\{\mathbf{X}_\beta\beta + Z\mathbf{X}_\psi\psi\})$$

Thus for consistent estimation, we need to solve

$$\sum_{i=1}^{n} \begin{pmatrix} \mathbf{x}_{\beta i}^\top \\ (z_i - e(x_i))\mathbf{x}_{\psi_i}^\top \exp\{-z_i \mathbf{x}_{\psi i}\psi\} \end{pmatrix} (y_i - \exp\{\mathbf{x}_{\beta i}\beta + z_i \mathbf{x}_{\psi i}\psi\}) = \mathbf{0}. \tag{1}$$

In the following simulation, we estimate ψ where in the data generating model we have

$$\log \mu(x, z; \beta, \psi) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + z\psi$$

for $Y \sim Poisson(\mu(x, z; \beta, \psi))$, and

$$\text{logit}(e(x)) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1 x_2$$

using five analyses:

1. Correctly specified outcome regression;
2. Incorrectly specified outcome regression with model
$$\log \mu(x, z; \beta, \psi) = \beta_0 + \beta_1 x_1 + z\psi;$$
3. G-estimation with correctly specified outcome regression with from 1. and correctly specified propensity score model;
4. G-estimation with incorrectly specified outcome regression with from 2. and correctly specified propensity score model;
5. G-estimation using the drgee package.

G-estimation using equation (1) can be carried out using the package `nleqslv` in `R`, which is a generic non-linear equation solver.

```
library(nleqslv)
library(drgee)
loglin<-function(xv,Xma,Xmb,zv,yv){

    nv<-nrow(Xma)
    alv<-xv[1:ncol(Xma)]
    bev<-xv[(ncol(Xma)+1):(ncol(Xma)+ncol(Xmb))]

    eXv<-expit(Xma %*% alv)
    muv<-exp(Xmb %*% bev)

    Xbd<-Xmb
    Xbd[,ncol(Xmb)]<-(zv-eXv)*exp(-zv*bev[length(bev)])

    r1<-t(Xbd) %*% (yv - muv)
    r2<-t(Xma) %*% (zv - eXv)
    return(c(r1,r2))
}

nreps<-1000
n<-1000
psi.ests<-matrix(0,nrow=nreps,ncol=5)
for(irep in 1:nreps){
    X<-rmvn(n,muX,Sig)
    Xa<-cbind(1,X[,1],X[,2],X[,1]*X[,2])
    eX<-expit(Xa %*% al)
    Z<-rbinom(n,1,eX)
    Xb<-cbind(1,X[,1],X[,2],Z)
    X1<-X[,1];X2<-X[,2]
    muY<-exp(Xb %*% c(be,psi))
    Y<-rpois(n,muY)

    #1. Correctly specified
    fit1<-glm(Y~X1+X2+Z+X1:X2,family=poisson)
    psi.ests[irep,1]<-coef(fit1)[4]

    #2. Mis-specified
    fit1<-glm(Y~X1+Z,family=poisson)
    psi.ests[irep,2]<-coef(fit1)[3]

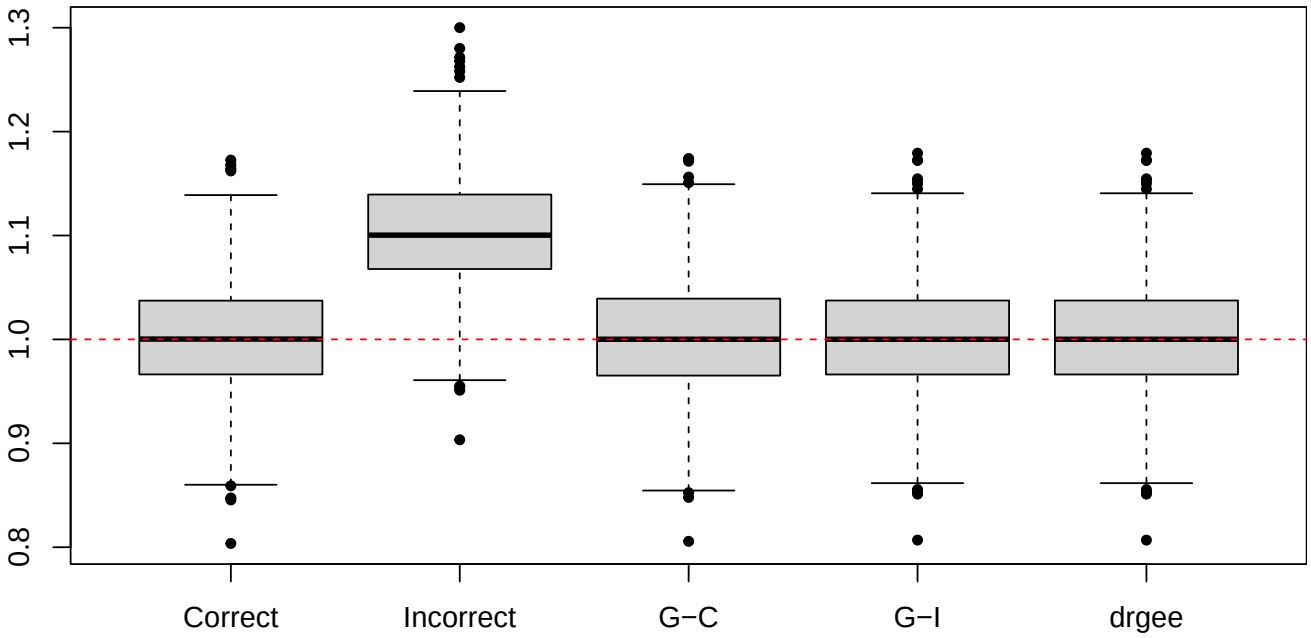
    #3. G-estimation - Correct specification
    xs<-rep(0,length(al)+length(be)+1)
    fit.ll<-nleqslv(xs,fn=loglin,Xma=Xa,Xmb=Xb,zv=Z,yv=Y,control=list(ftol=1e-10,maxit=500))
    psi.ests[irep,3]<-fit.ll$x[length(fit.ll$x)]

    #4. G-estimation - incorrect specification
    Xa0<-Xa
    Xb0<-cbind(1,X1,Z)
    xs<-rep(0,ncol(Xa0)+ncol(Xb0))
    fit.ll<-nleqslv(xs,fn=loglin,Xma=Xa0,Xmb=Xb0,zv=Z,yv=Y,control=list(ftol=1e-10,maxit=500))
    psi.ests[irep,4]<-fit.ll$x[length(fit.ll$x)]

    #drgee
    dr.est <- drgee(oformula = formula(Y~X1), eformula = formula(Z~X1*X2),
    iaformula = formula(~1), olink = "log", elink = "logit", estimation.method = "dr")
    psi.ests[irep,5]<-coef(dr.est)[1]
}
```

```
par(mar=c(2,2,2,0))
nv<-c('Correct','Incorrect','G-C','G-I','drgee')
```

```
boxplot(psi.ests,names=nv,pch=19,cex=0.7);abline(h=psi,col='red',lty=2)
```




```
bias.est<-apply(psi.ests-psi,2,mean)*sqrt(n)
var.est<-apply(psi.ests,2,var)*n
var.est                                  #Estimator variance

+ [1] 2.821650 3.018495 2.872015 2.873106 2.873094

mse.est<-bias.est^2+var.est
mse.est                                  #Estimator MSE

+ [1]  2.823683 13.359852  2.873547  2.874847  2.874836
```

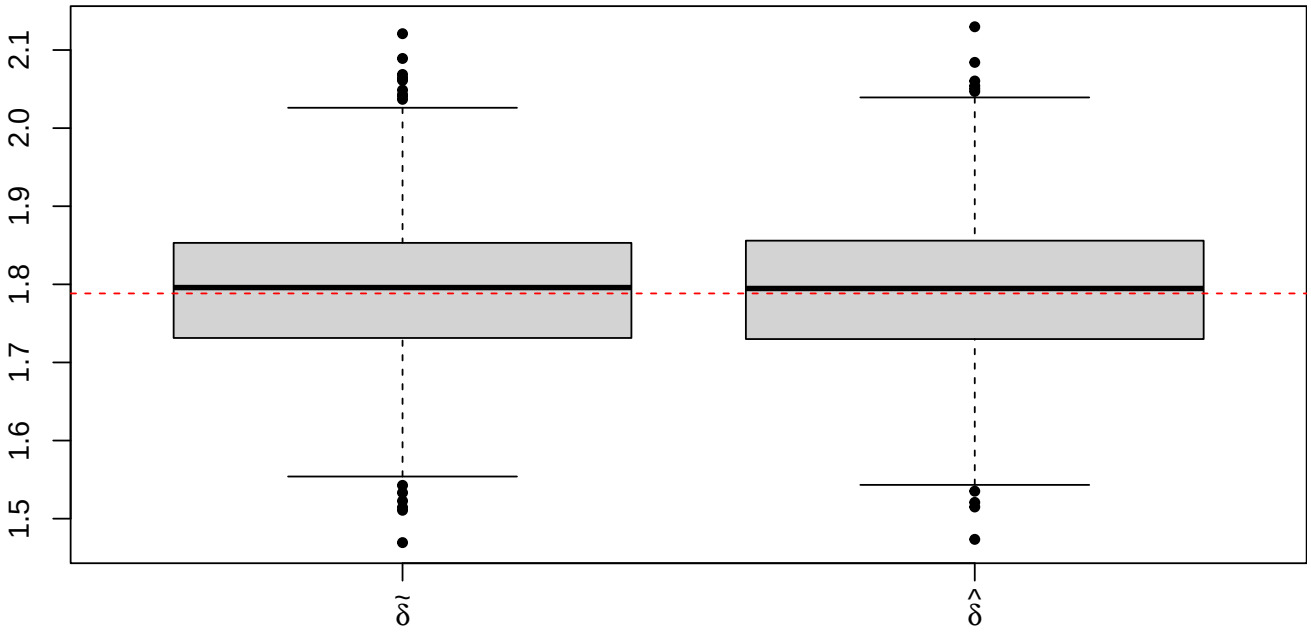
In this example, the G-estimation methods provide unbiased estimation of the ψ parameter, with estimator variance that is not inferior to that of the correctly specified outcome regression.

Note: Although we can recover unbiased estimates of ψ in this analysis, we cannot in general recover estimates of the ATE using G-estimation if the outcome mean model is mis-specified, as it is not possible to recover consistent estimates of β. We can achieve consistent estimation of the ATE using inverse probability weighting.

```
ipw.ests<-matrix(0,nrow=nreps,ncol=2)
for(irep in 1:nreps){
    X<-rmvn(n,muX,Sig)
    Xa<-cbind(1,X[,1],X[,2],X[,1]*X[,2])
    eX<-expit(Xa %*% al)
    Z<-rbinom(n,1,eX)
    Xb<-cbind(1,X[,1],X[,2],Z)
    X1<-X[,1];X2<-X[,2]
    muY<-exp(Xb %*% c(be,psi))
    Y<-rpois(n,muY)

    eX<-fitted(glm(Z~X1*X2,family=binomial))
    w0<-(1-Z)/(1-eX); w1<-Z/eX
    W0<-w0/sum(w0); W1<-w1/sum(w1)
    ipw.ests[irep,1]<-mean(w1*Y)-mean(w0*Y)
    ipw.ests[irep,2]<-sum(W1*Y)-sum(W0*Y)
}
```

```
par(mar=c(2,2,2,0))
nv<-c(expression(tilde(delta)),expression(hat(delta)))
boxplot(ipw.ests,names=nv,pch=19,cex=0.7);abline(h=true.ate1,col='red',lty=2)
```




```
bias.est<-apply(ipw.ests-true.ate1,2,mean)*sqrt(n)
var.est<-apply(ipw.ests,2,var)*n
var.est                                  #Estimator variance

+ [1] 9.214674 9.033296

mse.est<-bias.est^2+var.est
mse.est                                  #Estimator MSE

+ [1] 9.238488 9.050317
```

We can compute sandwich estimates of the asymptotic variance using the methods described previously.