BTRR_demo

Alec Reinhardt

3/18/2022

```
source('BTRR.R')
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: proto

## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shared object '/Library
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 6): Library not loaded: /opt/X
##   Referenced from: /Library/Frameworks/R.framework/Resources/modules//R_X11.so
##   Reason: image not found

## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/Resources/
## library/tcltk/libs//tcltk.so'' had status 1

## Could not load tcltk.  Will use slower R code instead.

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: lme4

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
## Attaching package: 'lmerTest'

## The following object is masked from 'package:lme4':
##
##     lmer
## The following object is masked from 'package:stats':
##
##     step
```

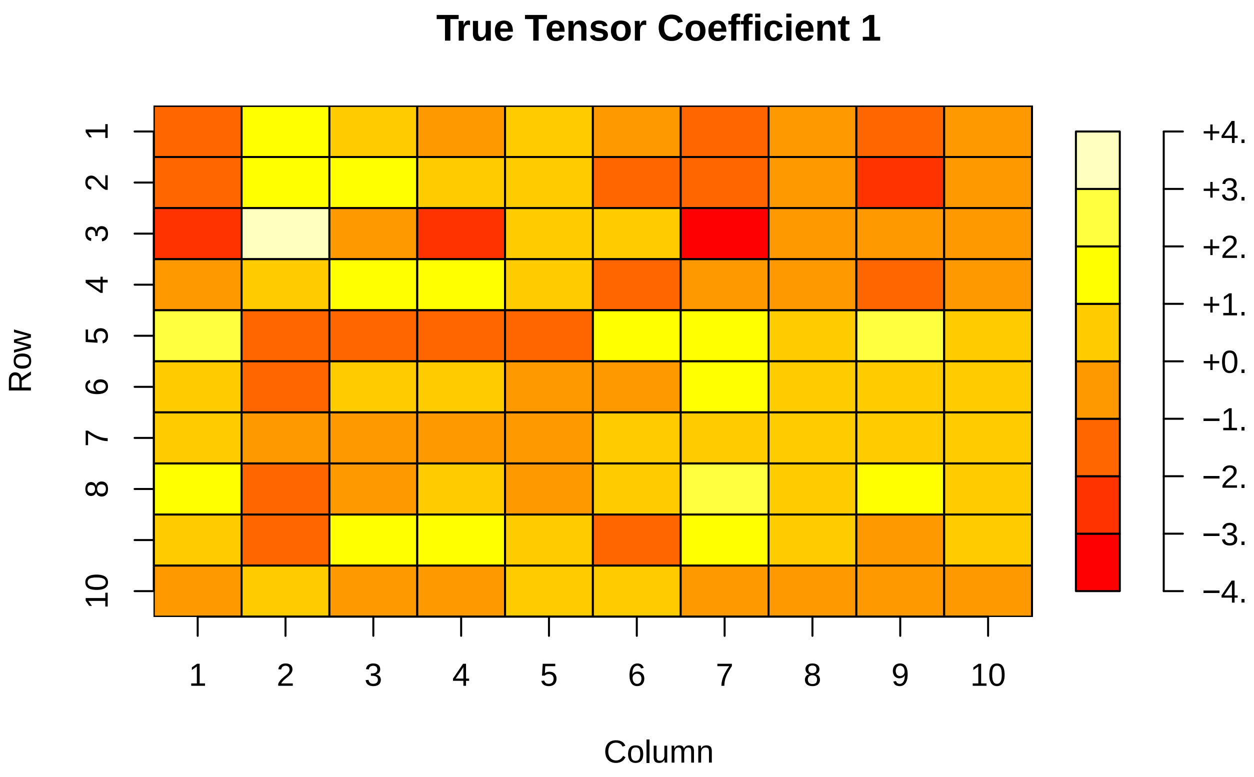
```
library(plot.matrix)
```

Simulate data

```
set.seed(1234)
N <- 50 # sample size (BTRR preferable to OLS if N is small)
K <- 3 # number of covariates
p <- c(10,10) # dimensions of each tensor outcome (BTRR preferable to OLS if prod(p) is large)
X.train <- array(rnorm(N*K), dim=c(N,K)) # simulated covariates
nIDs <- 3
ID <- sample(1:nIDs, N, replace=T) # ID index (for subjects 1-10)
table(ID) # make sure each subject has >=2 observations
```

```
## ID
##  1  2  3
## 18 19 13
```

```
# Note: for one overall intercept, set ID=NA and add a column of 1s to X.train

R.true <- 2 # true rank
# main effect coefficients: each a matrix generated from low-rank tensor decomposition,
# with normally-distributed tensor margins
Gamma.true <- t(sapply(1:K, function(x) c(TP.rankR(list(array(rnorm(p[1]*R.true),dim=c(p[1],R.true)),
                                                        array(rnorm(p[2]*R.true),dim=c(p[2],R.true))))))

# plot tensor (matrix) coefficient to see spatial patterns
plot(array(Gamma.true[1,],dim=p), main='True Tensor Coefficient 1')
```



```r
# random intercepts: generated from low-rank tensor decomposition
Bi.true <- t(sapply(1:nIDs, function(x) c(TP.rankR(list(array(rnorm(p[1]*R.true),dim=c(p[1],R.true)),
                                                      array(rnorm(p[2]*R.true),dim=c(p[2],R.true)))))
# set noise variance in simulated data
true.noise.var <- .5

# generate outcome data
Y.train <- t(sapply(1:N, function(x) X.train[x,] %*% Gamma.true + Bi.true[ID[x],]))
Y.train <- Y.train + array(rnorm(N*prod(p), sd=sqrt(true.noise.var)), dim=c(N,prod(p)))
Y.train <- array(Y.train, dim=c(N,p)) # make into N x p1 x p2 array

# set ~5% of values to be missing
p.missing <- 0.05
missing.inds <- array(rbinom(N*prod(p),1,p.missing),dim=c(N,p))
Y.train[missing.inds==1] <- NA
sum(is.na(Y.train)) / (N*prod(p))
```

```
## [1] 0.0476
```

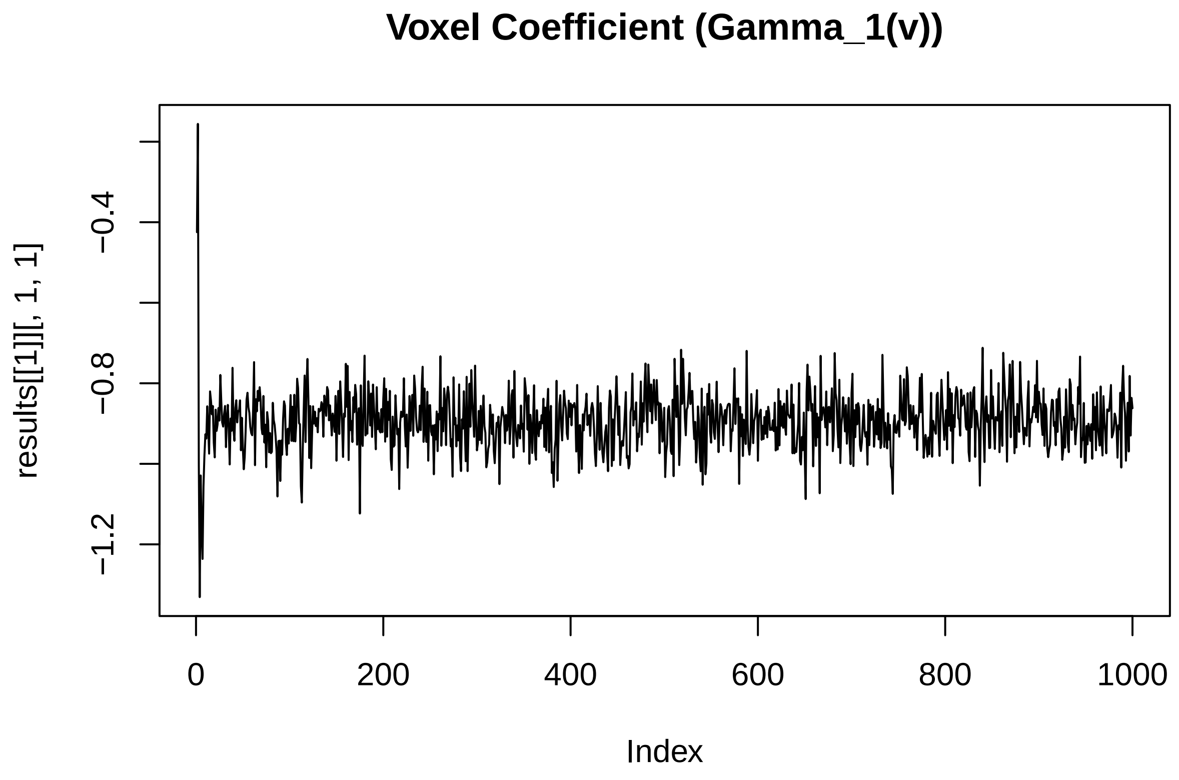
Fit model using BTRR (tensor.reg function)

```r
nsweep <- 1000 # number of MCMC iterations
R <- 2 # chosen rank for Gamma's and Bi's (tensor coefficients/intercepts)
# Note: can specify R<-c(R1,R2) for Gamma's rank R1 and Bi's rank R2
results <- tensor.reg(Y.train, X.train, ID, nsweep=nsweep, R=R, show.prog=T,prog.count=50)
```

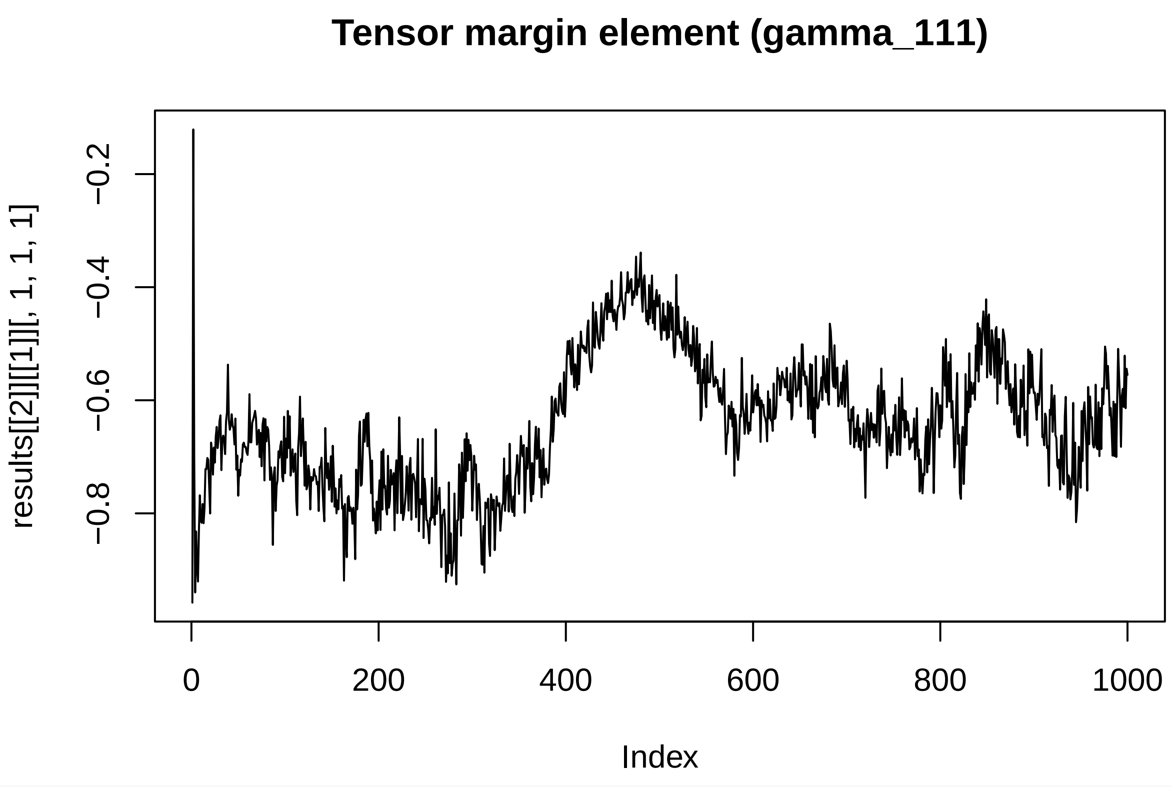
```
## [1] 50
## [1] 100
## [1] 150
## [1] 200
## [1] 250
## [1] 300
## [1] 350
## [1] 400
## [1] 450
## [1] 500
## [1] 550
## [1] 600
## [1] 650
## [1] 700
## [1] 750
## [1] 800
## [1] 850
## [1] 900
## [1] 950
## [1] 1000
```

MCMC Traceplots: Plotted for each model parameter. Expected convergence for tensor coefficients and noise variance terms.

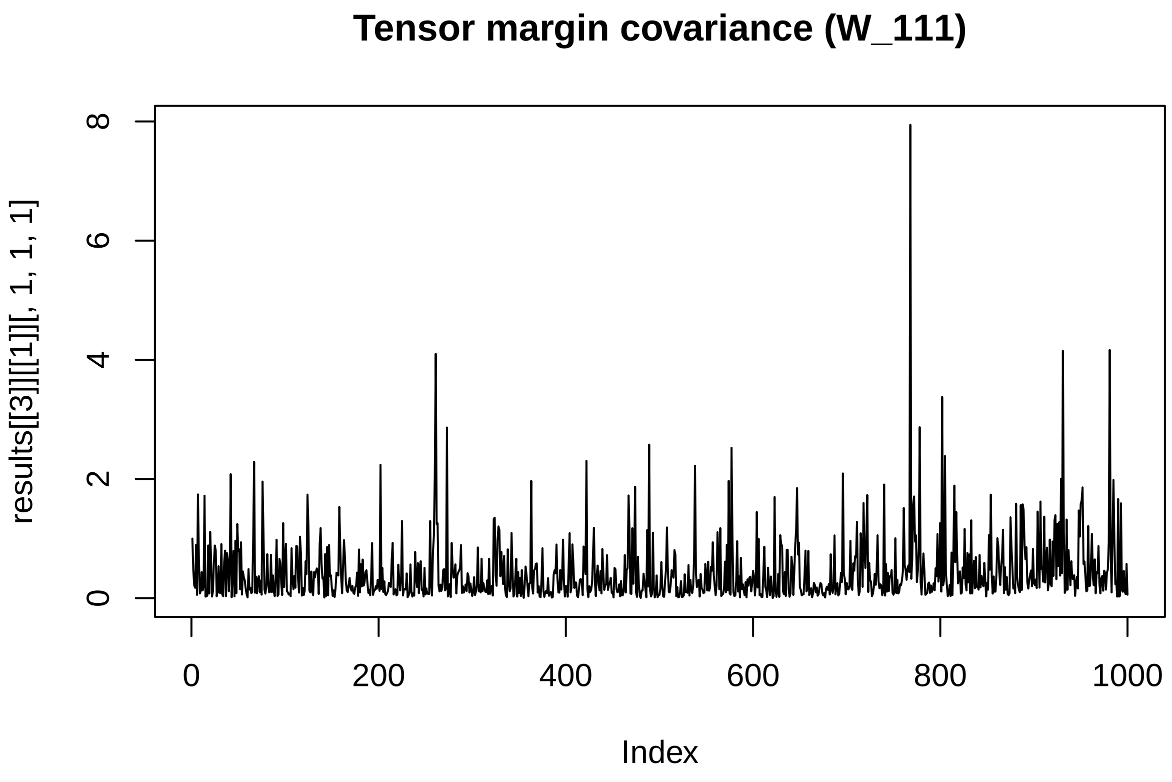
```
plot(results[[1]][,1,1],type="l",main="Voxel Coefficient (Gamma_1(v))") # first variable, first voxel
```



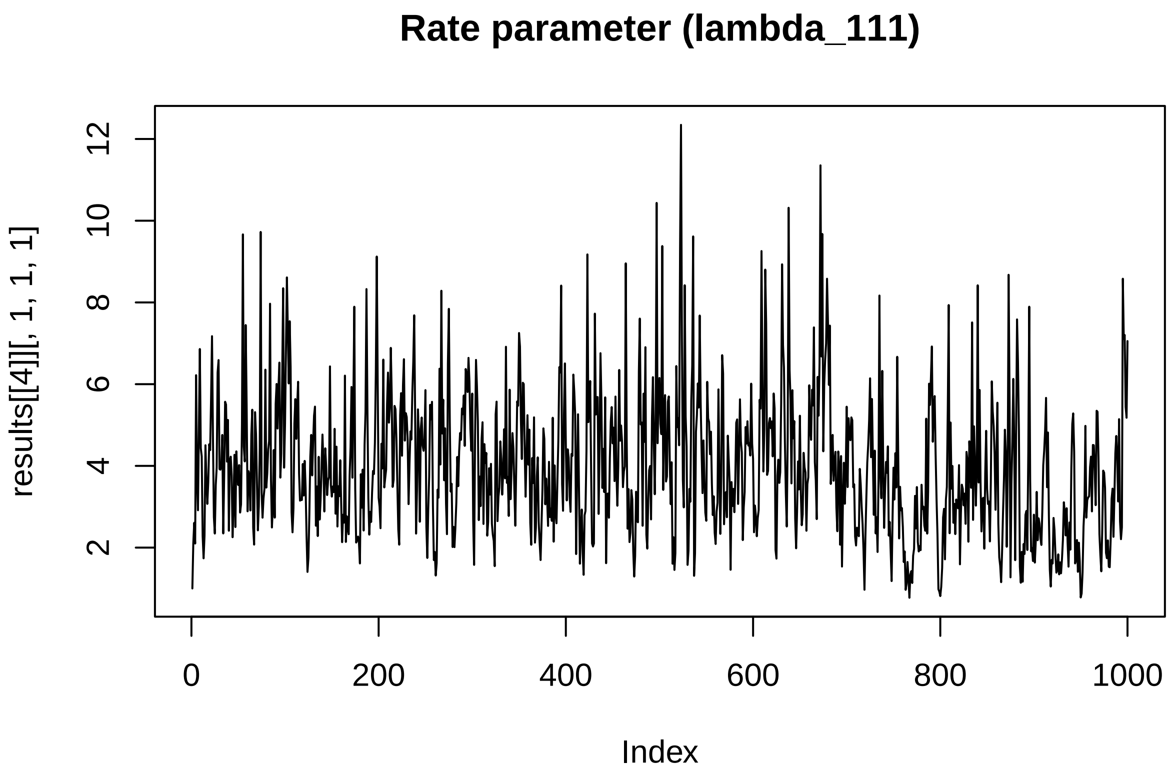
```
plot(results[[2]][[1]][,1,1,1],type="l", main="Tensor margin element (gamma_111)") # tensor margin elem
```



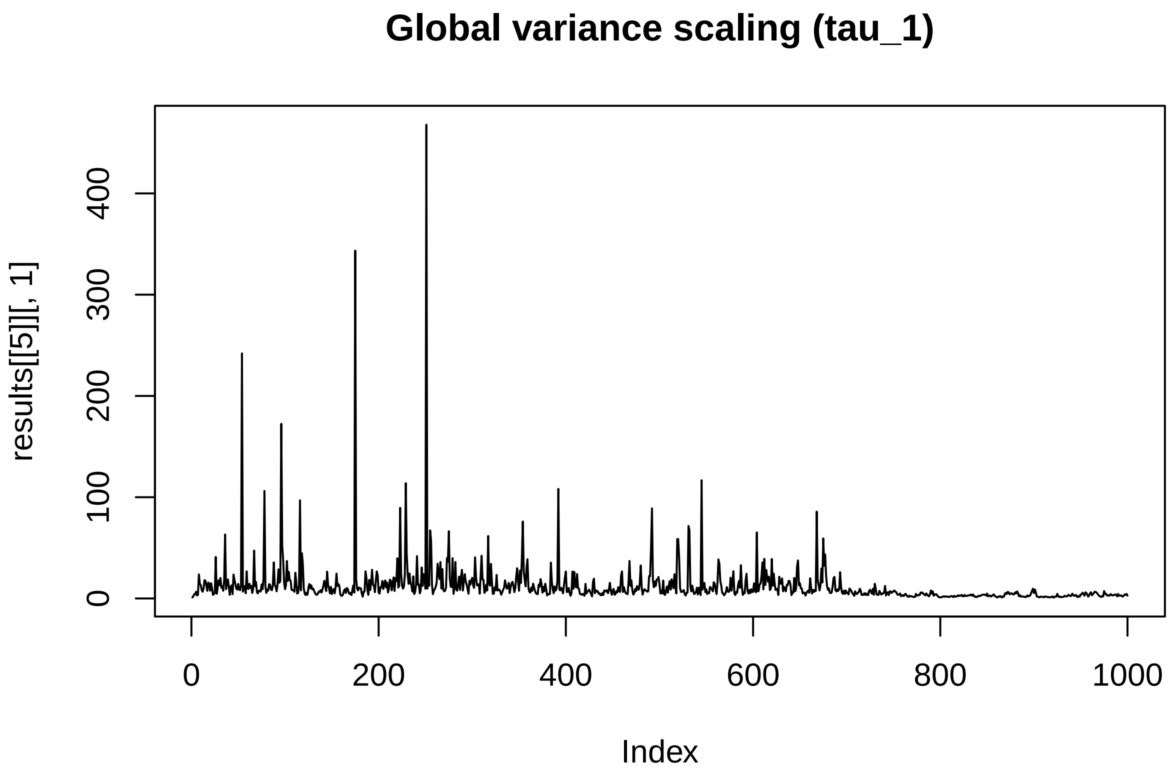
```
plot(results[[3]][[1]][,1,1,1],type="l", main="Tensor margin covariance (W_111)") # tensor margin cov d
```



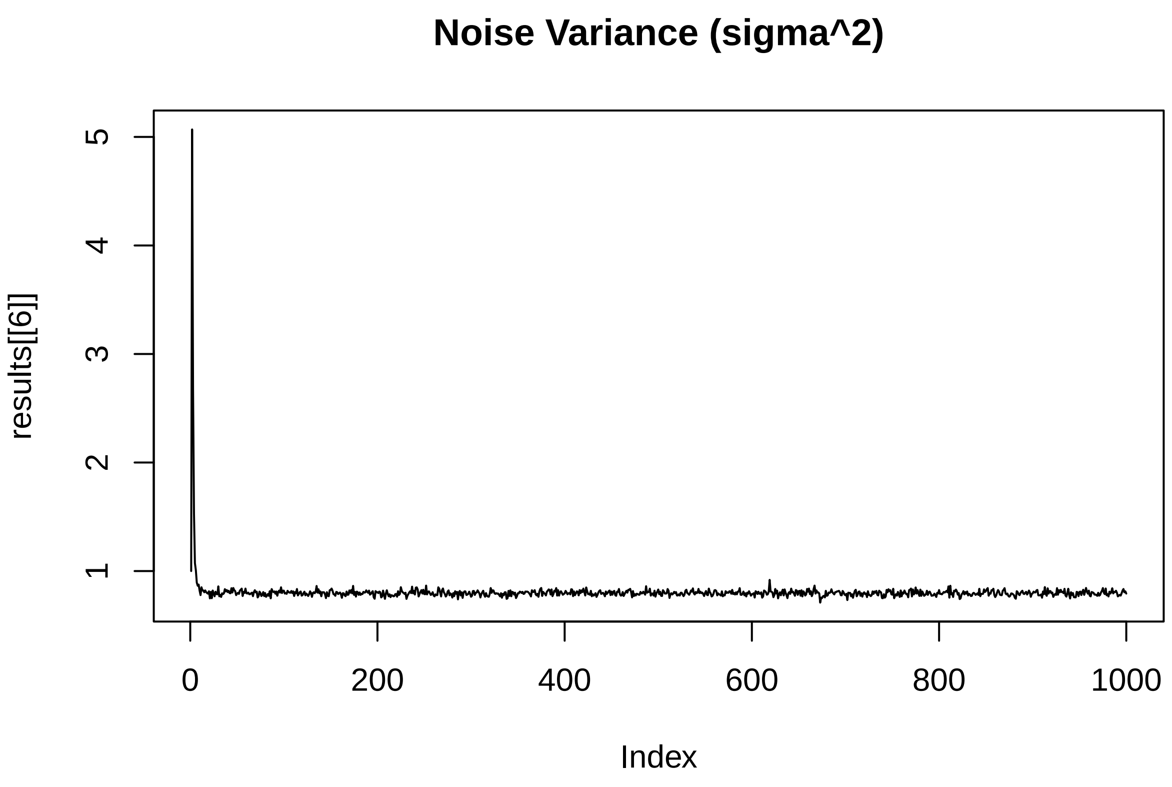
```
plot(results[[4]][,1,1,1],type="l", main="Rate parameter (lambda_111)") # lambda.drk
```



```
plot(results[[5]][,1], type="l", main="Global variance scaling (tau_1)") # tau.k
```



```
plot(results[[6]],type="l",main="Noise Variance (sigma^2)") # sigma^2
```

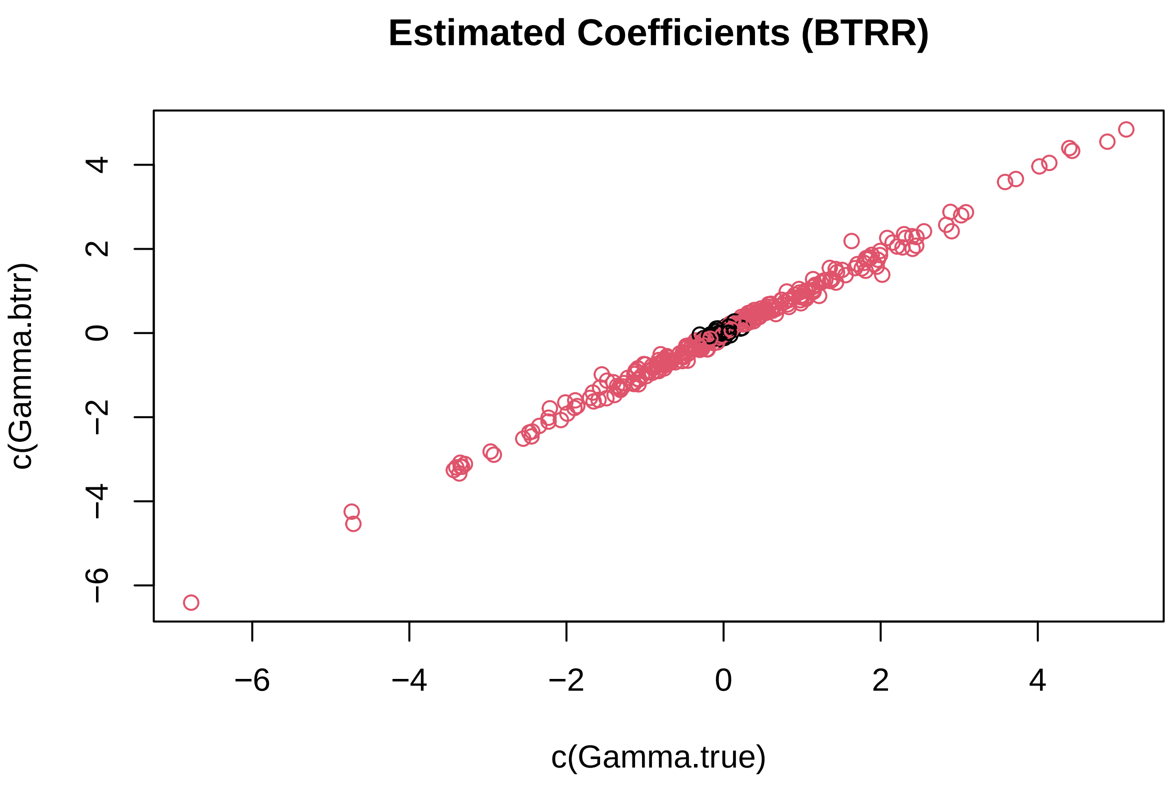


Get overall point and significance estimates (average over MCMC samples past burn in) from BTRR model

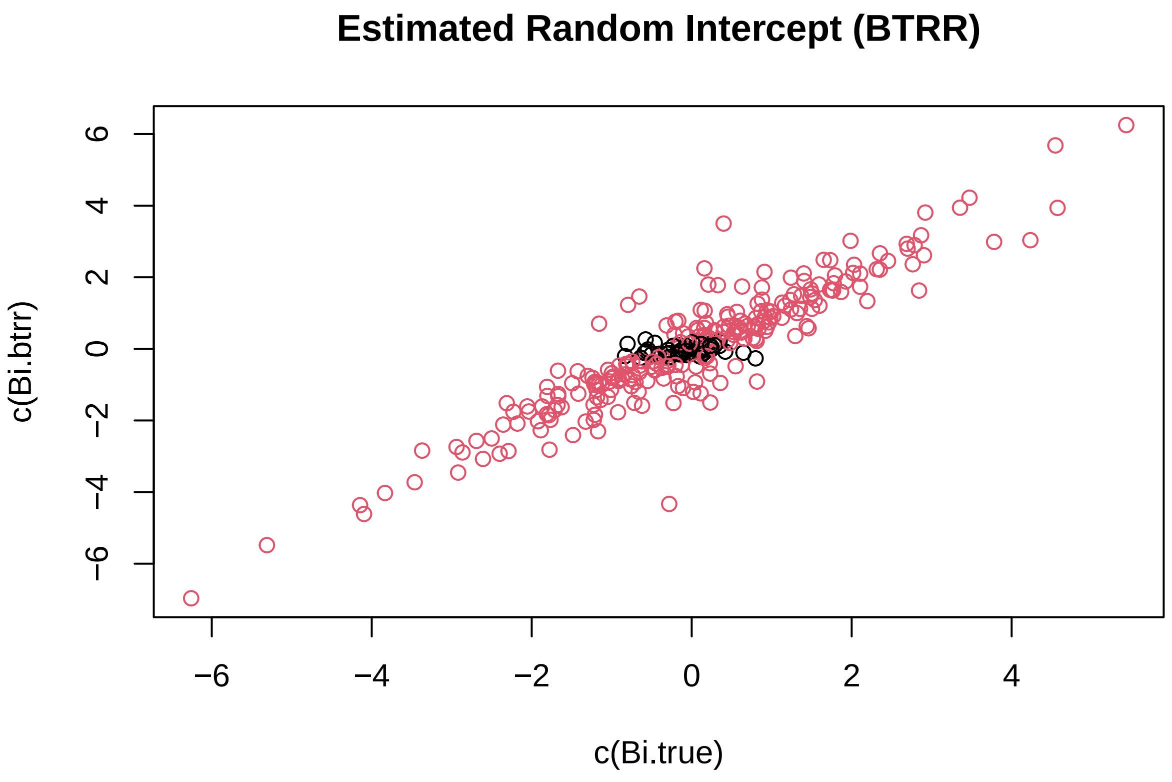
```
burn.in <- .3
Gamma.btrr <- apply(results[[1]][round(burn.in*nsweep):nsweep,,], c(2,3), mean)
Gamma.signif.btrr <- apply(results[[1]][round(burn.in*nsweep):nsweep,,], c(2,3), function(x) (quantile(

Bi.btrr <- apply(results[[7]][round(burn.in*nsweep):nsweep,,], c(2,3), mean)
Bi.signif.btrr <- apply(results[[7]][round(burn.in*nsweep):nsweep,,], c(2,3), function(x) (quantile(x,.
```

Plot true vs. estimated coefficients (BTRR)

```
plot(c(Gamma.true), c(Gamma.btrr), col=factor(Gamma.signif.btrr),main="Estimated Coefficients (BTRR)")
```



```r
plot(c(Bi.true), c(Bi.btrr), col=factor(Bi.signif.btrr), main="Estimated Random Intercept (BTRR)")
```



```r
paste0('Correlation for main coefficients (BTRR): ', signif(cor(c(Gamma.true), c(Gamma.btrr)),4))
```

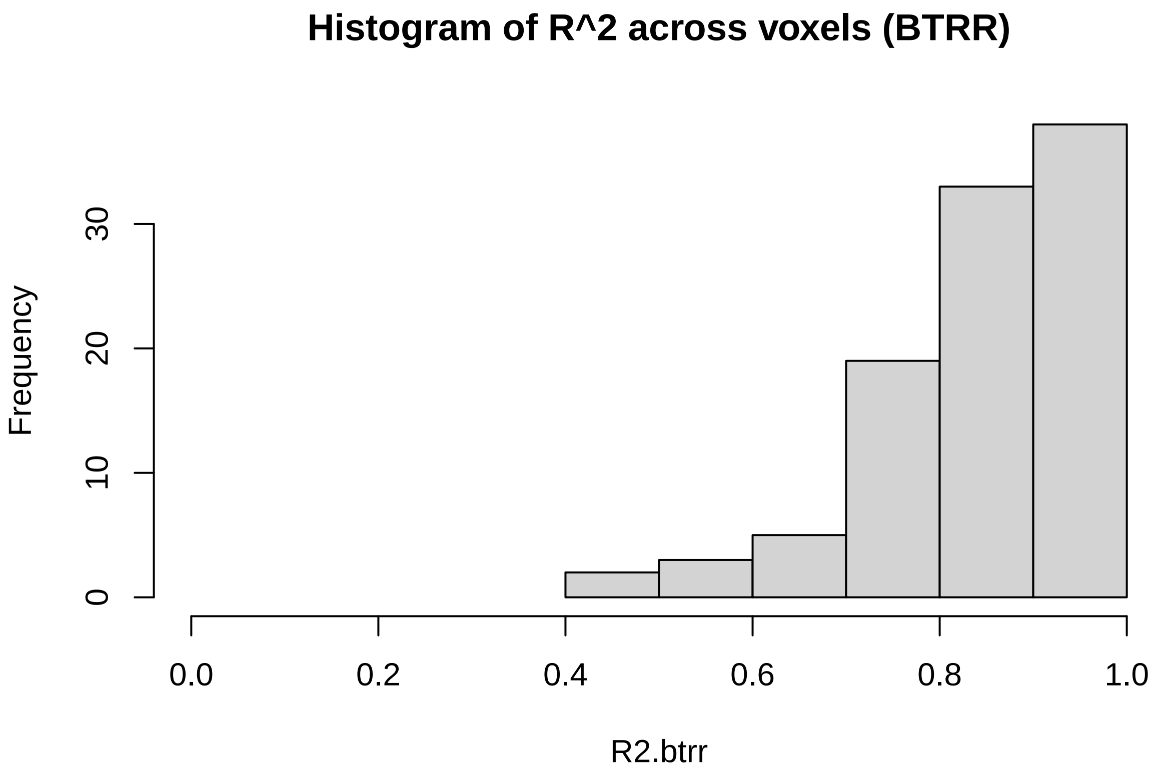
```
## [1] "Correlation for main coefficients (BTRR): 0.9965"
```

```
paste0('Correlation for random intercepts (BTRR): ', signif(cor(c(Bi.true), c(Bi.btrr)),4))
```

```
## [1] "Correlation for random intercepts (BTRR): 0.9209"
```

Find R^2 for each voxel (BTRR)

```
R2.btrr <- getR2(Y.train, X.train, ID, Gamma.btrr, Bi.btrr)
hist(R2.btrr,5,xlim=c(0,1),main='Histogram of R^2 across voxels (BTRR)')
```



```
paste0('Median R^2 across voxels (BTRR): ', signif(median(R2.btrr),4))
```

```
## [1] "Median R^2 across voxels (BTRR): 0.8759"
```

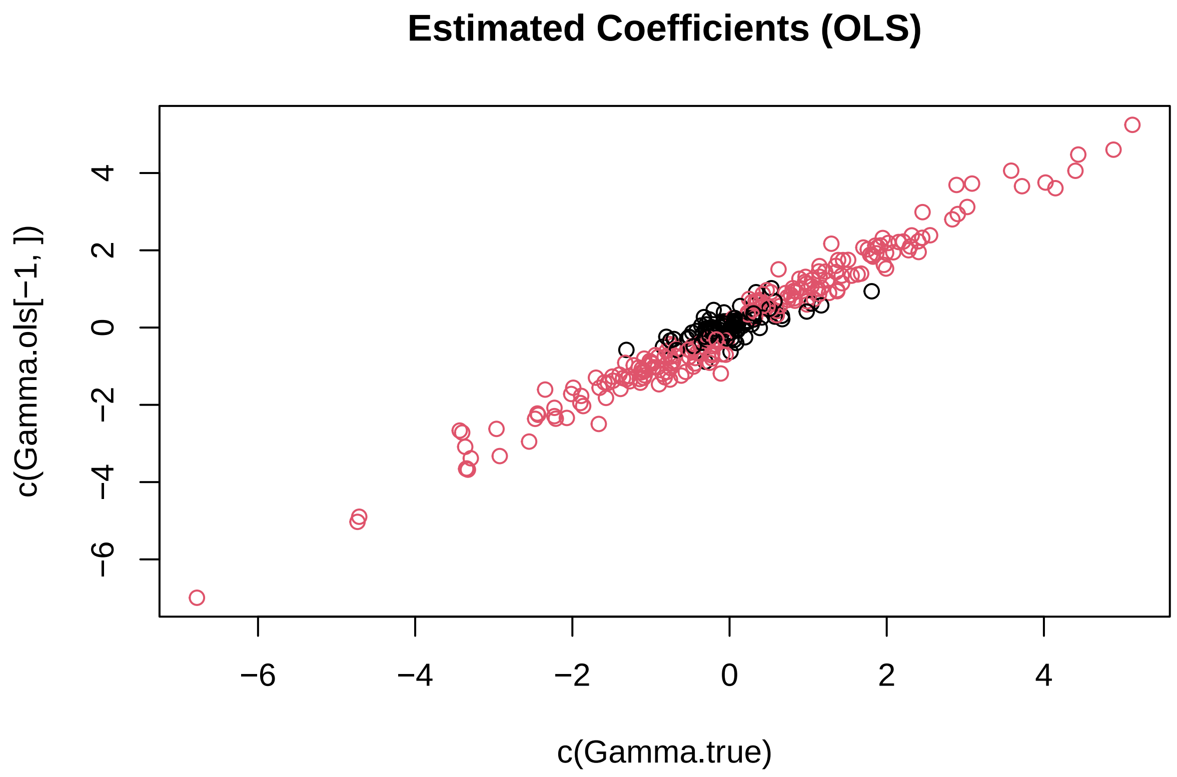
Fit model using OLS (not including random intercepts)

use cbind(1,X.train) to include overall intercept

```
ols_results <- vox.ols.reg(Y.train, cbind(1,X.train), find.signif=T)
Gamma.ols <- ols_results[[1]]
Gamma.signif.ols <- ols_results[[2]]
```

Plot estimated vs. true (OLS)

```
plot(c(Gamma.true), c(Gamma.ols[-1,]), col=factor(Gamma.signif.ols[-1,]),main="Estimated Coefficients (
```

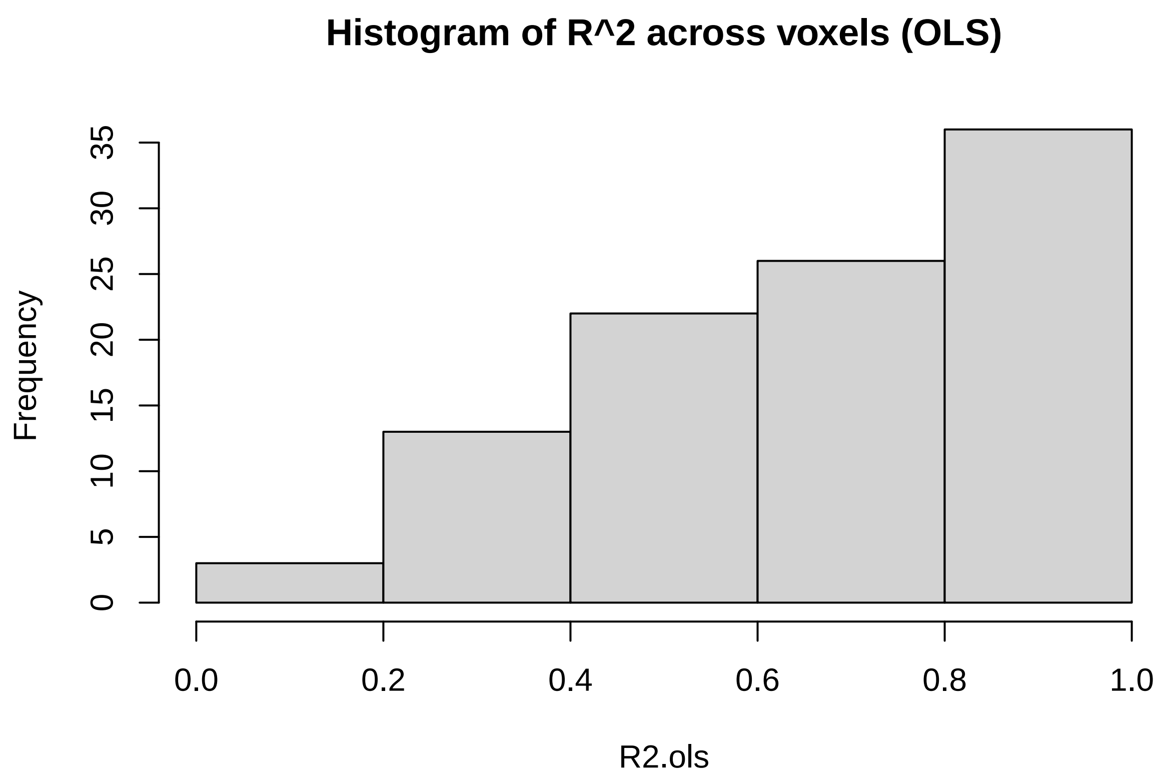


```
cor(c(Gamma.true), c(Gamma.ols[-1,]))
```

```
## [1] 0.9795621
```

Find R^2 for each voxel (OLS)

```
R2.ols <- getR2(Y.train, cbind(1,X.train), ID=NA, Gamma.ols)
hist(R2.ols,5,main='Histogram of R^2 across voxels (OLS)')
```



```
paste0('Median R^2 across voxels (OLS): ', signif(median(R2.ols),4))
```

```
## [1] "Median R^2 across voxels (OLS): 0.6933"
```