CWVSmix: Critical Window Variable Selection for Mixtures

CWVSmix_Example

[1] Simulate data for analysis:

- Setting the reproducibility seed and initializing packages for data simulation:

```
set.seed(2155)
library(CWVSmix)
library(boot)  #Inverse logit transformation
```

```
## Warning: package 'boot' was built under R version 4.2.3
```

- Setting the global data values:

```
n<-4000  #Sample size
m<-20  #Number of exposure time periods
q<-4  #Number of pollutants

x<-matrix(1,
nrow=n,
ncol=1)  #Covariate design matrix

z_main<-matrix(rnorm(n=(n*m*q)),
nrow=n,
ncol=(m*q))  #Main effect design matrix

z_int<-matrix(0,
nrow=n,
ncol=(m*q*(q-1)/2))  #Interaction effect design matrix
counter<-1
for(j in 1:m){
for(k in 1:(q-1)){
for(l in (k+1):q){
z_int[,counter]<-z_main[,(k + (j-1)*q)]*z_main[,(l + (j-1)*q)]
counter<-counter +
         1
}
}
}

z<-matrix(0,
nrow=n,
ncol=(ncol(z_main) + ncol(z_int)))  #Full set of exposures
for(j in 1:m){
z[,(1+(j-1)*q*(q+1)/2):(j*q*(q+1)/2)]<-cbind(z_main[,(1+(j-1)*q):(j*q)],
                                             z_int[,(1+(j-1)*q*(q-1)/2):(j*q*(q-1)/2)])
}

for(j in 1:ncol(z)){
z[,j]<-(z[,j] - median(z[,j]))/IQR(z[,j])  #Data standardization (interquartile range)
}
```

- Setting the values for the statistical model parameters:

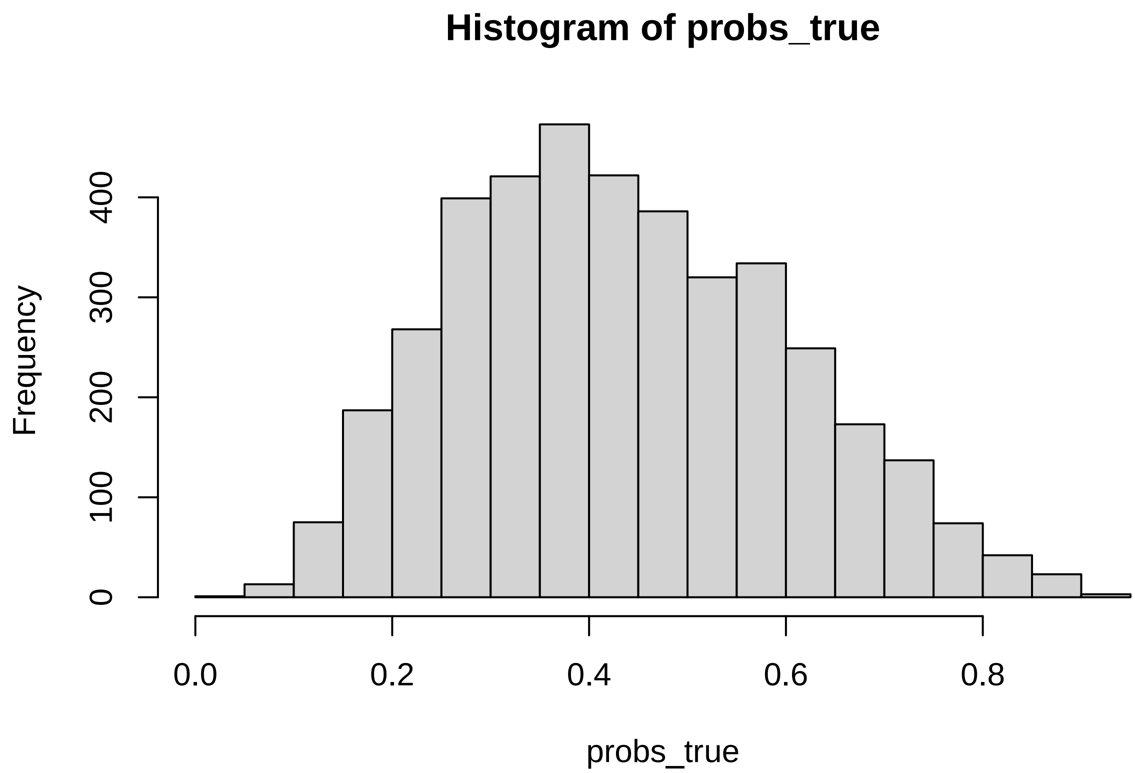
```r
beta_true<- -0.30
theta_true<-rep(0.60, times=m)
gamma_true<-c(rep(0, times=10),
rep(1, times=5),
rep(0, times=5))
alpha_true<-gamma_true*theta_true

lambda_true<-matrix(0,
nrow=m,
ncol=(q*(q+1)/2))

#Choosing pollutant/interaction weights on critical weeks
lambda_true[11,1]<-1
lambda_true[12,c(1,2,5)]<-c(0.30, 0.60, 0.10)
lambda_true[13,c(1,2,5)]<-c(0.30, 0.60, 0.10)
lambda_true[14,c(1,2,3)]<-c(0.30, 0.60, 0.10)
lambda_true[15,c(1,2,3,5,6)]<-c(0.40, 0.20, 0.10, 0.20, 0.10)

risk<-matrix(0,
nrow=n,
ncol=m)
for(j in 11:15){  #Critical weeks: 11-15
risk[,j]<-z[,(1 + (j-1)*q*(q+1)/2):(j*q*(q+1)/2)]%*%lambda_true[j,]
}

logit_p_true<-x%*%beta_true +
risk%*%alpha_true
probs_true<-inv.logit(logit_p_true)
hist(probs_true)
```




```r
trials<-rep(1, times = n)
```

- Simulating the analysis dataset:

```r
y<-rbinom(n=n,
          size=trials,
          prob=probs_true)
```

[2] Fit CWVS to identify/estimate critical windows of susceptibility:

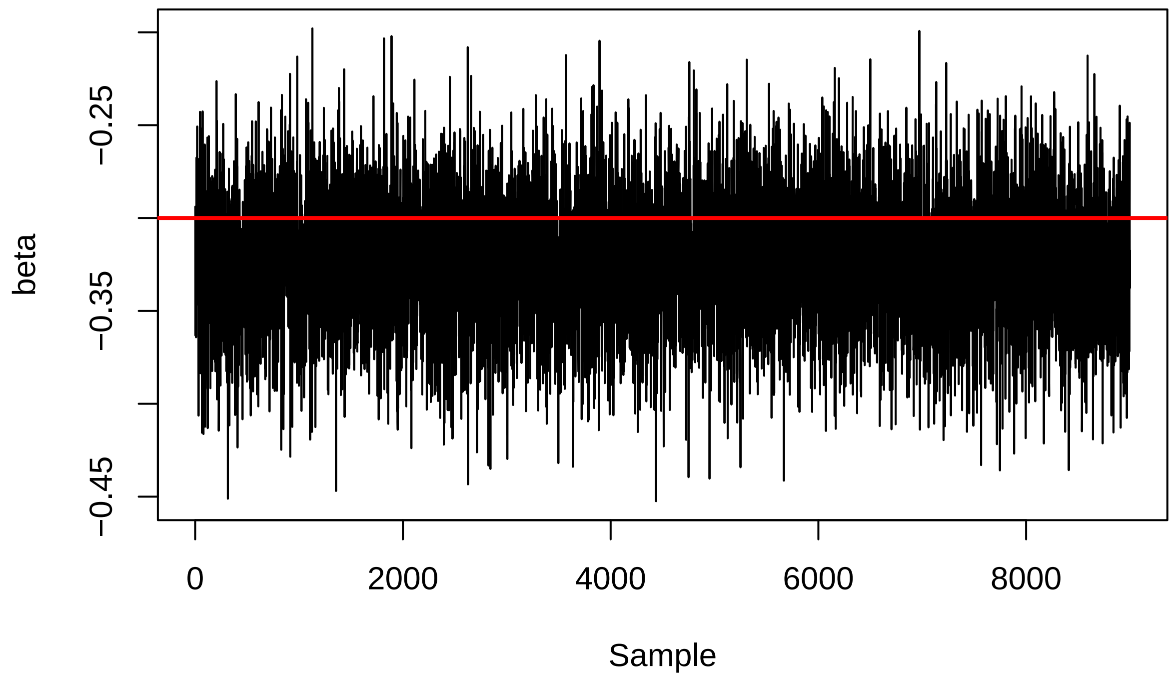
```r
results<-CWVSmix(mcmc_samples = 10000,
                 p = (q*(q+1)/2), y = y, x = x, z = z,
                 metrop_var_lambda_trans = rep(0.10, times=m),
                 metrop_var_rho_trans = 0.25,
                 metrop_var_phi1_trans = 1.00,
                 metrop_var_phi2_trans = 1.00,
                 metrop_var_A11_trans = 0.03,
                 metrop_var_A22_trans = 0.30,
                 interaction_indicator = 1,
                 likelihood_indicator = 0,
                 trials = trials)
```

```
## Progress: 10%
## lambda Acceptance (min): 14%
## lambda Acceptance (max): 61%
## rho Acceptance: 43%
## A11 Acceptance: 31%
## A22 Acceptance: 21%
```

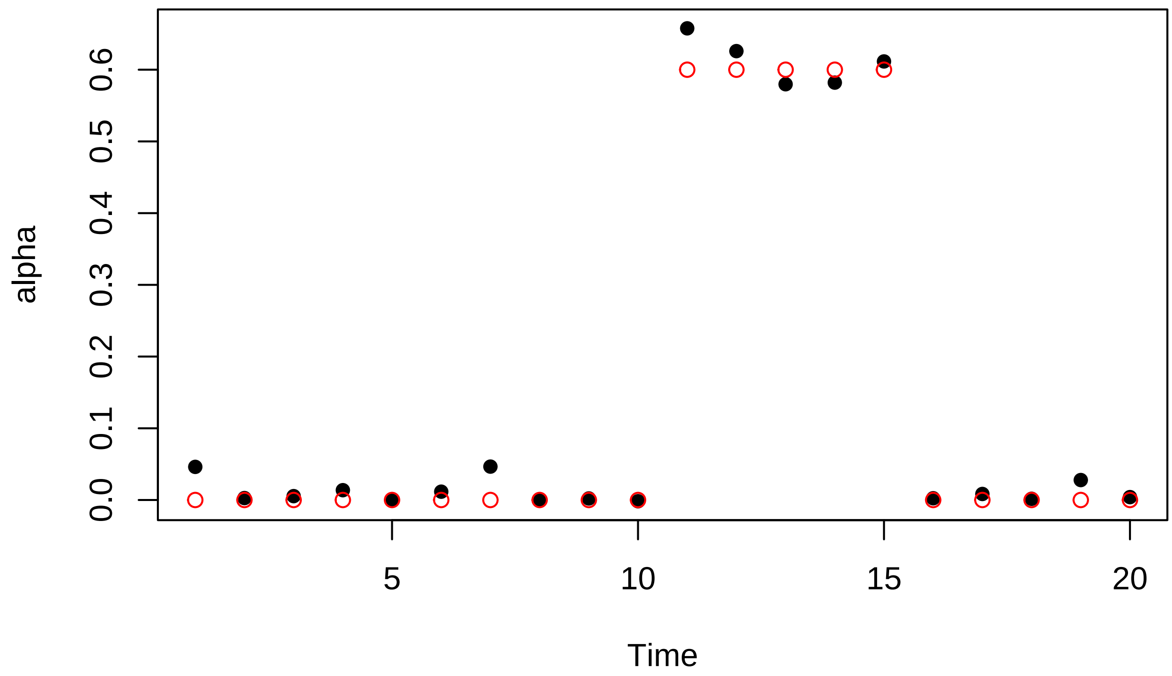
```
## phi1 Acceptance: 41%
## phi2 Acceptance: 39%
## ****************************
## Progress: 20%
## lambda Acceptance (min): 17%
## lambda Acceptance (max): 59%
## rho Acceptance: 38%
## A11 Acceptance: 29%
## A22 Acceptance: 20%
## phi1 Acceptance: 41%
## phi2 Acceptance: 41%
## ****************************
## Progress: 30%
## lambda Acceptance (min): 16%
## lambda Acceptance (max): 57%
## rho Acceptance: 37%
## A11 Acceptance: 29%
## A22 Acceptance: 28%
## phi1 Acceptance: 42%
## phi2 Acceptance: 43%
## ****************************
## Progress: 40%
## lambda Acceptance (min): 15%
## lambda Acceptance (max): 57%
## rho Acceptance: 36%
## A11 Acceptance: 29%
## A22 Acceptance: 32%
## phi1 Acceptance: 42%
## phi2 Acceptance: 43%
## ****************************
## Progress: 50%
## lambda Acceptance (min): 16%
## lambda Acceptance (max): 56%
## rho Acceptance: 35%
## A11 Acceptance: 29%
## A22 Acceptance: 33%
## phi1 Acceptance: 41%
## phi2 Acceptance: 43%
## ****************************
## Progress: 60%
## lambda Acceptance (min): 14%
## lambda Acceptance (max): 55%
## rho Acceptance: 34%
## A11 Acceptance: 29%
## A22 Acceptance: 31%
## phi1 Acceptance: 41%
## phi2 Acceptance: 43%
## ****************************
## Progress: 70%
## lambda Acceptance (min): 13%
## lambda Acceptance (max): 52%
## rho Acceptance: 33%
## A11 Acceptance: 29%
## A22 Acceptance: 31%
```

```
## phi1 Acceptance: 40%
## phi2 Acceptance: 43%
## ****************************
## Progress: 80%
## lambda Acceptance (min): 12%
## lambda Acceptance (max): 52%
## rho Acceptance: 32%
## A11 Acceptance: 29%
## A22 Acceptance: 29%
## phi1 Acceptance: 40%
## phi2 Acceptance: 43%
## ****************************
## Progress: 90%
## lambda Acceptance (min): 12%
## lambda Acceptance (max): 52%
## rho Acceptance: 32%
## A11 Acceptance: 28%
## A22 Acceptance: 27%
## phi1 Acceptance: 40%
## phi2 Acceptance: 43%
## ****************************
## Progress: 100%
## lambda Acceptance (min): 13%
## lambda Acceptance (max): 52%
## rho Acceptance: 33%
## A11 Acceptance: 28%
## A22 Acceptance: 26%
## phi1 Acceptance: 39%
## phi2 Acceptance: 43%
## ****************************
```

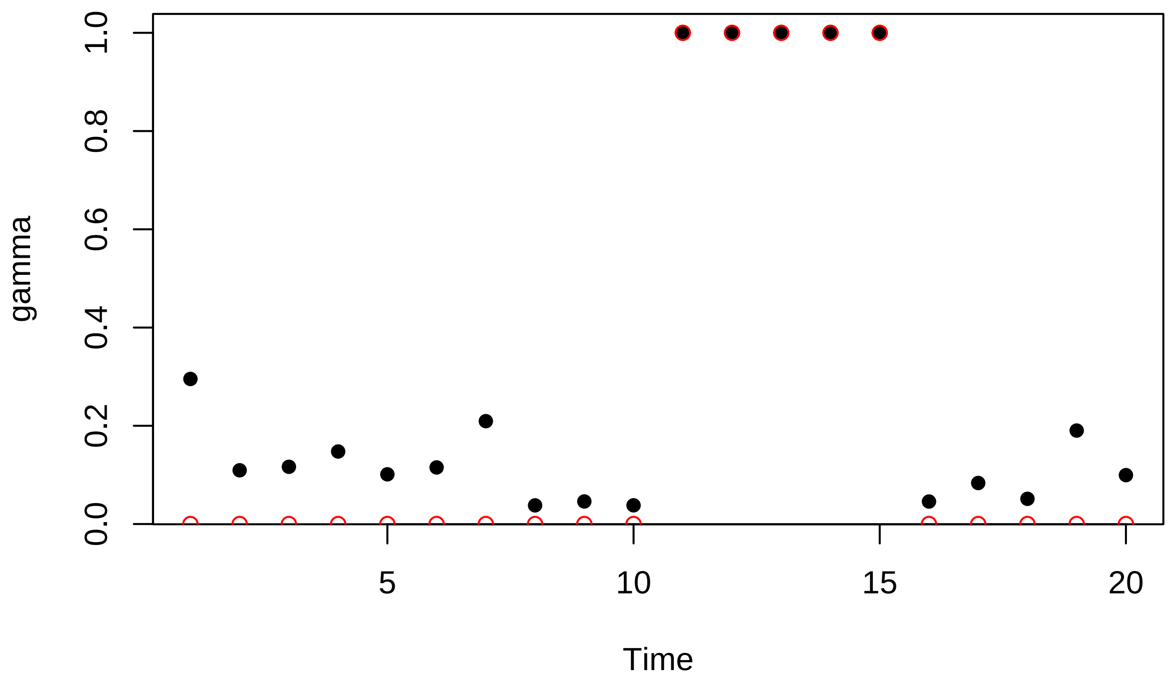
```r
plot(results$beta[1, 1001:10000],
     type="l",
     ylab="beta",
     xlab="Sample")
abline(h=beta_true,
       col="red",
       lwd=2)  #True value
```



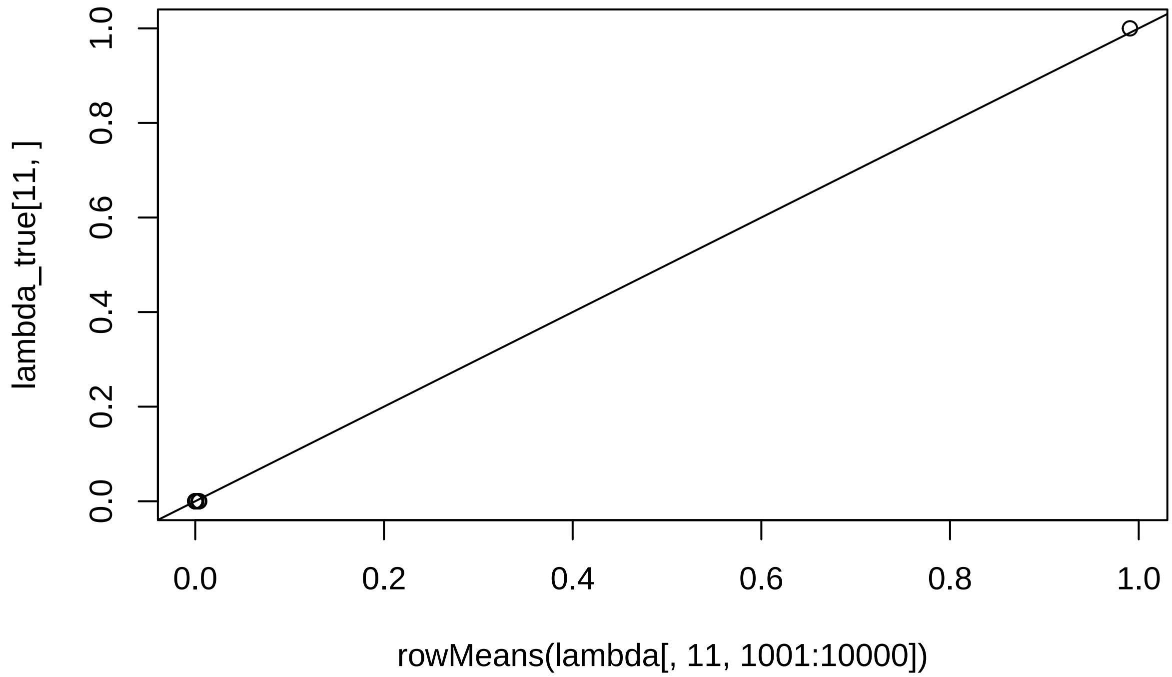

```
plot(rowMeans(results$alpha[,1001:10000]),
               pch=16,
               ylab="alpha",
               xlab="Time")
points(alpha_true,
       col="red")
```



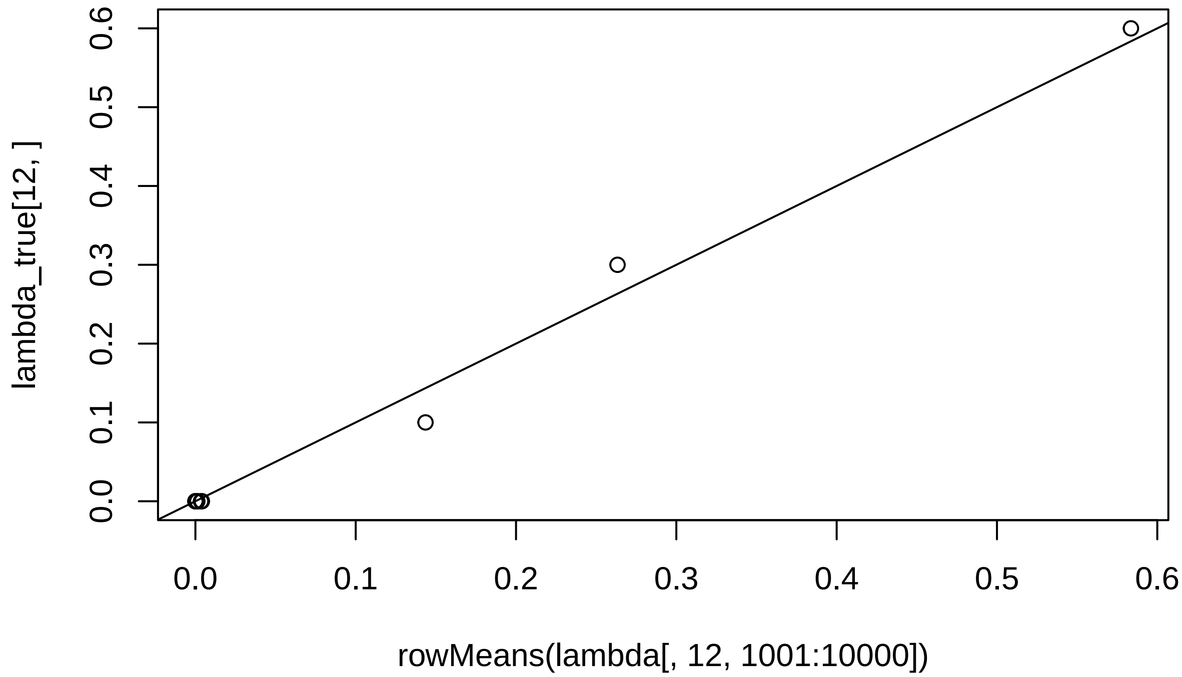

```r
plot(rowMeans(results$gamma[,1001:10000]),
     pch=16,
     ylab="gamma",
     xlab="Time")
points(gamma_true,
       col="red")
```



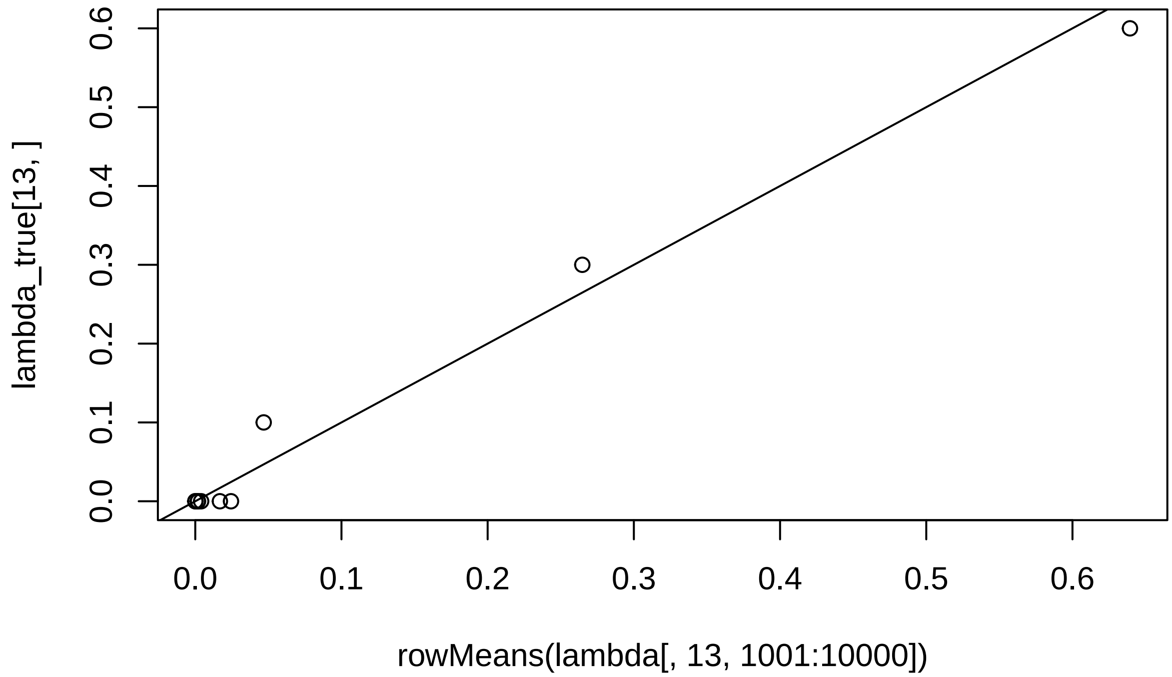

```
lambda<-array(as.numeric(unlist(results$lambda)),
              dim = c((q*(q+1)/2),
                      m,
                      10000))
plot(rowMeans(lambda[,11,1001:10000]),
     lambda_true[11,])
abline(0,1)
```



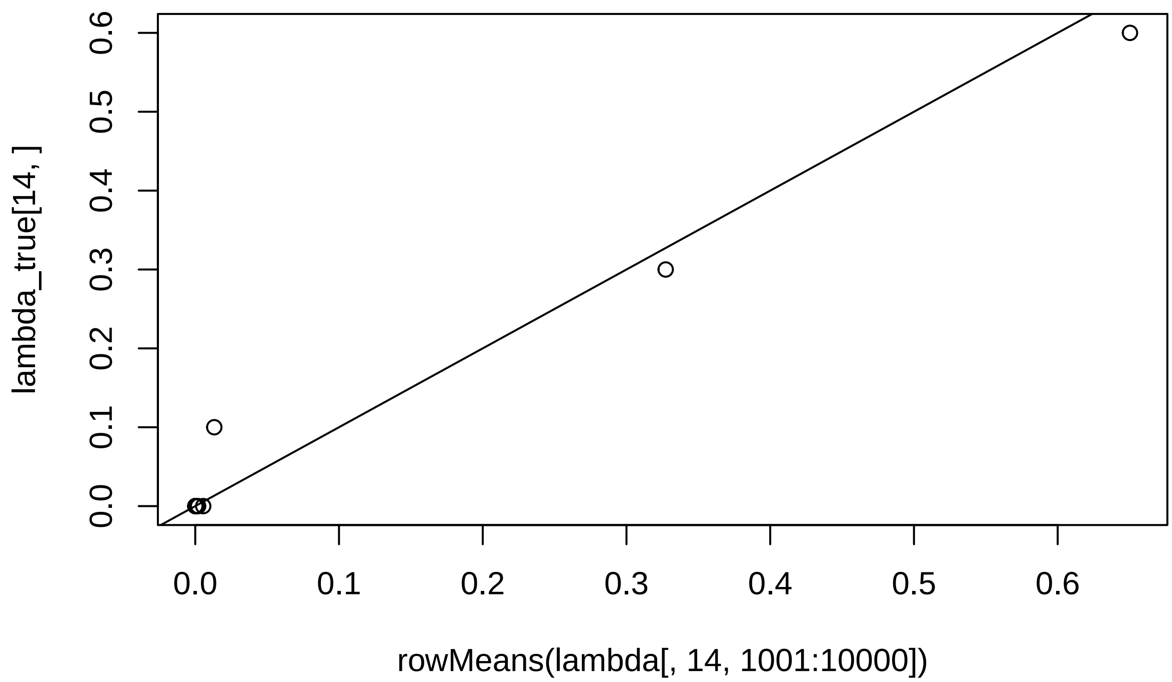

```
plot(rowMeans(lambda[,12,1001:10000]),
     lambda_true[12,])
abline(0,1)
```



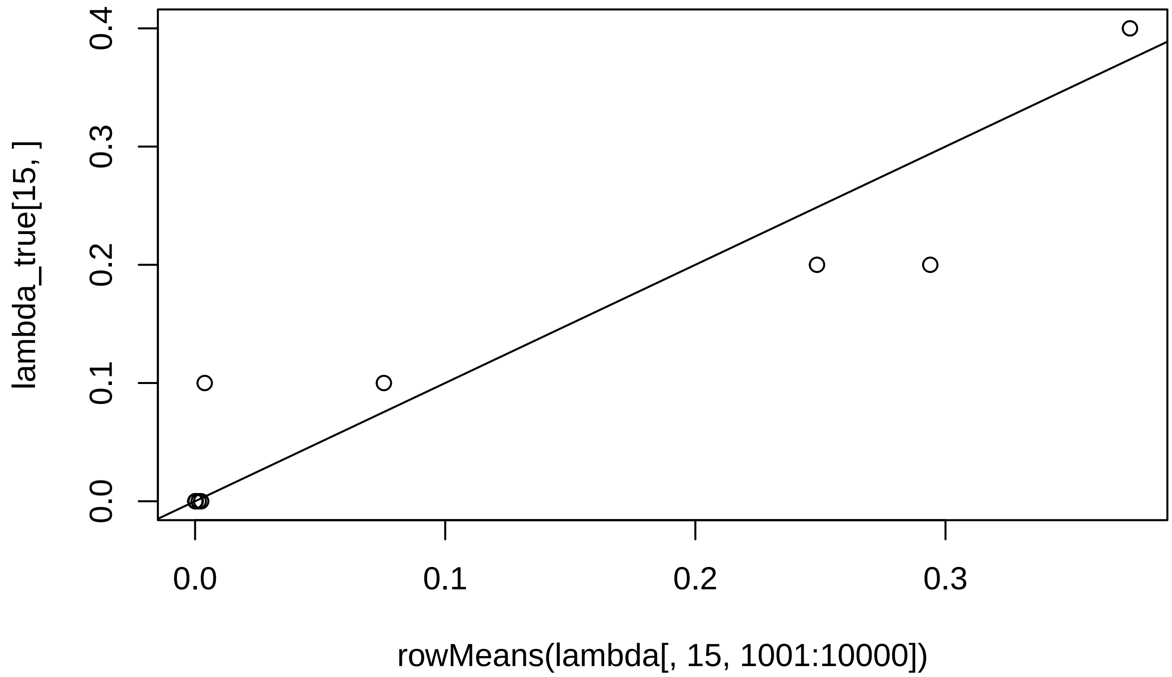

```r
plot(rowMeans(lambda[,13,1001:10000]),
     lambda_true[13,])
abline(0,1)
```




```r
plot(rowMeans(lambda[,14,1001:10000]),
     lambda_true[14,])
abline(0,1)
```




```r
plot(rowMeans(lambda[,15,1001:10000]),
     lambda_true[15,])
abline(0,1)
```


0.0
0.1
0.2
0.3
0.4
lambda_true[15,]
0.0
0.1
0.2
0.3
rowMeans(lambda[, 15, 1001:10000])