HDconfounding

Joseph Antonelli

2018-02-12

In this vignette we will show you how to use the main functions of the HDconfounding R package. We will simulate data under both a binary and continuous outcome and show how to analyze them using the various functions in the package. For the sake of building the vignette, we will use a relatively small number of covariates throughout (p=20), however, the methods extend to high-dimensional data. We simply restrict to a lower dimensional case here because the code needs to run each time a new user wants to build the vignette, and the functions used are no different for high-dimensional (p > n) data.

Loading and building the package

To build the package the user must install the devtools R package and then run the following code.

```
library(devtools)
install_github(repo = "jantonelli111/HDconfounding")
```

```
library(HDconfounding)
```

If the user wishes to build the vignette (which will take a couple minutes to build), then the following code can be run.

```
library(devtools)
install_github(repo = "jantonelli111/HDconfounding", build_vignettes = TRUE)
library(HDconfounding)
```

And to see the vignette the user can type the following into R. Note that the vignette is also available on the github repository github.com/jantonelli111/HDconfounding, which won't require building the vignette.

```
vignette("HDconfounding", package="HDconfounding")
#> Warning: vignette 'HDconfounding' not found
```

What this package can and can't be used for

The software in this package is intended to estimate causal effects with a large (potentially larger than the sample size) covariate space. The types of data that can be addressed in this package are continuous and binary outcomes, as well as binary or continuous treatments. Our software can estimate causal effects assuming either a homogeneous or heterogeneous treatment effect. It is important to note, however, that the current software only allows for heterogeneous treatment effects when the exposure/treatment is binary. If the treatment is continuous then the homogeneous versions of the codes should be used. For the sake of this vignette, we will mostly restrict to binary treatments, though the use of continuous treatments is exactly the same, except the heterogeneous functions should not be used.

Binary treatments

Continuous outcome, homogeneous treatment effect

First let's generate some data to practice estimating treatment effects with:

```
n = 200
p = 20
x = matrix(rnorm(n*p), n, p)
z = rbinom(n, 1, p=pnorm(0.7*x[,1] + 0.3*x[,2]))
y = rnorm(n, mean=z + 0.3*x[,1] + 0.6*x[,2] + 0.5*x[,3], sd=1)
```

So our sample size is 200, we have 20 covariates, the first 2 covariates are important for confounding adjustment, and the 3rd covariate is a predictor of the outcome. In this case, the treatment effect is homogeneous, i.e does not vary for different values of the covariates. We will now estimate it using both the homogeneous and heterogeneous versions of our approach. Both should yield unbiased answers in this setting, though we expect the heterogeneous approach to be slightly less efficient as it is a more complex model with more parameters.

First, to estimate the homogeneous treatment effect we can run the following

```
sslEB = SSL(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0="EB")
#> [1] "Running initial empirical Bayes estimates to calculate weights"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights"
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

This model estimates the tuning parameter λ_0 with empirical Bayes. We recommend this as it is difficult to have prior knowledge about what value this parameter should take. If the user wants to specify their own value of λ_0, then weight must also be provided (w in the main manuscript) and we recommend a value of 0.01.

```
ssl = SSL(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0=15, weight = 0.01)
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

The empirical Bayes estimate is calculated using MCMC so the running time is a little longer than pre-selecting λ_0, which is one reason one might want to choose their own value if they have a large data set. It is important to note though that a poorly chosen value of λ_0 can lead to poor estimation so we highly recommend the EB option. Let's look at the results on the treatment effect:

```
sslEB$TreatEffect
#> [1] 1.076095
sslEB$TreatEffectCI
#>     2.5%    97.5%
#> 0.749629 1.399088
```

There are other things one can pull from the results of this analysis. We can look at the posterior inclusion probabilities of the covariates:

```
sslEB$gammaPostMean
#>  [1] 0.316 0.057 0.041 0.001 0.001 0.002 0.000 0.001 0.000 0.001 0.001
#> [12] 0.000 0.002 0.001 0.000 0.000 0.001 0.002 0.000 0.000
```

And we see that the first three covariates are included in the slab component of the prior as expected.

Continuous outcome, heterogeneous treatment effect

Now let's analyze the same data, but allowing for heterogeneous treatment effects.

```
sslEBhetero = SSLhetero(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0="EB")
#> [1] "Running initial empirical Bayes estimates to calculate weights for the treated group"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights for treated group"
#> [1] "1000 MCMC scans have finished"
#> [1] "Running main analysis for treated group now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
#> [1] "Running initial empirical Bayes estimates to calculate weights for the control group"
#> [1] "1000 MCMC scans have finished"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights for control group"
#> [1] "Running main analysis for control group now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

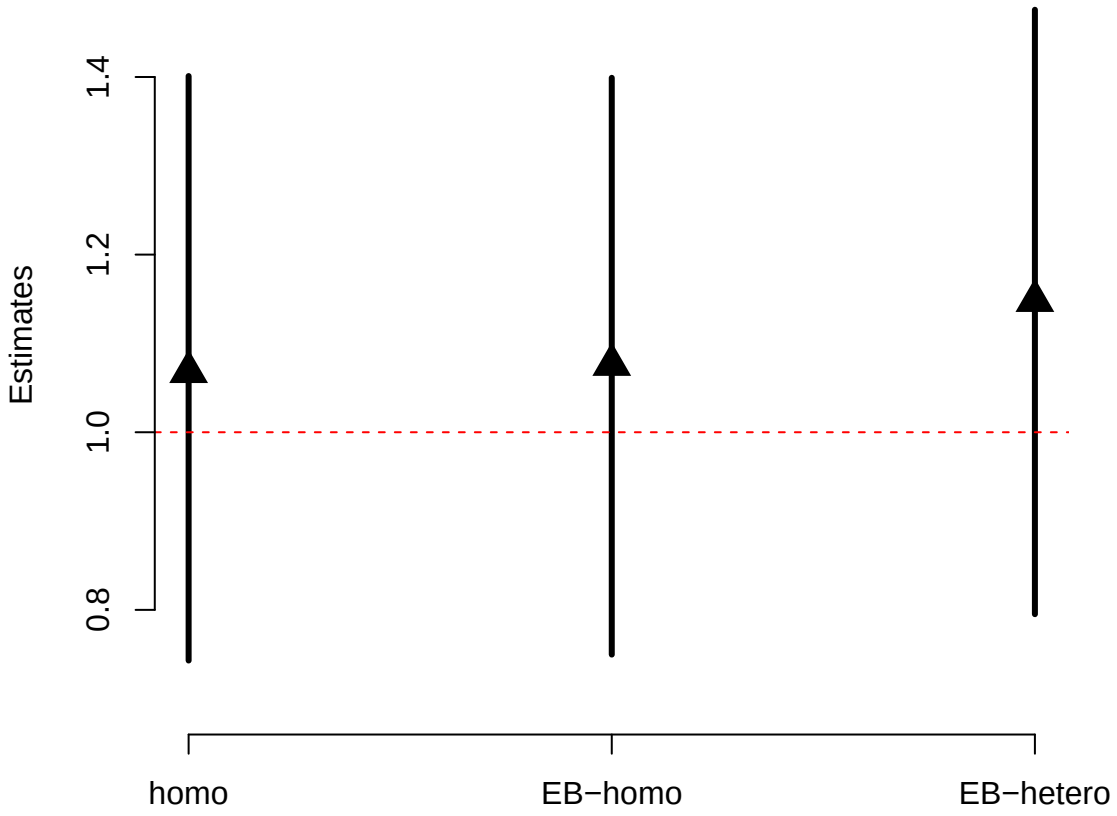
And look at the results:

```
sslEBhetero$TreatEffect
#> [1] 1.148072
sslEBhetero$TreatEffectCI
#>      2.5%     97.5%
#> 0.7951245 1.4757067
```

We can plot the estimates and intervals to compare the three models we have run. The red line is the truth.

```
estimates = c(ssl$TreatEffect, sslEB$TreatEffect,
              sslEBhetero$TreatEffect)
CIlower = c(ssl$TreatEffectCI[1], sslEB$TreatEffectCI[1],
            sslEBhetero$TreatEffectCI[1])
CIupper = c(ssl$TreatEffectCI[2], sslEB$TreatEffectCI[2],
            sslEBhetero$TreatEffectCI[2])

plot(1:3, estimates, pch=17, cex=2, ylim = range(c(CIlower, CIupper)) + c(-0.05, 0.05),
     xlab="", ylab="Estimates", axes=FALSE)
segments(x0 = 1:3, x1=1:3, y0=CIlower, y1=CIupper, lwd=3)
abline(h = 1, lty=2, col="red")
axis(2)
axis(1, at=1:3, c("homo", "EB-homo", "EB-hetero"))
```



Binary outcomes

First let's generate some data to practice estimating treatment effects with:

```
n = 200
p = 20
x = matrix(rnorm(n*p), n, p)
z = rbinom(n, 1, p=pnorm(0.7*x[,1] + 0.3*x[,2]))
y = rbinom(n, 1, p=pnorm(z + 0.3*x[,1] + 0.6*x[,2] + 0.5*x[,3]))
```

First, to estimate the homogeneous treatment effect we can run the following

```
sslEBBinary = SSLBinary(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0="EB")
#> [1] "Running initial empirical Bayes estimates to calculate weights"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights"
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

Or without empirical Bayes

```
sslBinary = SSL(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0=15, weight = 0.01)
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

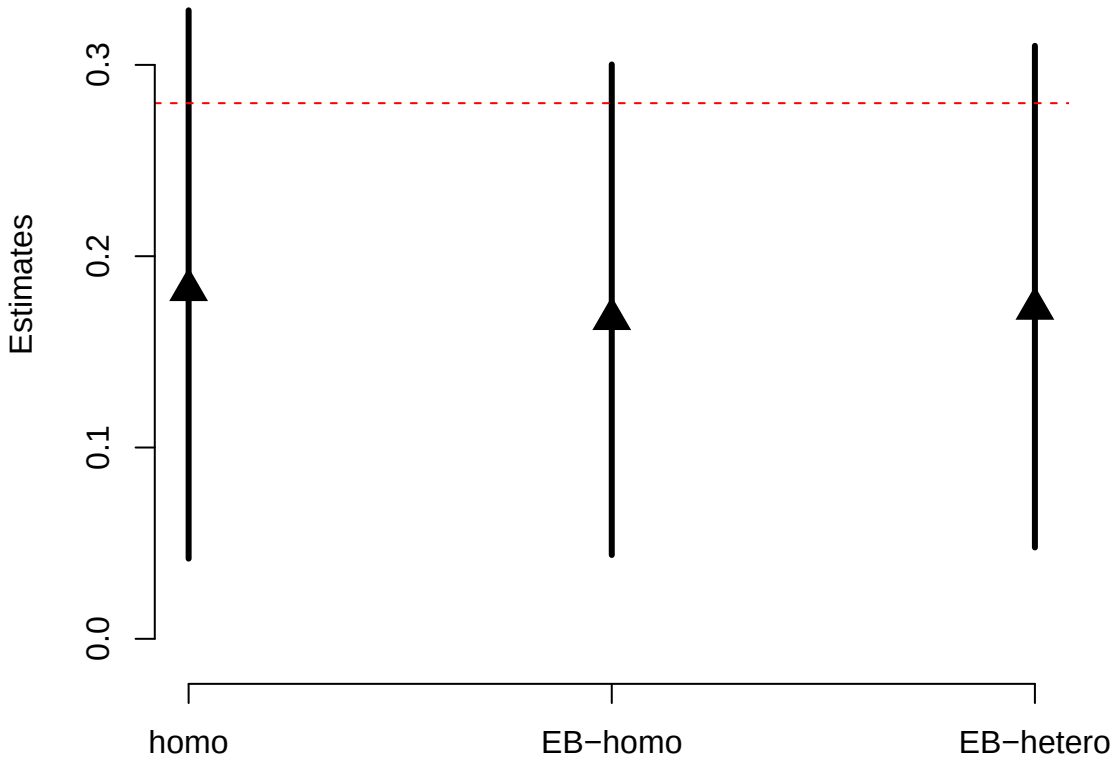
Or allowing for heterogeneous treatment effects (using empirical Bayes again)

```
sslEBheteroBinary = SSLheteroBinary(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0="EB")
#> [1] "Running initial empirical Bayes estimates to calculate weights for the treated group"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights for treated group"
#> [1] "Running main analysis for treated group now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
#> [1] "Running initial empirical Bayes estimates to calculate weights for the control group"
#> [1] "1000 MCMC scans have finished"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights for control group"
#> [1] "1000 MCMC scans have finished"
#> [1] "Running main analysis for control group now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

We can plot the estimates and intervals to compare the three models we have run. The red line is the truth.

```
estimatesBinary = c(sslBinary$TreatEffect, sslEBBinary$TreatEffect,
                    sslEBheteroBinary$TreatEffect)
CIlowerBinary = c(sslBinary$TreatEffectCI[1], sslEBBinary$TreatEffectCI[1],
                  sslEBheteroBinary$TreatEffectCI[1])
CIupperBinary = c(sslBinary$TreatEffectCI[2], sslEBBinary$TreatEffectCI[2],
                  sslEBheteroBinary$TreatEffectCI[2])

plot(1:3, estimatesBinary, pch=17, cex=2,
     ylim = range(c(CIlowerBinary, CIupperBinary)) + c(-0.05, 0.05),
     xlab="", ylab="Estimates", axes=FALSE)
segments(x0 = 1:3, x1=1:3, y0=CIlowerBinary, y1=CIupperBinary, lwd=3)
abline(h = 0.28, lty=2, col="red")
axis(2)
axis(1, at=1:3, c("homo", "EB-homo", "EB-hetero"))
```



Continuous treatment

Using the methods for a continuous treatment are very similar to those for the binary treatment, so we briefly detail them here.

continuous outcomes

For continuous outcomes, the methods are almost identitical to those for binary outcomes. The only difference is that you can't use the heterogeneous treatment effect feature.

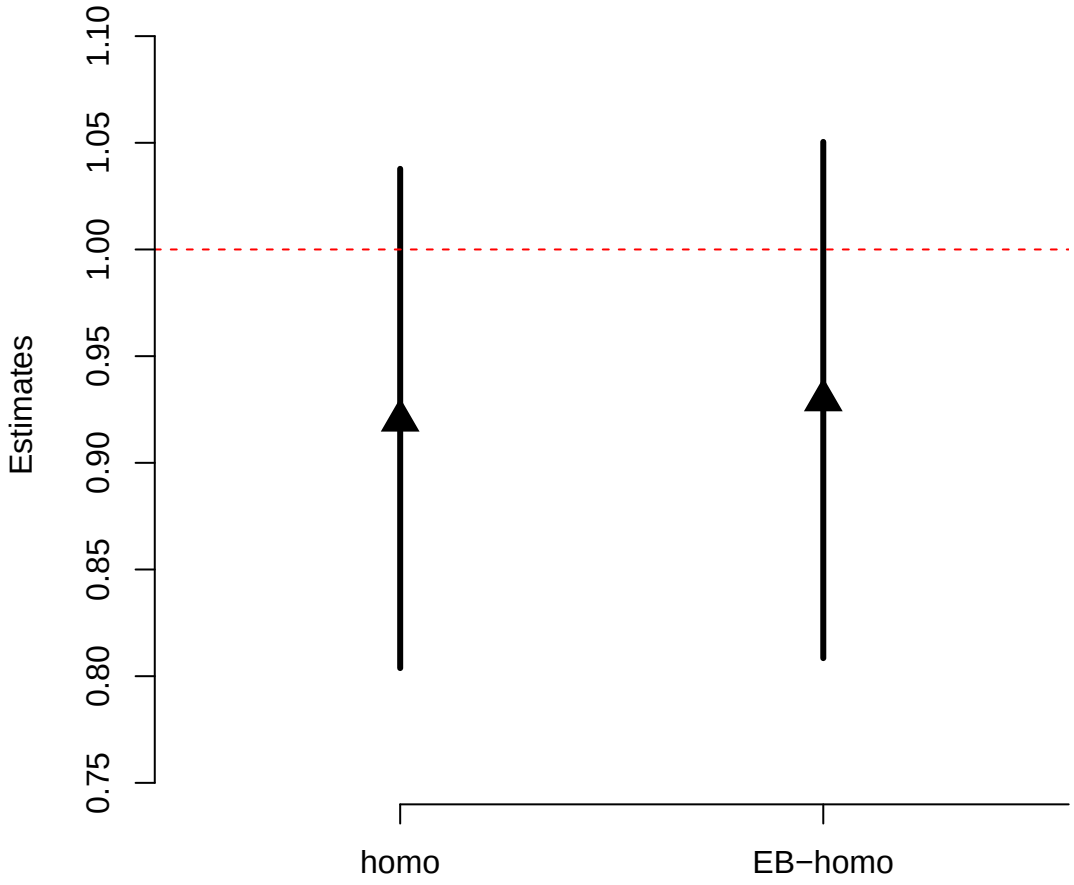
```
n = 200
p = 20
x = matrix(rnorm(n*p), n, p)
z = rnorm(n, mean=0.7*x[,1] + 0.3*x[,2])
y = rnorm(n, mean=z + 0.3*x[,1] + 0.6*x[,2] + 0.5*x[,3], sd=1)

sslEB = SSL(y=y, z=z, x=x, nScans=3000, burn=1000,
            thin=2, lambda0="EB", z_type="continuous")
#> [1] "Running initial empirical Bayes estimates to calculate weights"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

```
ssl = SSL(y=y, z=z, x=x, nScans=3000, burn=1000,
          thin=2, lambda0=15, weight = 0.01, z_type="continuous")
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"

estimates = c(ssl$TreatEffect, sslEB$TreatEffect)
CIlower = c(ssl$TreatEffectCI[1], sslEB$TreatEffectCI[1])
CIupper = c(ssl$TreatEffectCI[2], sslEB$TreatEffectCI[2])

plot(1:2, estimates, pch=17, cex=2,
     ylim = range(c(CIlower, CIupper)) + c(-0.05, 0.05),
     xlab="", ylab="Estimates", axes=FALSE, xlim=c(0.5, 2.5))
segments(x0 = 1:2, x1=1:2, y0=CIlower, y1=CIupper, lwd=3)
abline(h = 1, lty=2, col="red")
axis(2)
axis(1, at=1:3, c("homo", "EB-homo", "EB-hetero"))
```



You get an error message if you try to use the heterogeneous version of the functions

```
try({
  SSLhetero(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0="EB")
})
```

binary outcomes

There are slight differences when using a continuous treatment and a binary outcome. The reason for this is that the treatment effect depends on other covariates and the value of the treatment itself. One example is that the treatment effect going from z=0 to z=1 is not the same as from z=1 to z=2, etc. When using our programs with a binary outcome and a continuous treatment, you must also specify which two values you want to compare.

First let's generate some data:

```
n = 200
p = 20
x = matrix(rnorm(n*p), n, p)
z = rnorm(n, mean=0.7*x[,1] + 0.3*x[,2])
y = rbinom(n, 1, p=pnorm(z + 0.3*x[,1] + 0.6*x[,2] + 0.5*x[,3]))
```

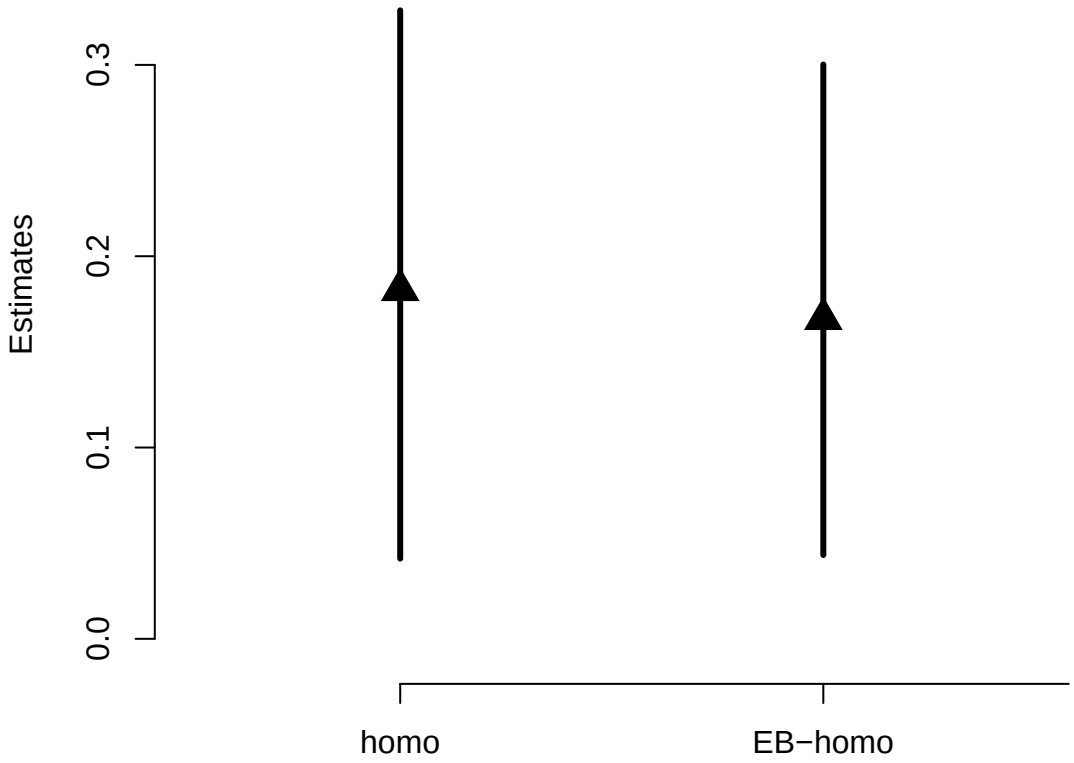
Now we can analyze it, but notice we have now added a parameter called comparison_groups

```
sslEB = SSLBinary(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2,
                  lambda0="EB", z_type="continuous", comparison_groups = c(2,0))
#> [1] "Running initial empirical Bayes estimates to calculate weights"
#> [1] "1000 MCMC scans have finished"
#> [1] "Now estimating empirical Bayes estimates of Lambda0 conditional on weights"
#> [1] "1000 MCMC scans have finished"
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
ssl = SSLBinary(y=y, z=z, x=x, nScans=3000, burn=1000, thin=2, lambda0=15,
                weight = 0.01, z_type="continuous", comparison_groups = c(2,0))
#> [1] "Running final analysis now"
#> [1] "1000 MCMC scans have finished"
#> [1] "2000 MCMC scans have finished"
#> [1] "3000 MCMC scans have finished"
```

This parameter tells us essentially what estimand we're interested in. In the case of a binary treatment it is always (1,0) because we're comparing treated to controls. When we have continuous treatments, we need to specify the two values of the treatment we are comparing. In this case we chose 2 and 0, so our code will output estimates of $E(Y^2 - Y^0)$

```
estimatesBinary = c(sslBinary$TreatEffect, sslEBBinary$TreatEffect)
CIlowerBinary = c(sslBinary$TreatEffectCI[1], sslEBBinary$TreatEffectCI[1])
CIupperBinary = c(sslBinary$TreatEffectCI[2], sslEBBinary$TreatEffectCI[2])

plot(1:2, estimatesBinary, pch=17, cex=2,
     ylim = range(c(CIlowerBinary, CIupperBinary)) + c(-0.05, 0.05),
     xlab="", ylab="Estimates", axes=FALSE, xlim=c(0.5, 2.5))
segments(x0 = 1:2, x1=1:2, y0=CIlowerBinary, y1=CIupperBinary, lwd=3)
axis(2)
axis(1, at=1:3, c("homo", "EB-homo", "EB-hetero"))
abline(h = 0.437, lty=2, col="red")
```


Estimates
0.0
0.1
0.2
0.3
homo
EB−homo