INVERSE PROBABILITY WEIGHTING WITH A CONTINUOUS EXPOSURE

The standard approach to inverse probability weighting (IPW) estimation for a continuous treatment is based on the same kind of inverse weight as in the binary treatment case, specifically

$$w(x,z) = \frac{1}{f_{Z|X}(z|x)}$$

to define IPW estimators

$$\widetilde{\mu}_{\text{IPW}}(z) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{1}_{\{z\}}(Z_i)Y_i}{f^{\mathcal{O}}_{Z|X}(Z_i|X_i)} \qquad \text{and} \qquad \widehat{\mu}_{\text{IPW}}(z) = \sum_{i=1}^{n} W_{iz}Y_i$$

with

$$W_{iz} = \frac{\dfrac{\mathbb{1}_{\{z\}}(Z_i)}{f^{\mathcal{O}}_{Z|X}(Z_i|X_i)}}{\displaystyle\sum_{j=1}^{n} \frac{\mathbb{1}_{z}(Z_j)}{f^{\mathcal{O}}_{Z|X}(Z_j|X_j)}}$$

The challenge with implementing this estimator is that the indicator function $\mathbb{1}_{\{z\}}(z_i)$ only picks off the observed treatments z_i that **exactly** match the target level z. This can be overcome by taking a small window around the target level, $(z-d, z+d)$ say. The weight is thus modified to be

$$w(x,z) = \frac{\mathbb{1}_{(z-d,z+d)}(Z_i)}{f_{Z|X}(z|x)}$$

which requires a modification to the $\widetilde{\mu}_{\text{IPW}}(z)$ estimator, namely

$$\widetilde{\mu}_{\text{IPW}}(z) = \frac{1}{2dn}\sum_{i=1}^{n} \frac{\mathbb{1}_{(z-d,z+d)}(Z_i)Y_i}{f^{\mathcal{O}}_{Z|X}(Z_i|X_i)}$$

EXAMPLE: In this example, we have a data generating process with $k = 10$ predictors, joint Normally distributed $X \sim Normal(\boldsymbol{\mu}, \Sigma)$, with treatment model given by

$$Z|X = x \sim Normal(\mathbf{x}_\alpha \alpha, \sigma_Z^2)$$

and outcome model

$$Y|X = x, Z = z \sim Normal(\mu(x,z), \sigma_Y^2)$$

with

$$\mu(x,z) = \mathbf{x}_0\beta + \psi_0 z = \beta_0 + \sum_{l=1}^{k} x_l\beta_l + \psi_0 z.$$

In the model, all predictors are predictors of Z, but only first three components are confounders. In the analysis

$$\alpha = (4,1,1,1,1,1,1,-2,-2,2)^\top \qquad \beta = (2,-2,2.2,3.6,0,0,0,0,0,0)^\top$$

and $\psi_0 = 1$.

```
#Set-up
set.seed(865177)
library(mvnfast)
n<-10000
k<-10
Mu<-sample(-5:5,size=k,rep=T)/2
Sigma<-0.1*0.8^abs(outer(1:k,1:k,'-'))
al<-c(4,rep(1,6),rep(-2,4))
be<-rep(0,k+2)
be[1:5]<-c(2,1,-2.0,2.2,3.6)
```

```
sigZ<-5;sigY<-1

#Exploratory sample
X<-rmvn(n,mu=Mu,Sigma)
Xal<-cbind(1,X)
ps.true<-Xal %*% al
Z<-rnorm(n,ps.true,sigZ)

#Find a plotting grid
Z0<-round(mean(Z),0)
zgrid<-seq(Z0-5,Z0+5,by=0.1)
nZ<-length(zgrid)
#Intercept
intercept.val<-be[1]+sum(Mu*be[-c(1:2)])
Xm<-cbind(1,Z,X)
muval<-as.vector(Xm %*% be)
Y<-rnorm(n,muval,sigY)
```

The quantity ψ_0 measures the unconfounded effect of treatment, as, marginally,

$$\mathbb{E}[Y(z)] = \psi_0 z + \mu_0 \beta.$$

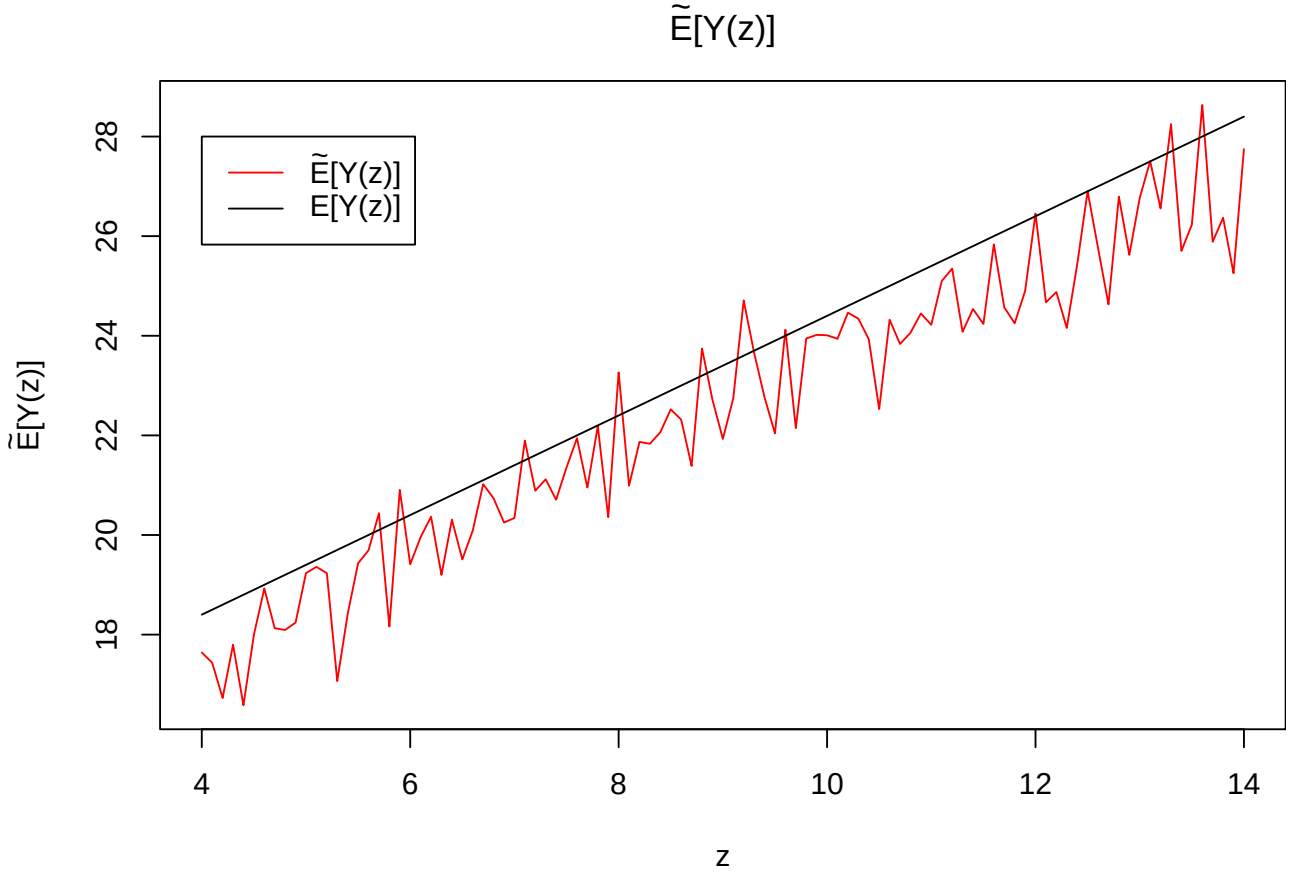
where $\mu_0 = (1, \boldsymbol{\mu}^\top)$ so that the intercept is $\mu_0\beta = 14.4$. Here, $\sigma_Z = 5$ and $\sigma_Y = 1$. We carry out a simulation study of 200 replicates with $n = 1000$.

```
#Monte Carlo study
n<-1000
nreps<-200
d<-0.05
zhat.samp<-ztilde.samp<-matrix(0,nrow=nreps,ncol=nZ)
for(irep in 1:nreps){
    X<-rmvn(n,mu=Mu,Sigma)
    Xal<-cbind(1,X)
    ps.true<-Xal %*% al
    Z<-rnorm(n,ps.true,sigZ)
    Xm<-cbind(1,Z,X)
    muval<-as.vector(Xm %*% be)
    Y<-rnorm(n,muval,sigY)

    fden<-lm(Z~X)
    den<-dnorm(Z,predict(fden),sd(fden$residuals))

    zhat<-ztilde<-rep(0,nZ)
    for(ix in 1:nZ){
        zl<-zgrid[ix]-d
        zu<-zgrid[ix]+d
        w<-(Z > zl & Z < zu)/den
        zhat[ix]<-sum(w*Y,na.rm=T)/sum(w,na.rm=T)
        ztilde[ix]<-mean(w*Y,na.rm=T)/(2*d)
    }
    zhat.samp[irep,]<-zhat
    ztilde.samp[irep,]<-ztilde
}
```

First we look at the performance of $\widetilde{\mu}_{\text{IPW}}(z)$:

```
par(mar=c(4,5,3,0))
ztilde.summary<-apply(ztilde.samp,2,quantile,prob=c(0.025,0.50,0.975),na.rm=TRUE)
plot(zgrid,ztilde.summary[2,],type='l',xlab='z',
        ylab=expression(paste(tilde(E),"[Y(z)]")),col='red')
lines(zgrid,ztilde.summary[1,],lty=2,col='red')
```

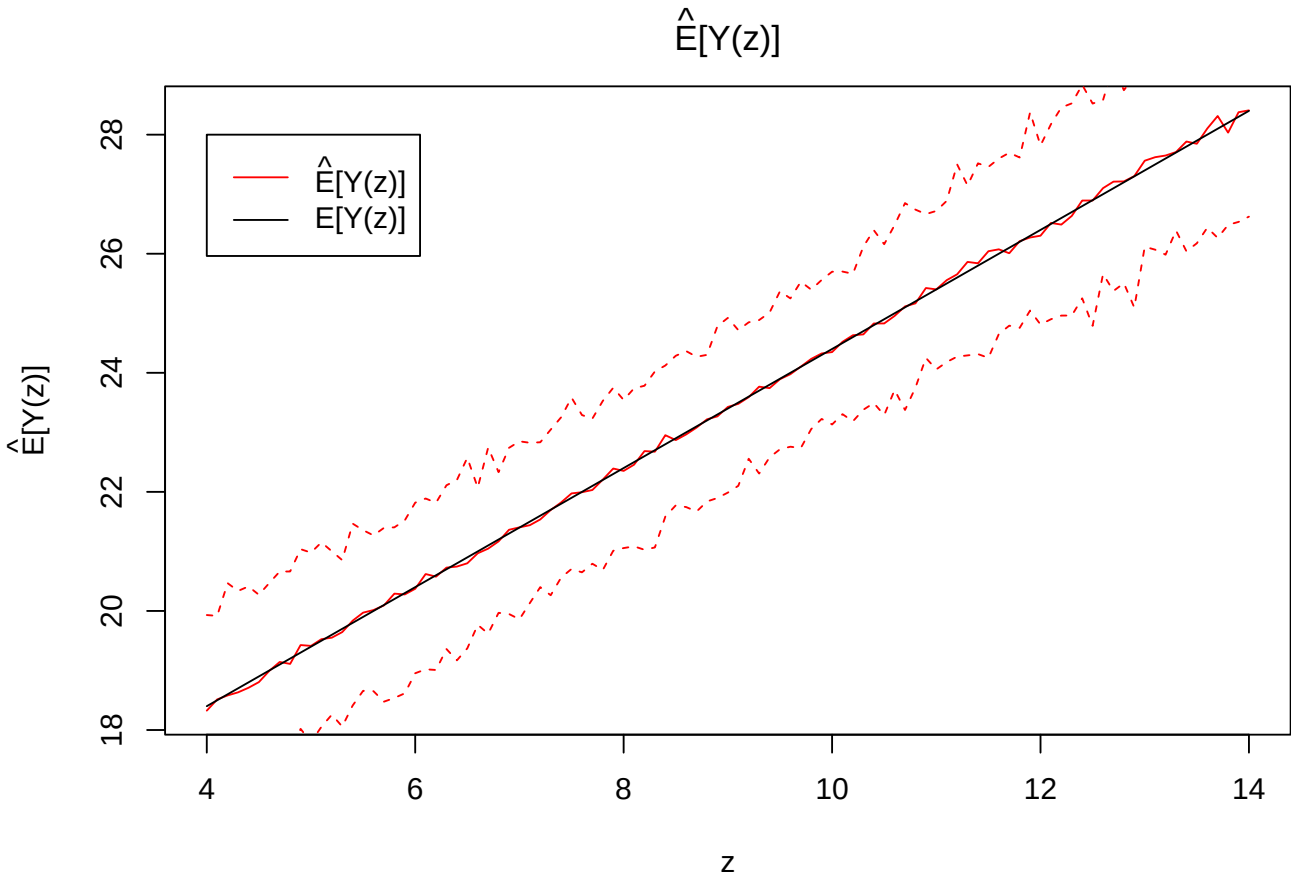
```
lines(zgrid,ztilde.summary[3,],lty=2,col='red')
lines(zgrid,intercept.val+be[2]*zgrid)
legend(zgrid[1],28,c(expression(paste(tilde(E),"[Y(z)]")),expression(paste(E,"[Y(z)]"))),
       col=c('red','black'),lty=1)
title(expression(paste(tilde(E),"[Y(z)]")))
```




```
fit.true<-lm(ztilde.summary[2,]~zgrid)
coef(summary(fit.true))

+               Estimate Std. Error  t value     Pr(>|t|)
+ (Intercept) 13.9382654 0.26483201 52.63059 3.442976e-74
+ zgrid        0.9530051 0.02799361 34.04366 1.865120e-56
```

The estimator is quite noisy; in the plot, the Monte Carlo confidence intervals are very wide. Next we look at the performance of $\widehat{\mu}_{\mathrm{IPW}}(z)$:

```
par(mar=c(4,5,3,0))
zhat.summary<-apply(zhat.samp,2,quantile,prob=c(0.025,0.50,0.975),na.rm=TRUE)
plot(zgrid,zhat.summary[2,],type='l',xlab='z',
       ylab=expression(paste(hat(E),"[Y(z)]")),col='red')
lines(zgrid,zhat.summary[1,],lty=2,col='red')
lines(zgrid,zhat.summary[3,],lty=2,col='red')
legend(zgrid[1],28,c(expression(paste(hat(E),"[Y(z)]")),expression(paste(E,"[Y(z)]"))),
       col=c('red','black'),lty=1)
title(expression(paste(hat(E),"[Y(z)]")))
lines(zgrid,intercept.val+be[2]*zgrid)
```




```
fit.true<-lm(zhat.summary[2,]~zgrid)
coef(summary(fit.true))

+              Estimate  Std. Error  t value      Pr(>|t|)
+ (Intercept) 14.354911 0.021889016 655.8043 6.628522e-182
+ zgrid        1.005962 0.002313741 434.7773 3.071856e-164
```

The performance is much better when the weights are standardized, and the confidence intervals (indicated by dotted lines) are much shorter.

Stabilized Weighting: It is also sometimes recommended to use the 'stabilized' weight based on the (estimated) marginal distribution $f_Z(z)$, that is

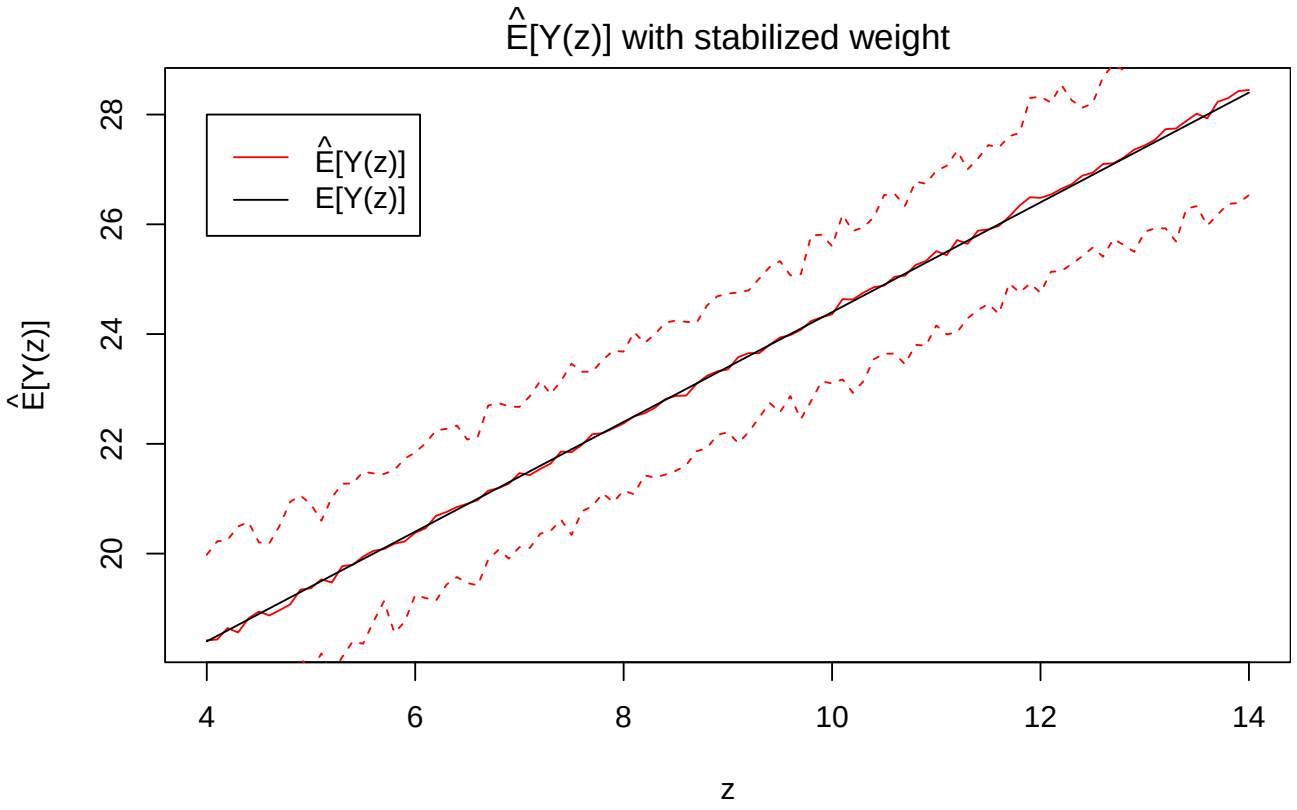
$$w(x,z) = \frac{f_Z(z)}{f_{Z|X}(z|x)}.$$

```
#Monte Carlo study
n<-1000
nreps<-200
d<-0.05
zstab.samp<-matrix(0,nrow=nreps,ncol=nZ)
for(irep in 1:nreps){
    X<-rmvn(n,mu=Mu,Sigma)
    Xal<-cbind(1,X)
    ps.true<-Xal %*% al
    Z<-rnorm(n,ps.true,sigZ)
    Xm<-cbind(1,Z,X)
    muval<-as.vector(Xm %*% be)
    Y<-rnorm(n,muval,sigY)
```

```
    fnum<-lm(Z~1)
    fden<-lm(Z~X)
    zstab<-rep(0,nZ)
    ratio<-dnorm(Z,predict(fnum),sd(fnum$residuals))/dnorm(Z,predict(fden),sd(fden$residuals))
    for(ix in 1:nZ){
        zl<-zgrid[ix]-d
        zu<-zgrid[ix]+d
        w<-(Z > zl & Z < zu)*ratio
        zstab[ix]<-sum(w*Y,na.rm=T)/sum(w,na.rm=T)
    }
    zstab.samp[irep,]<-zstab
}
```

We look at the performance of $\widehat{\mu}_{\text{IPW}}(z)$ with the stabilized weight:

```
par(mar=c(4,5,2,0))
zstab.summary<-apply(zstab.samp,2,quantile,prob=c(0.025,0.50,0.975),na.rm=TRUE)
plot(zgrid,zstab.summary[2,],type='l',xlab='z',
        ylab=expression(paste(hat(E),"[Y(z)]")),col='red')
lines(zgrid,zstab.summary[1,],lty=2,col='red')
lines(zgrid,zstab.summary[3,],lty=2,col='red')
legend(zgrid[1],28,c(expression(paste(hat(E),"[Y(z)]")),expression(paste(E,"[Y(z)]"))),
        col=c('red','black'),lty=1)
title(expression(paste(hat(E),"[Y(z)]",' with stabilized weight')))
lines(zgrid,intercept.val+be[2]*zgrid)
```




```
fit.true<-lm(zstab.summary[2,]~zgrid)
coef(summary(fit.true))
```

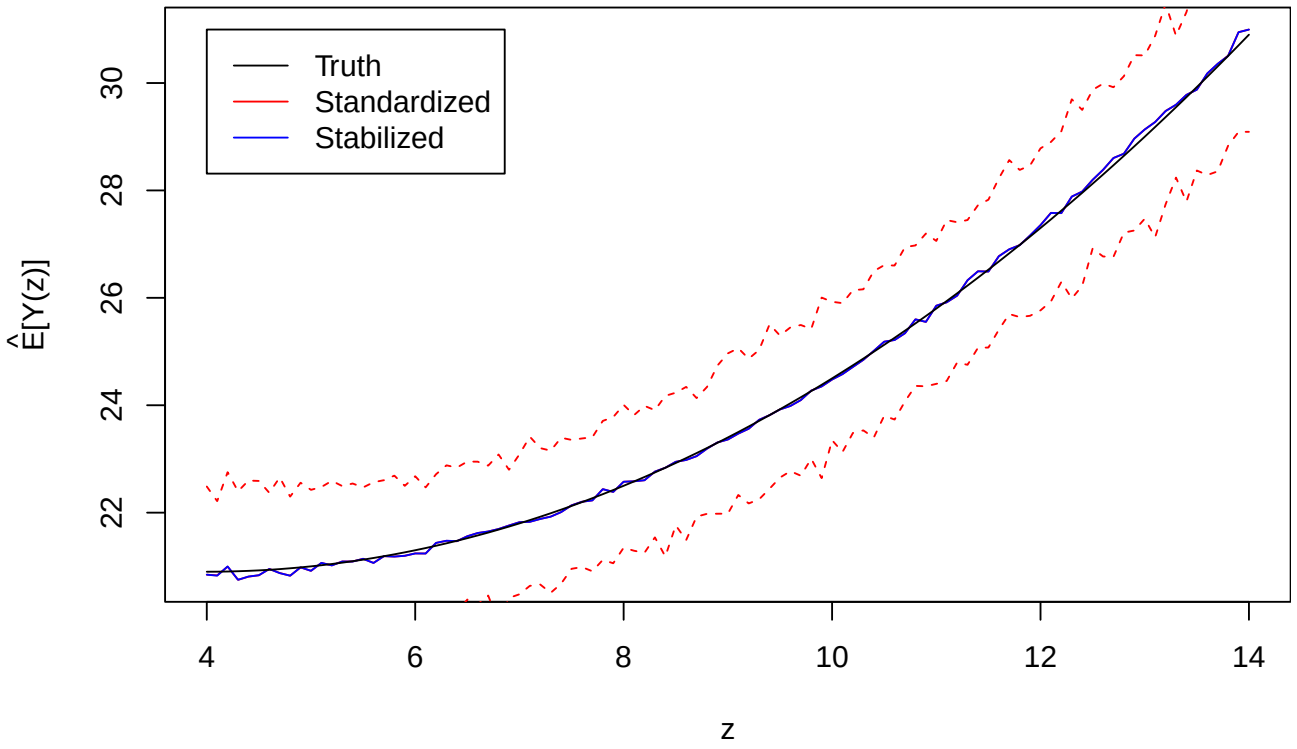
```
+             Estimate Std. Error  t value      Pr(>|t|)
+ (Intercept) 14.31811 0.01883054 760.3662 2.897801e-188
+ zgrid        1.01090 0.00199045 507.8752 6.439593e-171
```

EXAMPLE: The methods also work for quadratic data generating models

$$\mu(x, z) = \mathbf{x}_0\beta + \psi_0 z + \psi_1 z^2$$

```
#Monte Carlo study
zhat.samp<-zstab.samp<-matrix(0,nrow=nreps,ncol=nZ)
for(irep in 1:nreps){
    X<-rmvn(n,mu=Mu,Sigma)
    Xal<-cbind(1,X)
    ps.true<-Xal %*% al
    Z<-rnorm(n,ps.true,sigZ)
    Xm<-cbind(1,Z,X)
    muval<-as.vector(Xm %*% be)
    Y<-rnorm(n,muval+0.1*(Z-Z0)^2,sigY)

    fnum<-lm(Z~1)
    fden<-lm(Z~X)
    den<-dnorm(Z,predict(fden),sd(fden$residuals))
    ratio<-dnorm(Z,predict(fnum),sd(fnum$residuals))/den

    zhat<-zstab<-rep(0,nZ)
    for(ix in 1:nZ){
        zl<-zgrid[ix]-d
        zu<-zgrid[ix]+d
        w0<-(Z > zl & Z < zu)/den
        w1<-(Z > zl & Z < zu)*ratio
        zhat[ix]<-sum(w0*Y,na.rm=T)/sum(w0,na.rm=T)
        zstab[ix]<-sum(w1*Y,na.rm=T)/sum(w1,na.rm=T)
    }
    zhat.samp[irep,]<-zhat
    zstab.samp[irep,]<-zstab
}
```

We look at the performance of $\widehat{\mu}_{\text{IPW}}(z)$ with the stabilized weight:



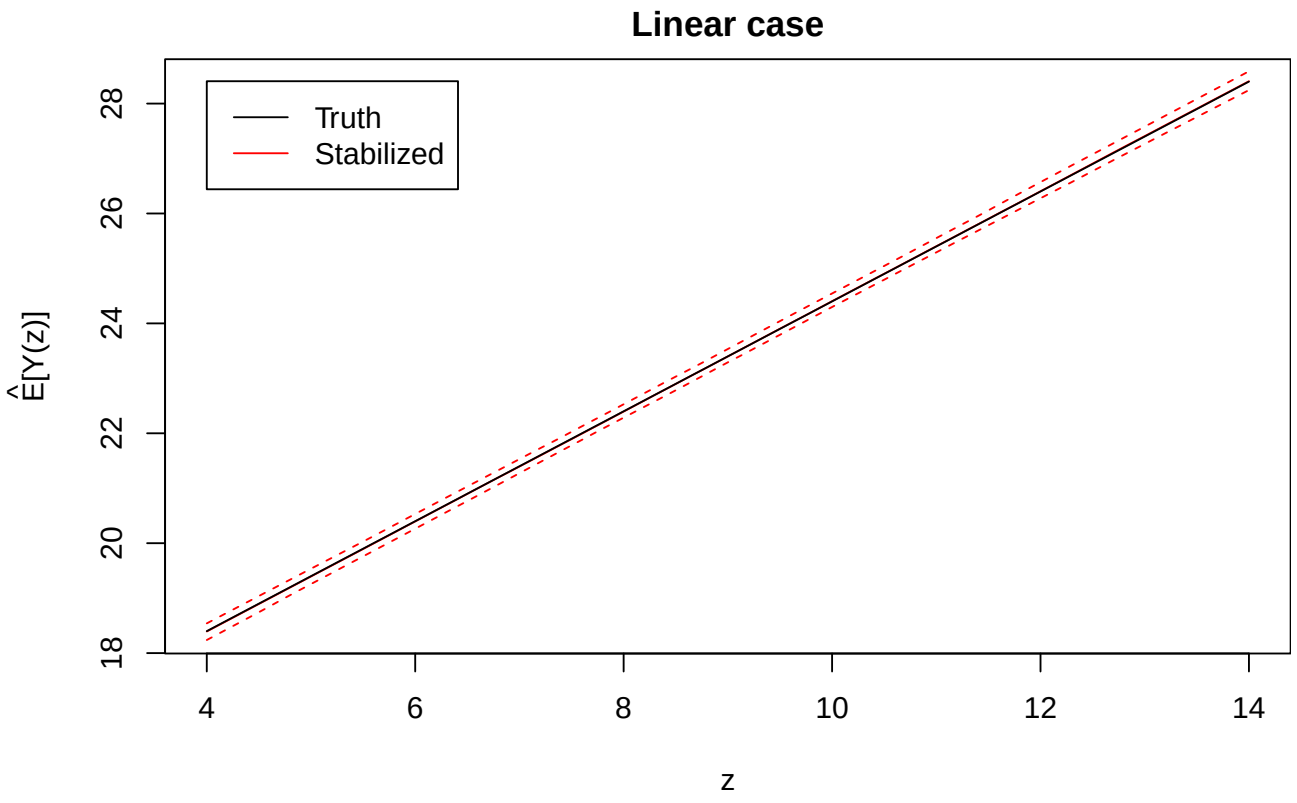
Weighted Least Squares: In weighted least squares (WLS), estimation is carried out according to a proposed mean model, with minimization of a weighted least squares objective function. For example, the mean model $m(x,z) \equiv m(z) = \beta_0 + \psi_0 z$ might be fitted according to the WLS objective function

$$\sum_{i=1}^{n} w(x_i, z_i)(y_i - \beta_0 - \psi_0 z_i)^2$$

which can also be carried out using the `lm` function.

```
#Monte Carlo study
n<-1000
nreps<-200
d<-0.05
zstab.samp<-matrix(0,nrow=nreps,ncol=nZ)
for(irep in 1:nreps){
    X<-rmvn(n,mu=Mu,Sigma)
    Xal<-cbind(1,X)
    ps.true<-Xal %*% al
    Z<-rnorm(n,ps.true,sigZ)
    Xm<-cbind(1,Z,X)
    muval<-as.vector(Xm %*% be)
    Y<-rnorm(n,muval,sigY)

    fnum<-lm(Z~1)
    fden<-lm(Z~X)
    ratio<-dnorm(Z,predict(fnum),sd(fnum$residuals))/dnorm(Z,predict(fden),sd(fden$residuals))

    fitw<-lm(Y~Z,weight=ratio)
    zstab.samp[irep,]<-predict(fitw,newdata=data.frame(Z=zgrid))
}
```

We look at the performance of $\widehat{\mu}_{\text{IPW}}(z)$ with the stabilized weight:



```r
#Monte Carlo study
n<-1000
nreps<-200
d<-0.05
zhat.samp<-zstab.samp<-matrix(0,nrow=nreps,ncol=nZ)
for(irep in 1:nreps){
    X<-rmvn(n,mu=Mu,Sigma)
    Xal<-cbind(1,X)
    ps.true<-Xal %*% al
    Z<-rnorm(n,ps.true,sigZ)
    Xm<-cbind(1,Z,X)
    muval<-as.vector(Xm %*% be)
    Y<-rnorm(n,muval+0.1*(Z-Z0)^2,sigY)

    fnum<-lm(Z~1)
    fden<-lm(Z~X)
    den<-dnorm(Z,predict(fden),sd(fden$residuals))
    ratio<-dnorm(Z,predict(fnum),sd(fnum$residuals))/den

    fitw<-lm(Y~Z+I(Z^2),weight=ratio)
    zstab.samp[irep,]<-predict(fitw,newdata=data.frame(Z=zgrid))
}
```

We look at the performance of $\widehat{\mu}_{\text{IPW}}(z)$ with the stabilized weight: