Testing Residual Simulations

Sarah Urbut

2016-07-14

Contents

0.1 Prerequisites

This is to understand how MASH performs in the setting of uncorrelated errors, where we assume the

$$E \sim N(0, V)$$

and V is not diagonal.

For now, you have to install the development versions of **bookdown** from Github:

```
devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter.

0.2 Introduction

The purpose of this is to test the new results with residual matrix coding, and to demonstrate that in simulationw with a known residual matrix, the likelihood is better using the correlated residuals than without.

The purpose of the document is thus three fold:

1) We show that using a variance matrix as an input to `compute.hm.log.lik.pen.vmat` (as opposed to `comput.hm.log.lik.pen` with a matrix of standard errors to be diagonalized) results in the same values when the matrix input the `var.mat` is simply the diagonalized vector of squared standard errors (and all genes have the same standard error in this simulation.)

2) We demonstrate this result under the EE and EZ model. In the EZ model, recall that the vector of standard erros is simply a vector of 1s, and accordingly the var.mat is the Identity matrix.

3) In the setting with a residual matrix in which the correlation in errors is 0.80, I demonstrate that under both the EE and EZ model, the likelihoods are better using the residual matrix input as opposed to the identiy matrix.

- We see that in the EZ model, this is just `cov2cor` of the known var.mat, because the t statistics are $\frac{\hat{\beta}}{s_j}$, and so the V.j of $\hat{\beta}$ must be scaled accordingly such that the element along the diagonal is 1 (i.e., divided s_{jr}^2/s_{jr}^2 and divide $cov(s_{jr}, s_{jr'})$ by $s_{jr}, s_{jr'}$
- Further, I show that it is very close to the empirical estimation of the covariance matrix of the null Z statistics. Likewise, the true var.mat of the betahats is very close to the empirical estimate of the covariance matric of the null betahats. This can be used for future testing to make sure you've uncovered the true residual matrix.

4) In my **Posteriors** section, I show that using the `compute.quant.per.snp` and `compute.quant.per.snp.with.vmat` code produce the same results when inputting v.mat as the squared diagonalized s.j.
5) Most importantly, I produce RMSE and ROC curve for both the EE and ET computed posteriors, using the diagonal and reisdual assumption, again in the setting where there exists strong correlation in the residduals. The results are perfect: when correlation existis, incorporating it is succesfully marked as both more accurate and powerful by MASH.

So there's no problem with my code!

0.3 Description of Simulation Framework

Here, I simulate some data in which the residuals are correlated.

Thus $\beta \sim N(0, Uk)$ and $\hat{\beta} \sim N(0, Uk + Vj)$

Such that $\hat{\beta} = \beta + E$ where $E \sim N(0, Vj)$

```
rm(list=ls())

library('mash')

sim.with.error=function(J,d=44,betasd=1,esd=0.11,n=400,rho=0.8){
   n=n
  covmat=readRDS(system.file('simdata/covmatforsimulation.rds', package = 'mash'))[2:9]
  covmat=lapply(seq(1:length(covmat)),function(x){covmat[[x]]/max(diag(covmat[[x]]))})



  library("mvtnorm")
  library("MASS")
  K=length(covmat)


  if(n!=0){
  z = sample(K,n,replace=TRUE)
  omega=abs(rnorm(n,mean=0,sd=betasd))##effect size variance can be big or small
  beta=t(sapply(seq(1:n),function(j){
    k=z[j]
    o=omega[j]
    mvrnorm(1,mu=rep(0,d),Sigma=o*covmat[[k]])
  }))
  beta=rbind(beta,matrix(rep(0,(J-n)*d),ncol=d))}
  if(n==0){
    beta=matrix(rep(0,(J-n)*d),ncol=d)
  }

  s.j.r=as.matrix(abs(rnorm(d,esd,0.001)))##simulate with the same standard error for every J
  v.j.r=s.j.r%*%t(s.j.r)##now v.j.r will be the same for every J
  v.mat=rho*v.j.r+(1-rho)*diag(diag(v.j.r))#make the errors correlated
  e=rmvnorm(J,mean=rep(0,d),sigma=v.mat)
  betahat = beta + e
  s.j=matrix(rep(s.j.r),nrow(betahat),byrow=T,ncol=d)
  t.stat=betahat/abs(s.j)
  if(n!=0){
    return(list(beta=beta,betahat=betahat,component.mats=covmat,sebetahat=s.j,t.stat=t.stat,component.i
  }
  if(n==0){
    return(list(beta=beta,betahat=betahat,sebetahat=s.j,t.stat=t.stat,error=e,var.mat=v.mat))
  }
}
```

You can see that in this simulation, the $V.j$ matrix for ever gene is the same for every j, and the standard errors for the tissues are all roughly equivalnet (centered aroun 0.10.)

I simulate 1000 values, 400 of which are `true` shared structured with $rho = 0.8$ in the correlation of residual matrix.

```
## $names
## [1] "beta"             "betahat"          "component.mats" "sebetahat"
## [5] "t.stat"           "component.id"     "error"          "var.mat"
## [9] "omega"
```

0.4 Testing Simulations

To test the simulations are accurate, make sure that the covariance of the nulls is the same as the covariance of the errors. This should be true because under the simulation framework above,

$\hat{\beta}_{null} = 0 + E$

```
(cov(s$betahat[401:1000,])[1:5,1:5])
```

```
##              V1          V2          V3          V4          V5
## V1 0.011537213 0.008843323 0.008783923 0.009307824 0.008813660
## V2 0.008843323 0.010796276 0.008371736 0.008835393 0.008388811
## V3 0.008783923 0.008371736 0.010829103 0.008890581 0.008411636
## V4 0.009307824 0.008835393 0.008890581 0.012011765 0.008812083
## V5 0.008813660 0.008388811 0.008411636 0.008812083 0.010608879
```

```
(cov(s$error[401:1000,])[1:5,1:5])
```

```
##               [,1]        [,2]        [,3]        [,4]        [,5]
## [1,] 0.011537213 0.008843323 0.008783923 0.009307824 0.008813660
## [2,] 0.008843323 0.010796276 0.008371736 0.008835393 0.008388811
## [3,] 0.008783923 0.008371736 0.010829103 0.008890581 0.008411636
## [4,] 0.009307824 0.008835393 0.008890581 0.012011765 0.008812083
## [5,] 0.008813660 0.008388811 0.008411636 0.008812083 0.010608879
```

You can see they are the same.

And most importantly, we want to make sure that this resembles the *true* residual matrix since for all j, $E \sim N(0, V)$

```
s$var.mat[1:5,1:5]
```

```
##               [,1]        [,2]        [,3]        [,4]        [,5]
## [1,] 0.012390065 0.009778049 0.009638546 0.009922640 0.009854220
## [2,] 0.009778049 0.012057322 0.009508241 0.009788494 0.009720999
## [3,] 0.009638546 0.009508241 0.011715734 0.009648842 0.009582310
## [4,] 0.009922640 0.009788494 0.009648842 0.012416548 0.009864746
## [5,] 0.009854220 0.009720999 0.009582310 0.009864746 0.012245907
```

0.5 Comparing with the old code

First, let's test that even though we have correlation of the error terms, using the old code which did not allow for a nondiagonal V_j matrix and the new code specificying a diagonal matrix results in the same answers. Remember that the old code simply took V_j as `diag(s_{j})`.

First, we test this under the EE model where we input a JxR matrix of $\hat{\beta}$ hats and their estimated standard errors into the old code.

For simplicity and because this isn't to compare our inference measures with others, I'm just going to use a list of the 8 simulated covariance matrices, so our likelihood matrix will only need to be $1000x8$.

Here with the new code, where we specify the same vmat for each gene:

```
covmat=s$component.mats
compute.hm.train.log.lik.pen.vmat(train.b = s$betahat[1:1000,],covmat=covmat,A = "testeewithdiagvmat",p
```

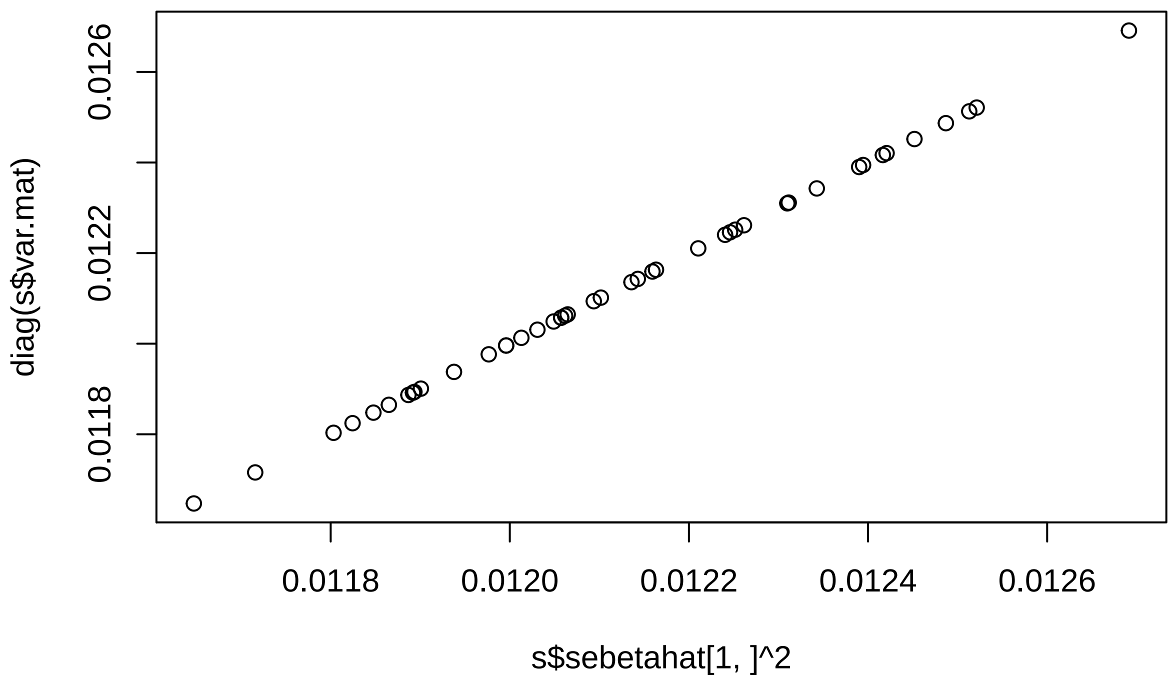
```
## pdf
##   2
```

```
lik.mat=readRDS("liketraintesteewithdiagvmat.rds")
pis=readRDS("pistesteewithdiagvmat.rds")$pihat
test=exp(lik.mat)
total.lik.func(test,pis)
```

```
## [1] 43563.78
```

We double check that `s.j^2` is accordingly the diagonal `s$var.mat`

```
plot(s$sebetahat[1,]^2,diag(s$var.mat))
```



Then we compute the likelihood using the old code. Remember that here, we input a matrix of standard errors. In this case the vector of simulated standard errors will be the same for each gene. I did this so I could be assured that the new code and old code performed the same when errors are assumed uncorrelated.

```
compute.hm.train.log.lik.pen(train.b = s$betahat[1:1000,],se.train = s$sebetahat,covmat = covmat,A="tes
```

```
## pdf
##   2
```

```
lik.mat=readRDS("liketraintestwitholdcode.rds")
pis=readRDS("pistestwitholdcode.rds")$pihat
test=exp(lik.mat)
total.lik.func(test,pis)
```

```
## [1] 43563.78
```

Great! Now let's make sure the likelihoods are better when we use the residual matrix. Here, we input the true variance matrix of the errors, `vmat=s$var.mat` and we show that there is a greatly improved likelihood when correlation among residuals exist!

```
covmat=s$component.mats
compute.hm.train.log.lik.pen.vmat(train.b = s$betahat[1:1000,],covmat=covmat,A = "testwithrealvmat",pen
```

```
## pdf
##   2
```

```
lik.mat=readRDS("liketraintestwithrealvmat.rds")
pis=readRDS("pistestwithrealvmat.rds")$pihat
test=exp(lik.mat)
total.lik.func(test,pis)
```

```
## [1] 60654.01
```

0.5.1 Testing Under ET Model

Great!! Now let's make sure this is true with the ET model.Recall that in the ET model:

$$\frac{\beta}{\hat{s}} \sim N(0, Uk)$$

Thus the input summary statistic is $t_j \sim N(0, V_j)$ where t_{jr} has standard error of 1.

First, we repeat the checks between the old and new code check to see they give the same result when the residuals are assumed to be uncorrelated (i.e., var mat is the Identity matrix)

```
##Here i just rescale the covariance matrices to be consistent with the fact that the true Z's are on o
covmat=lapply(s$component.mats,function(x){
  #median(abs(s$t.stat))^2*
  (1/0.11)^2*x})
compute.hm.train.log.lik.pen.vmat(train.b = s$t.stat[1:1000,],covmat=covmat,A = "testwithtdiag",pen = 1
```

```
## pdf
##   2
```

```
lik.mat=readRDS("liketraintestwithtdiag.rds")
pis=readRDS("pistestwithtdiag.rds")$pihat
test=exp(lik.mat)
total.lik.func(test,pis)
```

```
## [1] -53511.16
```

And now we repeat with the old code which required a JxR matrix of standard erros, so we need repeat

```
compute.hm.train.log.lik.pen(train.b = s$t.stat[1:1000,],se.train = s$sebetahat/s$sebetahat,covmat = co
```

```
## pdf
##   2
```

```
lik.mat=readRDS("liketraint.testwitholdcode.rds")
pis=readRDS("pist.testwitholdcode.rds")$pihat
test=exp(lik.mat)
total.lik.func(test,pis)
```

```
## [1] -53511.16
```

Good! And let's make sure the likelihood with the real vmat is better under the ET assumption. Recall that here, we can use `cov2cor` of the var mat so that the diagonals are 1 and every off diagonal element is divided by the sj_i, sj_r

```
compute.hm.train.log.lik.pen.vmat(train.b = s$t.stat[1:1000,],covmat=covmat,A = "test.tcov2cor",pen = 1
```

```
## pdf
##   2
```

Just to validate using the vmat as the `cov2cor` of the true var.mat, let's look at the actual empirical covariance of the null t statistics values:

```
cov(s$t.stat[401:1000,])[1:5,1:5]
```

```
##           V1        V2        V3        V4        V5
## V1 0.9311665 0.7235246 0.7290662 0.7504313 0.7155237
## V2 0.7235246 0.8954124 0.7043773 0.7221044 0.6903662
## V3 0.7290662 0.7043773 0.9243214 0.7371314 0.7022638
## V4 0.7504313 0.7221044 0.7371314 0.9673997 0.7146324
## V5 0.7155237 0.6903662 0.7022638 0.7146324 0.8663204
```

```
cov2cor(s$var.mat)[1:5,1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  1.0  0.8  0.8  0.8  0.8
## [2,]  0.8  1.0  0.8  0.8  0.8
## [3,]  0.8  0.8  1.0  0.8  0.8
## [4,]  0.8  0.8  0.8  1.0  0.8
## [5,]  0.8  0.8  0.8  0.8  1.0
```

And here we go. You'll note that **the likelihood in this case, where the residuals are highly structured, is much improved**!! Great!!!

```
lik.mat=readRDS("liketraintest.tcov2cor.rds")
pis=readRDS("pistest.tcov2cor.rds")$pihat
test=exp(lik.mat)
total.lik.func(test,pis)
```

```
## [1] -36472.73
```

0.5.2 Posteriors with EE

Now, let's also test our ability to compute posteriors under both methods. Again, let's use but EE and ET, first try with diagonalised standard errors to show that the results match our previous computations.

```
j=sample(100,1)
pis=readRDS("pistestwitholdcode.rds")
b.with.old.code=total.quant.per.snp(j = j,covmat = s$component.mats,b.gp.hat = s$betahat,se.gp.hat = s$

##and here was where I used the new code (i.e. with vmat), with the squared standard error on the diago
```

```r
pis=readRDS("pistesteewithdiagvmat.rds")
b.with.new.code=total.quant.per.snp.with.vmat(j = j,covmat = s$component.mats,b.gp.hat = s$betahat,var.

rbind(b.with.new.code$posterior.means,b.with.old.code$posterior.means)
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]     [,6]       [,7]
## [1,] 0.2015357 0.1483815 0.0746145 0.1589729 0.1128013 0.178949 -0.1012554
## [2,] 0.2015357 0.1483815 0.0746145 0.1589729 0.1128013 0.178949 -0.1012554
##             [,8]       [,9]      [,10]      [,11]      [,12]       [,13]
## [1,] -0.1142208 -0.1012226 -0.0982012 -0.1157122 -0.1238635 -0.07876207
## [2,] -0.1142208 -0.1012226 -0.0982012 -0.1157122 -0.1238635 -0.07876207
##             [,14]      [,15]       [,16]     [,17]      [,18]      [,19]
## [1,] -0.07819967 -0.1146526 -0.09919273 0.1289659 0.08317053 0.07451512
## [2,] -0.07819967 -0.1146526 -0.09919273 0.1289659 0.08317053 0.07451512
##            [,20]     [,21]     [,22]      [,23]     [,24]      [,25]
## [1,] 0.08833867 0.1273516 0.1021804 0.07117179 0.1455498 0.06213119
## [2,] 0.08833867 0.1273516 0.1021804 0.07117179 0.1455498 0.06213119
##            [,26]      [,27]    [,28]        [,29]     [,30]      [,31]
## [1,] 0.05349847 0.05732494 0.235553 -0.002808655 0.1685246 0.06964702
## [2,] 0.05349847 0.05732494 0.235553 -0.002808655 0.1685246 0.06964702
##           [,32]      [,33]      [,34]     [,35]      [,36]     [,37]
## [1,] 0.1021515 0.01624453 0.06777181 0.0568386 0.06029456 0.1002719
## [2,] 0.1021515 0.01624453 0.06777181 0.0568386 0.06029456 0.1002719
##           [,38]     [,39]      [,40]     [,41]      [,42]      [,43]
## [1,] 0.1767244 0.1137695 -0.0067579 0.1817134 0.06913085 0.05501313
## [2,] 0.1767244 0.1137695 -0.0067579 0.1817134 0.06913085 0.05501313
##           [,44]
## [1,] 0.5025378
## [2,] 0.5025378
```

```r
rbind(b.with.new.code$lfsr,b.with.old.code$lfsr)
```

```
##               [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] 1.777467e-13 4.019007e-14 4.629175e-11 8.267163e-10 1.021477e-10
## [2,] 1.778577e-13 4.030110e-14 4.629186e-11 8.267164e-10 1.021478e-10
##               [,6]         [,7]         [,8]         [,9]        [,10]
## [1,] 1.421126e-08 5.101713e-06 8.784653e-06 2.445589e-05 3.996607e-05
## [2,] 1.421126e-08 5.101713e-06 8.784653e-06 2.445589e-05 3.996607e-05
##              [,11]        [,12]        [,13]        [,14]        [,15]
## [1,] 8.066687e-06 3.485455e-06 1.751646e-05 1.540247e-05 5.488705e-06
## [2,] 8.066687e-06 3.485455e-06 1.751646e-05 1.540247e-05 5.488705e-06
##            [,16]        [,17]        [,18]        [,19]        [,20]
## [1,] 7.5505e-06 4.908296e-13 6.422345e-07 6.119513e-05 1.601397e-09
## [2,] 7.5505e-06 4.909406e-13 6.422345e-07 6.119513e-05 1.601397e-09
##              [,21]        [,22]        [,23]        [,24]        [,25]
## [1,] 2.442491e-14 1.595424e-09 6.909691e-05 3.825839e-09 1.427649e-07
## [2,] 2.453593e-14 1.595424e-09 6.909691e-05 3.825839e-09 1.427649e-07
##              [,26]        [,27]        [,28]        [,29]        [,30]
## [1,] 5.680655e-07 1.055955e-10 3.630429e-14 0.0006555118 1.492993e-09
## [2,] 5.680655e-07 1.055956e-10 3.641532e-14 0.0006555118 1.492993e-09
##              [,31]        [,32]        [,33]        [,34]        [,35]
## [1,] 4.363539e-09 4.239942e-13 9.004036e-05 4.288192e-11 5.699246e-05
## [2,] 4.363539e-09 4.241052e-13 9.004036e-05 4.288203e-11 5.699246e-05
##             [,36]        [,37]       [,38]        [,39]        [,40]
## [1,] 0.0001075988 1.687539e-14 3.83360e-13 6.195044e-14 0.0003229651
## [2,] 0.0001075988 1.698641e-14 3.83471e-13 6.206147e-14 0.0003229651
##             [,41]        [,42]        [,43]        [,44]
## [1,] 6.682632e-11 1.023028e-09 3.981323e-10 4.746182e-10
## [2,] 6.682643e-11 1.023029e-09 3.981324e-10 4.746183e-10
```

```
##and let's do the same thing for the t stat

j=sample(100,1)
pis=readRDS("pist.testwitholdcode.rds")
t.with.old.code=total.quant.per.snp(j = j,covmat = covmat,b.gp.hat = s$t.stat,se.gp.hat = s$sebetahat/s$

##and here was where I used the new code (i.e. with vmat), with the squared standard error on the diagon

pis=readRDS("pistestwithtdiag.rds")
t.with.new.code=total.quant.per.snp.with.vmat(j = j,covmat = covmat,b.gp.hat = s$t.stat,var.mat = diag(

rbind(t.with.new.code$posterior.means,t.with.old.code$posterior.means)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 24.43584 17.20741 12.81251 20.67968 14.06292 25.27323 7.589553
## [2,] 24.43584 17.20741 12.81251 20.67968 14.06292 25.27323 7.589553
##          [,8]     [,9]    [,10]    [,11]   [,12]    [,13]  [,14]    [,15]
## [1,] 9.841762 9.850467 11.34124 9.825364 8.82721 7.931264 8.0972 9.072362
## [2,] 9.841762 9.850467 11.34124 9.825364 8.82721 7.931264 8.0972 9.072362
##         [,16]    [,17]    [,18]    [,19]    [,20]    [,21]    [,22]
## [1,] 8.480551 17.50992 8.594155 19.36681 14.71005 15.98254 15.32113
## [2,] 8.480551 17.50992 8.594155 19.36681 14.71005 15.98254 15.32113
##         [,23]    [,24]    [,25]    [,26]    [,27]    [,28]    [,29]
## [1,] 16.28779 22.51058 15.07915 15.21326 8.417744 20.94597 18.09774
## [2,] 16.28779 22.51058 15.07915 15.21326 8.417744 20.94597 18.09774
##         [,30]    [,31]    [,32]    [,33]    [,34]    [,35]    [,36]
## [1,] 25.91101 11.30769 14.07423 10.65925 10.72332 14.83631 18.99049
## [2,] 25.91101 11.30769 14.07423 10.65925 10.72332 14.83631 18.99049
##         [,37]    [,38]  [,39]    [,40]    [,41]    [,42]    [,43]    [,44]
## [1,] 9.661218 11.07759 15.266 4.668178 24.44532 10.20505 9.342254 11.94294
## [2,] 9.661218 11.07759 15.266 4.668178 24.44532 10.20505 9.342254 11.94294
```

```
rbind(t.with.new.code$lfsr,t.with.old.code$lfsr)
```

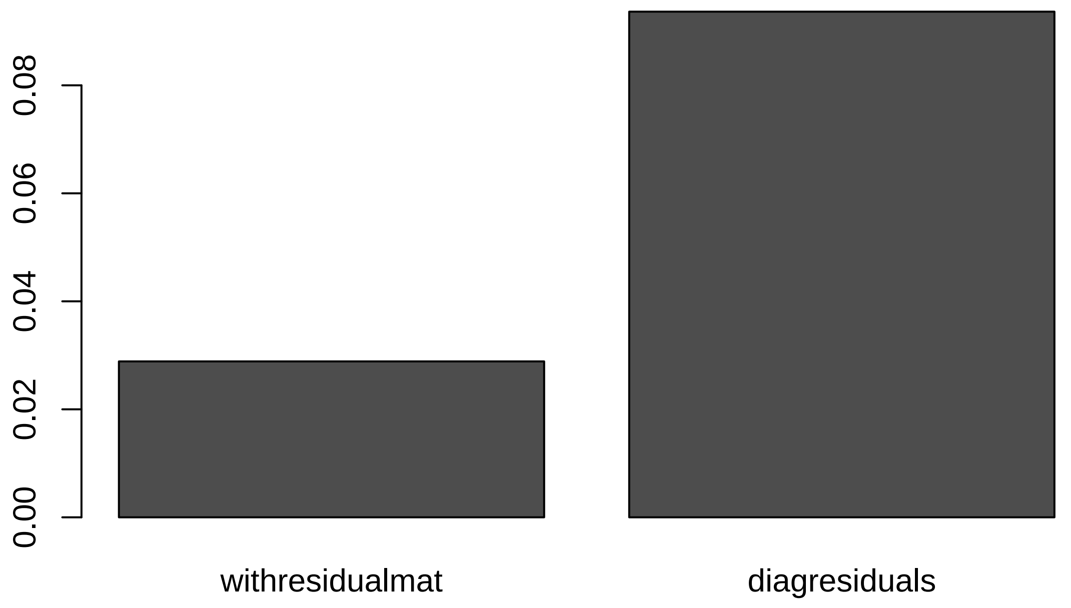
```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [2,]    0    0    0    0    0    0    0    0    0     0     0     0     0
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,]     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
## [1,]     0     0     0     0     0     0     0     0     0     0     0
## [2,]     0     0     0     0     0     0     0     0     0     0     0
##      [,36] [,37] [,38] [,39]        [,40] [,41] [,42] [,43] [,44]
## [1,]     0     0     0     0 1.060667e-06     0     0     0     0
## [2,]     0     0     0     0 1.060667e-06     0     0     0     0
```

And now, let's make sure the RMSE is better. We'll use the EE, and I first checked to make sure that the posterior results using the old code and the diagonalized vmat with the new code were the same.

```
covmat=s$component.mats
pis=readRDS("pistestwitholdcode.rds")$pihat
weightedquants=lapply(seq(1:nrow(s$betahat)),function(j){total.quant.per.snp(j,covmat,b.gp.hat=s$betaha

##check to make sure results same

pis=readRDS("pistesteewithdiagvmat.rds")$pihat
weightedquants=lapply(seq(1:nrow(s$betahat)),function(j){total.quant.per.snp(j,covmat,b.gp.hat=s$betaha

pis=readRDS("pistestwithrealvmat.rds")$pihat
weightedquants=lapply(seq(1:nrow(s$betahat)),function(j){total.quant.per.snp.with.vmat(j,covmat,b.gp.ha

post.means.var.mat=as.matrix(read.table("testwithrealvmatposterior.means.txt")[,-1])
post.means.diag.mat=as.matrix(read.table("eewithdiagvmatposterior.means.txt")[,-1])
beta=as.matrix(s$beta)
```

And the RMSE is so much better!!!

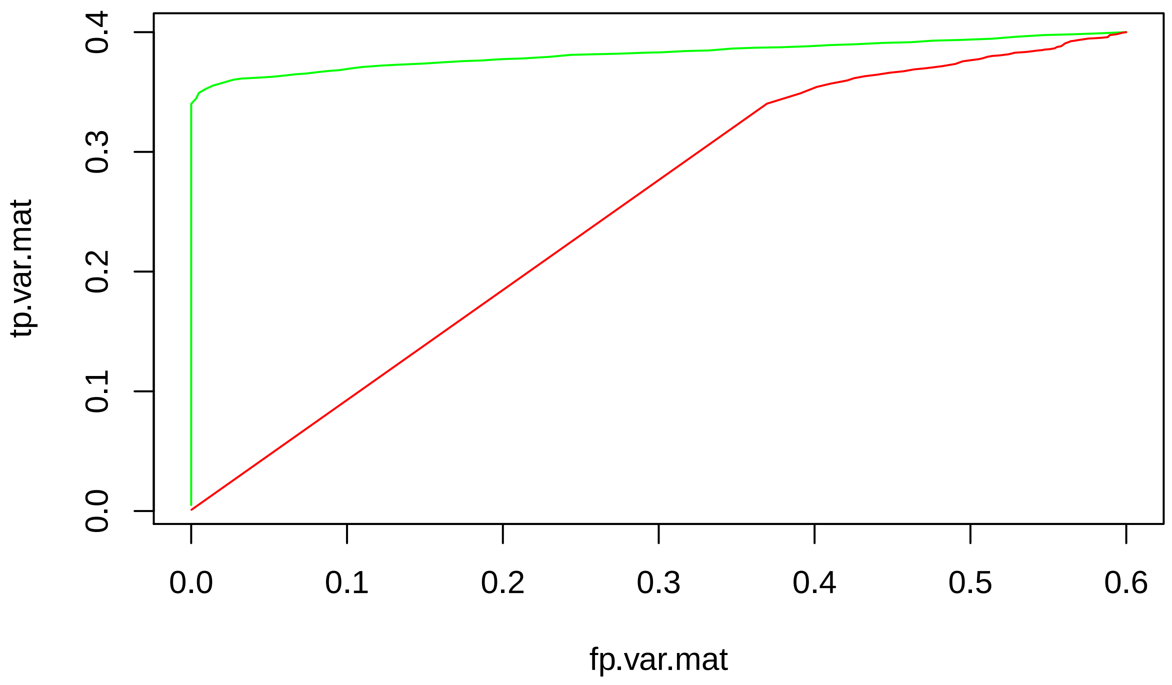
```
rmse.table=data.frame("withresidualmat"=sqrt(mean((beta[1:1000,]-post.means.var.mat[1:1000,])^2)),"diag
barplot(as.matrix(rmse.table),names=colnames(rmse.table))
```



As are the ROC curves:

```
lfsr.diag.mat=read.table("testwitholdcodelfsr.txt")[,-1]
lfsr.var.mat=read.table("testwithrealvmatlfsr.txt")[,-1]

thresh=seq(0,0.5,by=0.01)
fp.var.mat=NULL
fp.diag.mat=NULL
tp.var.mat=NULL
tp.diag.mat=NULL
```

```
for(t in 1:length(thresh)){
  sig=thresh[t]
  fp.var.mat[t]=mean(beta==0&lfsr.var.mat<sig)
  fp.diag.mat[t]=mean(beta==0&lfsr.diag.mat<sig)
  tp.var.mat[t]=mean(beta!=0&lfsr.var.mat<sig)
  tp.diag.mat[t]=mean(beta!=0&lfsr.diag.mat<sig)
}

plot(fp.var.mat,tp.var.mat,type="l",col="green")
lines(fp.diag.mat,tp.diag.mat,col="red")
```



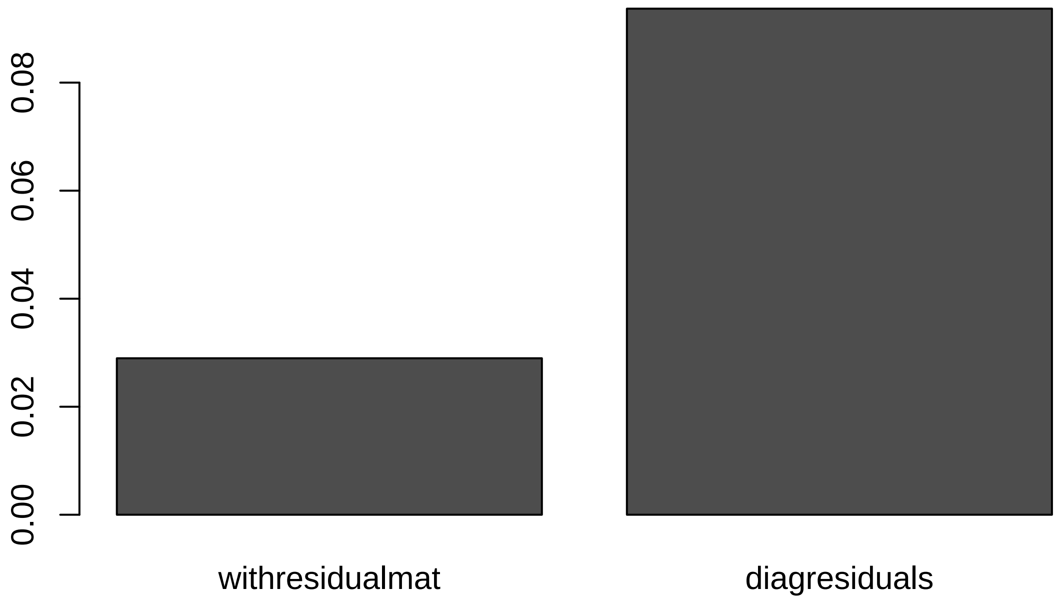
0.5.3 Posteriors With ET

And now, let's make sure the RMSE is better. We'll use the ET now, and repeat the process in which we compare the posteriors computed using the old cold and new code with diagonals, and then

```
covmat=lapply(s$component.mats,function(x){
  #median(abs(s$t.stat))^2*
  (1/0.11)^2*x})
pis=readRDS("pist.testwitholdcode.rds")$pihat
weightedquants=lapply(seq(1:nrow(s$betahat)),function(j){total.quant.per.snp(j,covmat,b.gp.hat=s$t.stat


pis=readRDS("pistest.tcov2cor.rds")$pihat
weightedquants=lapply(seq(1:nrow(s$betahat)),function(j){total.quant.per.snp.with.vmat(j,covmat,b.gp.ha


post.means.var.mat=as.matrix(read.table("ETwithrealvmatposterior.means.txt")[,-1])*as.matrix(s$sebetaha
post.means.diag.mat=as.matrix(read.table("EToldcodeposterior.means.txt")[,-1])*as.matrix(s$sebetahat)
beta=as.matrix(s$beta)
```

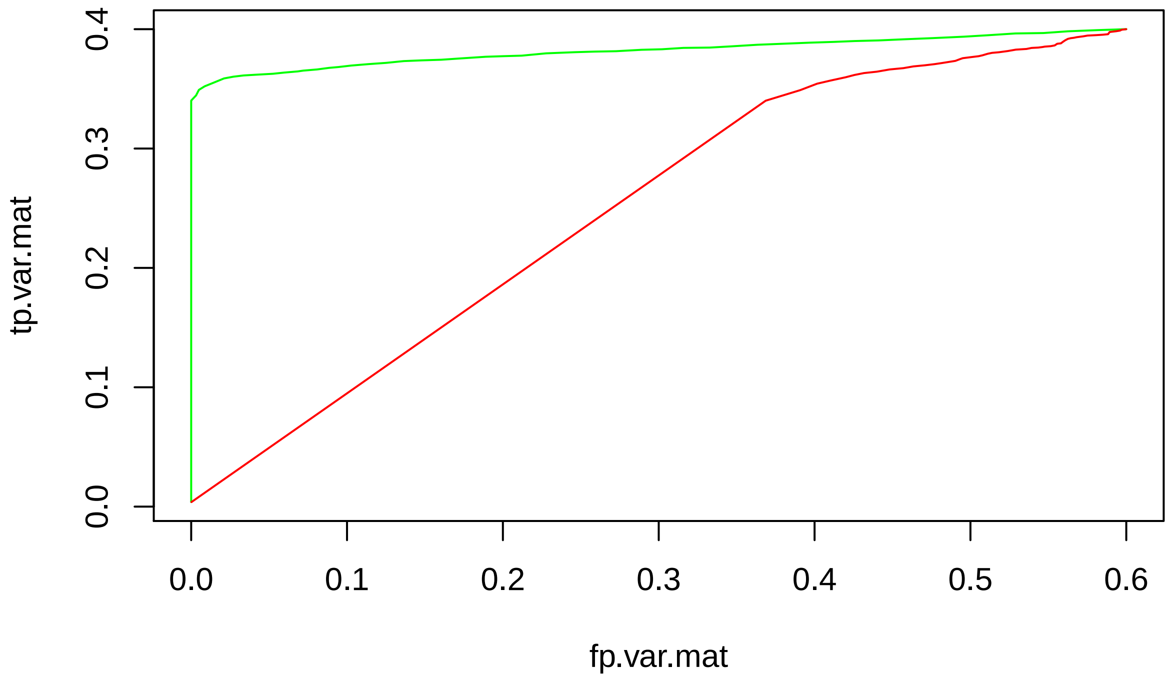
And the RMSE is so much better!!!

```
rmse.table=data.frame("withresidualmat"=sqrt(mean((beta[1:1000,]-post.means.var.mat[1:1000,])^2)),"diag
barplot(as.matrix(rmse.table),names=colnames(rmse.table))
```



As are the ROC curces:

```
lfsr.var.mat=read.table("ETwithrealvmatlfsr.txt")[,-1]
lfsr.diag.mat=read.table("EToldcodelfsr.txt")[,-1]

thresh=seq(0,0.5,by=0.01)
fp.var.mat=NULL
fp.diag.mat=NULL
tp.var.mat=NULL
tp.diag.mat=NULL
for(t in 1:length(thresh)){
  sig=thresh[t]
  fp.var.mat[t]=mean(beta==0&lfsr.var.mat<sig)
  fp.diag.mat[t]=mean(beta==0&lfsr.diag.mat<sig)
  tp.var.mat[t]=mean(beta!=0&lfsr.var.mat<sig)
  tp.diag.mat[t]=mean(beta!=0&lfsr.diag.mat<sig)
}

plot(fp.var.mat,tp.var.mat,type="l",col="green")
lines(fp.diag.mat,tp.diag.mat,col="red")
```


tp.var.mat
fp.var.mat
0.0
0.1
0.2
0.3
0.4
0.5
0.6

Bibliography