

BATools: An R Package for Whole Genomic Analysis with Bayesian Models

Chunyu Chen, Lei Zhou, Robert J. Tempelman

2015-10-06

The package **BATools** is used to perform genome-wide association using a various Bayesian models. It is implemented using both Markov Chain Monte Carlo (MCMC) and expectation maximization (EM) algorithm.

The basic functions in **BATools** is **bafit**, which fits a genomic selection model using different prior selection. The main characteristic of this package are:

- Fit model with different prior specification including the Antedependence model
- Flexibility to choose between MCMC and EM algorithm, both of which are able to estimate the hyperparameters
- Accepts **gpData** objects from package **synbreed** as input data. It can also use **numeric** and **matrix** as input
- It is computationally efficient
- GWA using EM BayesA/C (under development)

1. Introduction

Whole genome prediction (WGP) is an evolutionary development in animal breeding. Currently, many models have been developed for WGP, which included rrBLUP, BayesA, BayesB, BayesC, Bayesian Lasso, Antedependence BayesA/B (Meuwissen et al. 2001, VanRaden 2008, de los Campos et al. 2009, Habier et al. 2011, and Yang and Tempelman 2012). The major difference of these models are different prior assumptions on marker effects. Software packages like **BGLR** and **GenSel** implement BayesA, BayesB and Bayes Lasso model using MCMC algorithm. No public software is available to implement Antedependence models. At the same time, no R package is available for implement BayesA/C using EM algorithm for animal breeding. **BATools** package provides tools to fit Antedependence models in addition to some of the most popular models and provider faster EM algorithms to fit the model. The table below is a comparison between **BATools** and **BGLR**:

Model/Algorithms	MCMC	EM
rrBLUP	BATools/BGLR	BATools
BayesA	BATools/BGLR	BATools
BayesB	BATools/BGLR	under development
BayesC	BATools/BGLR	BATools
Bayesian Lasso	BATools/BGLR	under development
AnteBayesA	BATools	under development
AnteBayesB	BATools	under development

2. Basic Model

The basic model used by **BATools** is:

$$\mathbf{y} = \mathbf{X} \cdot \mathbf{b} + \mathbf{Z} \cdot \mathbf{g} + \mathbf{e},$$

where:

- \mathbf{y} is the vector of response variables
- $\mathbf{X} \cdot \mathbf{b}$ models the fixed effects
- \mathbf{g} is the SNP marker effect and \mathbf{Z} is corresponding genotype matrix of $n \cdot m$
- \mathbf{e} are the vector of effects residual, $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$

Notice that for different models, the priors on \mathbf{g}_i are different:

- rrBLUP: $\mathbf{g}_j \sim N(\mathbf{0}, \mathbf{I}\sigma_g^2)$
- BayesA: $\mathbf{g}_j \sim N(\mathbf{0}, \mathbf{D}\sigma_g^2)$, where $\mathbf{D} = \{\tau_1, \tau_2, \dots, \tau_m\}$ and $\tau_j \sim \chi^{-2}(\nu_g, \nu_g)$
- BayesB: $\mathbf{g}_j \sim N(\mathbf{0}, \mathbf{D}\sigma_g^2)$, where $\mathbf{D} = \{\tau_1, \tau_2, \dots, \tau_m\}$ and

$$\tau_j = \begin{cases} 0 & \text{with probability } \pi \\ \sim \chi^{-2}(\nu_g, \nu_g) & \text{with probability } 1 - \pi \end{cases}$$

- BayesC: $\mathbf{g}_i \sim N(\mathbf{0}, \mathbf{D}\sigma_g^2)$, where $\mathbf{D} = \{\tau_1 + \frac{1-\tau_1}{c}, \tau_2 + \frac{1-\tau_2}{c}, \dots, \tau_m + \frac{1-\tau_m}{c}\}$ and $\tau_j \sim \text{Bernoulli}(\pi)$, $\tau_j = 0, 1$
- Bayesian Lasso: $\mathbf{g}_j \sim N(\mathbf{0}, \mathbf{D}\sigma_g^2)$, where $\mathbf{D} = \{\tau_1, \tau_2, \dots, \tau_m\}$ and $\tau_j \sim \text{Exp}(\lambda^2)$

Furthermore, the Antedepedence models specify correlation structure for \mathbf{g} based on the relative physical location of SNP markers along the chromosome :

$$g_j = \begin{cases} \delta_j & \text{if } j = 1 \\ t_{j,j-1}\delta_{j-1} + \delta_j & \text{if } 2 \leq j \leq m \end{cases}$$

where $t_{j,j-1} \sim N(\mu_t, \sigma_t^2)$

3. BATools example

In **BATools**, we adhered the data structure of the object **gpData** in the **synbreed** package. The input and output objects are named as **baData** and **BAout**, which are R object class **list**. Therefore, users can directly use **synbreed** object as the input for **BATools**, and vice versa. More detailed explanation about **baData** and **BAout** can be found in the package manual file.

Example

We will use a toy dataset from the MSUPRP population to illustrate the use of **BATools**

Load packages and data

```
library(BATools)
data("MSUPRP_sample")
summary(MSUPRP_sample)

## object of class 'gpData'
## covar
##   No. of individuals 253
##           phenotyped 176
##           genotyped 251
## pheno
##   No. of traits:      3
##
##           ph_24h      temp_24h      driploss
##   Min.    :5.190    Min.    :1.100    Min.    :0.000
##   1st Qu.:5.450    1st Qu.:1.600    1st Qu.:0.560
##   Median :5.540    Median :1.900    Median :0.940
##   Mean   :5.552    Mean   :1.983    Mean   :1.141
##   3rd Qu.:5.640    3rd Qu.:2.300    3rd Qu.:1.545
##   Max.    :6.350    Max.    :3.400    Max.    :4.330
##   NA's    :35      NA's    :18      NA's    :18
##
## geno
##   No. of markers 20597
##   genotypes 0 1 2
##   frequencies 0.287 0.404 0.308
##   NA's 0.000 %
## map
##   No. of mapped markers 20597
##   No. of chromosomes    18
```

```
##
## markers per chromosome
##
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
## 1968 1419 1261 1559 1076 1143 1289 939 1426 958 936 833 1319 1263 1079
## 16 17 18
## 972 739 418
##
## pedigree
## Number of
## individuals 253
## males : 98 , females : 155
## Par 1 (sire) 10
## Par 2 (dam) 44
## generations 3
```

Then we can create the data object used in BATools by `create.baData`. In this example we treat the sex as fixed effects

```
pheno<-data.frame(MSUPRP_sample$pheno[,])
geno<-MSUPRP_sample$geno[,seq(1,dim(MSUPRP_sample$geno)[2],20)] #[,1:500]
ped<-MSUPRP_sample$pedigree
map=MSUPRP_sample$map
sex<-ped$sex
sex<-as.factor(sex)
x<-model.matrix( ~ sex -1,contrasts.arg=list(sex=contrasts(sex, contrasts=F)))
colnames(x)<-c("female","male")
rownames(x)<-ped$ID
pig=create.baData(pheno=pheno,geno=geno,map=map,pedigree=ped,fixed=x,makeAinv=F)
```

Set up initial values for the model

We choose to demonstrate how to fit BayesA using MCMC and EM. We start with MCMC:

```
init=set_init(pig,df=5,pi_snp=1,h2=0.5,c=NULL,model="BayesA",centered=T,trait="driploss")
run_para=list(niter=5000,burnIn=2500,skip=10)
print_mcmc=list(piter=500)
update_para=list(df=FALSE,scale=TRUE,pi=FALSE)
op<-create.options(model="BayesA",method="MCMC",ante=FALSE,priors=NULL,init=init,
  update_para=update_para,run_para=run_para,save.at="BayesA",cv=NULL,print_mcmc=print_mcmc)
```

Fit the model

We then fit the model using MCMC for the trait `driploss` with the above setups:

```
ba<-bafit(dataobj=pig,op=op,trait="driploss")

## iter= 500  vare= 0.432924 scale= 0.0004069
## timepercycle= 0.001 estimated time left= 3.12 seconds
## iter= 1000  vare= 0.366249 scale= 0.00056412
## timepercycle= 0.001 estimated time left= 2.78 seconds
## iter= 1500  vare= 0.385861 scale= 0.00039244
## timepercycle= 0.001 estimated time left= 2.46 seconds
## iter= 2000  vare= 0.358691 scale= 0.000336
## timepercycle= 0.001 estimated time left= 2.09 seconds
## iter= 2500  vare= 0.363131 scale= 0.00049178
## timepercycle= 0.001 estimated time left= 1.74 seconds
## iter= 3000  vare= 0.348773 scale= 0.00034089
## timepercycle= 0.001 estimated time left= 1.43 seconds
## iter= 3500  vare= 0.327509 scale= 0.00025024
## timepercycle= 0.001 estimated time left= 1.08 seconds
## iter= 4000  vare= 0.403936 scale= 0.00021155
## timepercycle= 0.001 estimated time left= 0.73 seconds
## iter= 4500  vare= 0.307009 scale= 0.00049487
## timepercycle= 0.001 estimated time left= 0.37 seconds
## iter= 5000  vare= 0.36142 scale= 0.00037677
## timepercycle= 0.001 estimated time left= 0 seconds

ba

## BATools analysis of trait: driploss
##
## estimated fixed effects:
##   female      male
## 1.0076589 0.8685423
##
## estimated hyperparameters:
##      vare      scale
## 0.3807336238 0.0003754588
##
## effective sample size for hyperparameters:
##      vare      scale
## 99.05445 14.79361
```

Graphics

We can obtain the traceplot for MCMC:

```
par(mar=c(2,2,2,2))  
baplot(dataobj=pig,BAout=ba,type="trace",op=op)
```

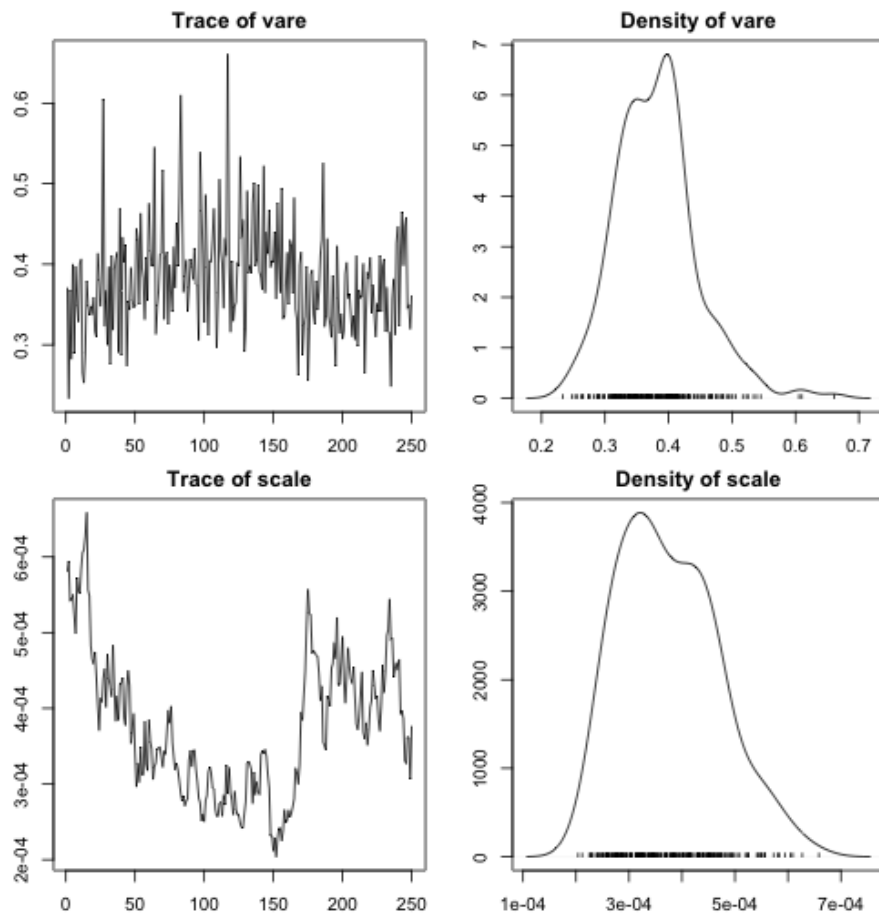


Figure 1: Traceplot

EM algorithm

To use the EM algorithm in BATools, we first run an analysis using rrBLUP:

```
#####run rrBLUP REML#####  
init=set_init(pig,df=5,scale=NULL,vare=NULL,pi_snp=1,h2=0.5,c=NULL,model="rrBLUP",
```

```

        centered=T,trait="driploss")
run_para=list(maxiter=100)
update_para=list(df=FALSE,scale=TRUE)
op<-create.options(model="rrBLUP",method="EM",ante=FALSE,priors=NULL,init=init,
update_para=update_para,run_para=run_para,save.at="rrBLUP",cv=NULL,print_mcmc=NULL,convcrit=

rr<-bafit(dataobj=pig,op=op,trait="driploss")

## rrBLUP iter= 1
## Residual Variance is 0.3916505 Scale is 0.0004661316
## Convergence criteria is 1e-04 and current value is 1e+10
## rrBLUP iter= 2
## Residual Variance is 0.3884433 Scale is 0.0005341798
## Convergence criteria is 1e-04 and current value is 0.008258348
## rrBLUP iter= 3
## Residual Variance is 0.3900734 Scale is 0.0005310876
## Convergence criteria is 1e-04 and current value is 0.004179042
## rrBLUP iter= 4
## Residual Variance is 0.3899009 Scale is 0.0005317811
## Convergence criteria is 1e-04 and current value is 0.0004426353
## rrBLUP iter= 5
## Residual Variance is 0.3899315 Scale is 0.00053166
## Convergence criteria is 1e-04 and current value is 7.85073e-05
##
## rrBLUP converged after 5 iterations at 7.85073e-05

rr

## BATools analysis of trait: driploss
##
## estimated fixed effects:
##   female    male
## 1.170468 1.040369
##
## SD
##   female    male
## 1.302095 1.303128
##
## estimated hyperparameters:
##       vare       scale
## 0.38993150 0.00053166

```

Then we use rrBLUP results as starting values for EM BayesA:

```

init=set_init(pig,df=5,scale=rr$hyper_est[2],vare=rr$hyper_est[1],g=rr$ghat,b=rr$betahat,
pi_snp=1,h2=0.5,model="BayesA",centered=T,trait="driploss",from="rrBLUP")

```

```

run_para=list(maxiter=100)
update_para=list(df=FALSE,scale=TRUE,pi=FALSE)
op<-create.options(model="BayesA",method="EM",ante=FALSE,priors=NULL,init=init,D="V",
  update_para=update_para,run_para=run_para,save.at="BayesA",cv=NULL,print_mcmc=NULL)
ba_em<-bafit(dataobj=pig,op=op,trait="driploss")

```

```

## BayesA EM iter= 1
## Residual Variance is 0.3863244 Scale is 0.0005531357
## Convergence criteria is 1e-04 and current value is 1e+10
## BayesA EM iter= 2
## Residual Variance is 0.3848432 Scale is 0.0006469239
## Convergence criteria is 1e-04 and current value is 0.003856724
## BayesA EM iter= 3
## Residual Variance is 0.3837225 Scale is 0.0006529888
## Convergence criteria is 1e-04 and current value is 0.002920448
## BayesA EM iter= 4
## Residual Variance is 0.3820295 Scale is 0.0006576012
## Convergence criteria is 1e-04 and current value is 0.004441596
## BayesA EM iter= 5
## Residual Variance is 0.3818452 Scale is 0.0006579109
## Convergence criteria is 1e-04 and current value is 0.0004826109
## BayesA EM iter= 6
## Residual Variance is 0.3816379 Scale is 0.0006585268
## Convergence criteria is 1e-04 and current value is 0.0005430623
## BayesA EM iter= 7
## Residual Variance is 0.3816101 Scale is 0.000658589
## Convergence criteria is 1e-04 and current value is 7.292798e-05
##
## BayesA converged after 7 iterations at 7.292798e-05

```

```

ba_em

```

```

## BATools analysis of trait: driploss
##
## estimated fixed effects:
##   female      male
## 1.168412 1.036599
##
## SD
##   female      male
## 1.340235 1.341329
##
## estimated hyperparameters:
##           vare           scale
## 0.381610104 0.000658589

```


Graphics

Let's look at the estimated phenotypes v.s. true phenotypes for EM:

We can also compare the difference between MCMC and EM:

```
plot(ba$ghat,ba_em$ghat,xlab="MCMC",ylab="EM",main="BayesA MCMC v.s. EM")  
abline(a=0,b=1)
```

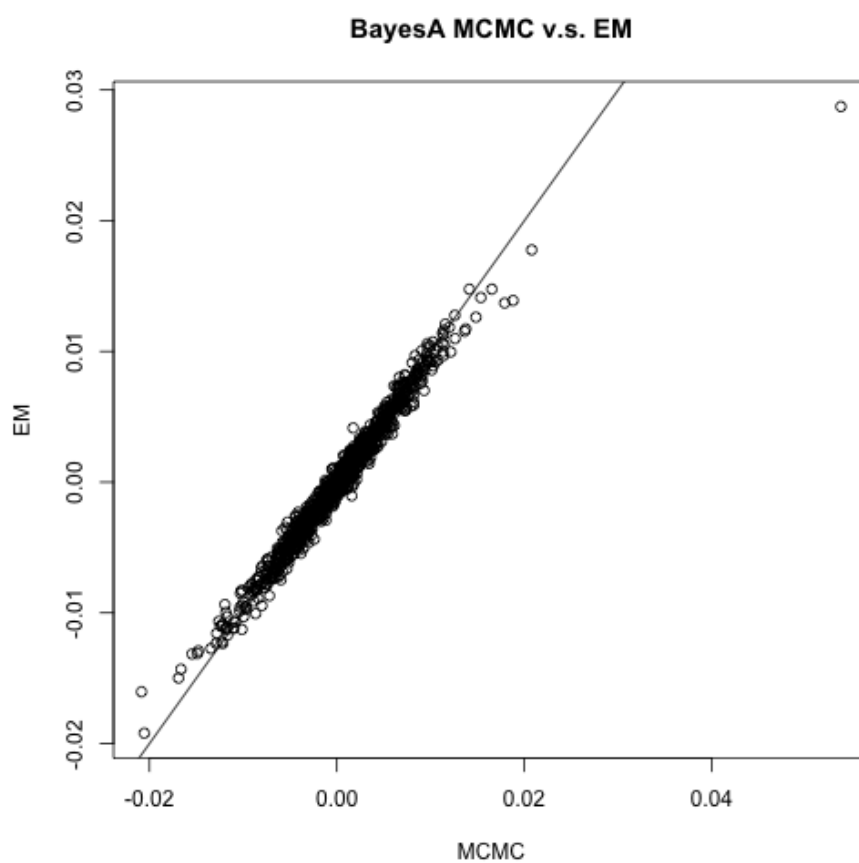


Figure 2:

BayesC

Running BayesC is similar to running BayesA:

```

init=list(df=5,scale=0.2,pi=0.1,c=1000)
run_para=list(niter=2000,burnIn=1000,skip=1)
print_mcmc=list(piter=200)
update_para=list(df=F,scale=T,pi=T)
priors=list(shape_scale=5,rate_scale=0.1,alphapi=1,betapi=9)
op<-create.options(model="SSVS",method="MCMC",
ante=FALSE,priors=NULL,init=init,update_para=update_para,
run_para=run_para,save.at="SSVS",cv=NULL,print_mcmc=print_mcmc)
bc<-bafit(dataobj=pig,op=op,trait="driploss")

## iter= 200  vare= 0.536829 scale= 0.29439797
## timepercycle= 0.001 estimated time left= 1.66 seconds
## iter= 400  vare= 0.595115 scale= 0.23640913
## timepercycle= 0.001 estimated time left= 1.45 seconds
## iter= 600  vare= 0.437507 scale= 0.29226811
## timepercycle= 0.001 estimated time left= 1.24 seconds
## iter= 800  vare= 0.479568 scale= 0.57186156
## timepercycle= 0.001 estimated time left= 1.07 seconds
## iter= 1000  vare= 0.36131 scale= 0.65779683
## timepercycle= 0.001 estimated time left= 0.89 seconds
## iter= 1200  vare= 0.423843 scale= 0.52539915
## timepercycle= 0.001 estimated time left= 0.73 seconds
## iter= 1400  vare= 0.456333 scale= 0.69004647
## timepercycle= 0.001 estimated time left= 0.55 seconds
## iter= 1600  vare= 0.390411 scale= 0.50739371
## timepercycle= 0.001 estimated time left= 0.37 seconds
## iter= 1800  vare= 0.373935 scale= 0.32093542
## timepercycle= 0.001 estimated time left= 0.19 seconds
## iter= 2000  vare= 0.470413 scale= 0.20767048
## timepercycle= 0.001 estimated time left= 0 seconds

bc

## BATools analysis of trait: driploss
##
## estimated fixed effects:
##   female   male
## 1.665812 1.530076
##
## estimated hyperparameters:
##           vare           scale           pi
## 0.410472946 0.452772665 0.001962844
##
## effective sample size for hyperparameters:
##           vare           scale           pi
## 130.06907 17.07845 98.97089

```

```

init=set_init(pig,df=5,scale=rr$hyper_est[2],vare=rr$hyper_est[1],g=rr$ghat,beta=rr$betahat,
              pi_snp=0.05,h2=0.5,c=1000,model="SSVS",centered=T,trait="driploss",from="rrBLU")
run_para=list(maxiter=100)
update_para=list(df=FALSE,scale=TRUE,pi=T)
op<-create.options(model="SSVS",method="EM",ante=FALSE,priors=NULL,init=init,
update_para=update_para,run_para=run_para,save.at="SSVS",cv=NULL,print_mcmc=NULL,convcrit=1e-05)
bc_em<-bafit(dataobj=pig,op=op,trait="driploss")

```

```

## SSVS EM iter= 1
## Residual Variance is 0.435391 Scale is 0.02920585
## Convergence criteria is 1e-04 and current value is 1e+10
## SSVS EM iter= 2
## Residual Variance is 0.4393395 Scale is 0.06430095
## Convergence criteria is 1e-04 and current value is 0.07953811
## SSVS EM iter= 3
## Residual Variance is 0.4070716 Scale is 0.1329879
## Convergence criteria is 1e-04 and current value is 0.1772093
## SSVS EM iter= 4
## Residual Variance is 0.3837358 Scale is 0.2655403
## Convergence criteria is 1e-04 and current value is 0.1775111
## SSVS EM iter= 5
## Residual Variance is 0.3783207 Scale is 0.2394833
## Convergence criteria is 1e-04 and current value is 0.05943913
## SSVS EM iter= 6
## Residual Variance is 0.377587 Scale is 0.2433619
## Convergence criteria is 1e-04 and current value is 0.008787261
## SSVS EM iter= 7
## Residual Variance is 0.377706 Scale is 0.2429965
## Convergence criteria is 1e-04 and current value is 0.0008555282
## SSVS EM iter= 8
## Residual Variance is 0.3776931 Scale is 0.2430379
## Convergence criteria is 1e-04 and current value is 9.64421e-05
##
## SSVS converged after 8 iterations at 9.64421e-05

```

bc_em

```

## BATools analysis of trait: driploss
##
## estimated fixed effects:
##   female      male
## 1.2033943 0.9734142
##
## SD
##   female      male

```

```
## 0.9368899 0.9366963
##
## estimated hyperparameters:
##      vare      scale      pi
## 0.377693117 0.243037886 0.002985153
```

We can also compare the difference bewteen MCMC and EM for BayesC:

```
plot(bc$ghat,bc_em$ghat,xlab="MCMC",ylab="EM",main="BayesC MCMC v.s. EM")
abline(a=0,b=1)
```

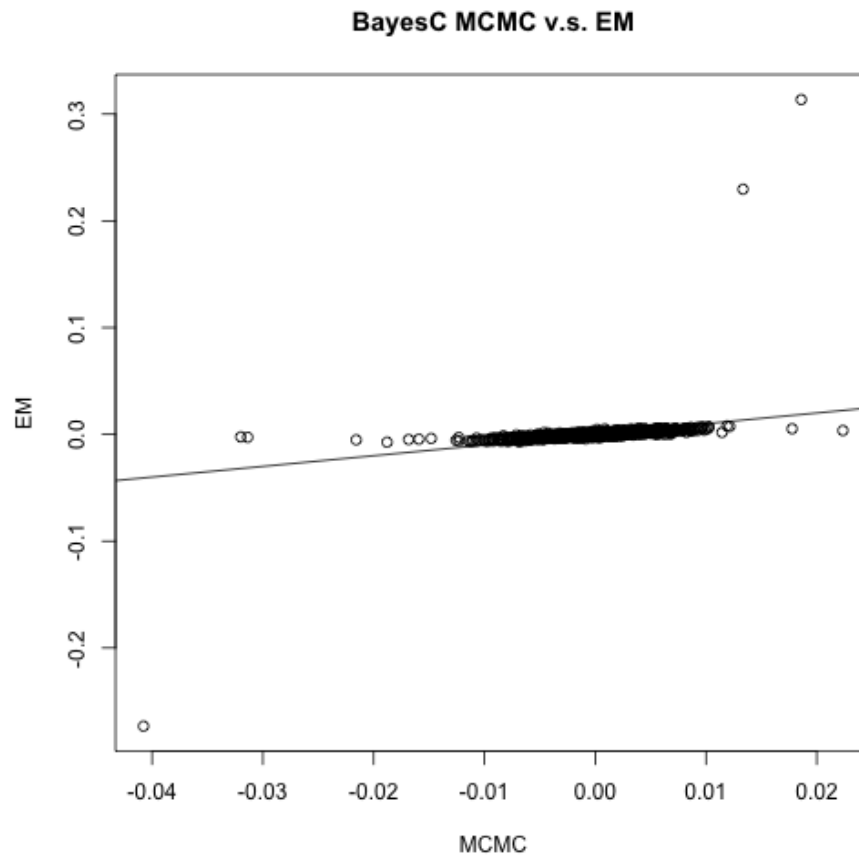


Figure 3: BayesC

We can also compare the difference bewteen BayesA and BayesC for MCMC:

```
plot(ba$ghat,bc$ghat,xlab="BayesA",ylab="BayesC",main="BayesA v.s. BayesC in MCMC")
abline(a=0,b=1)
```

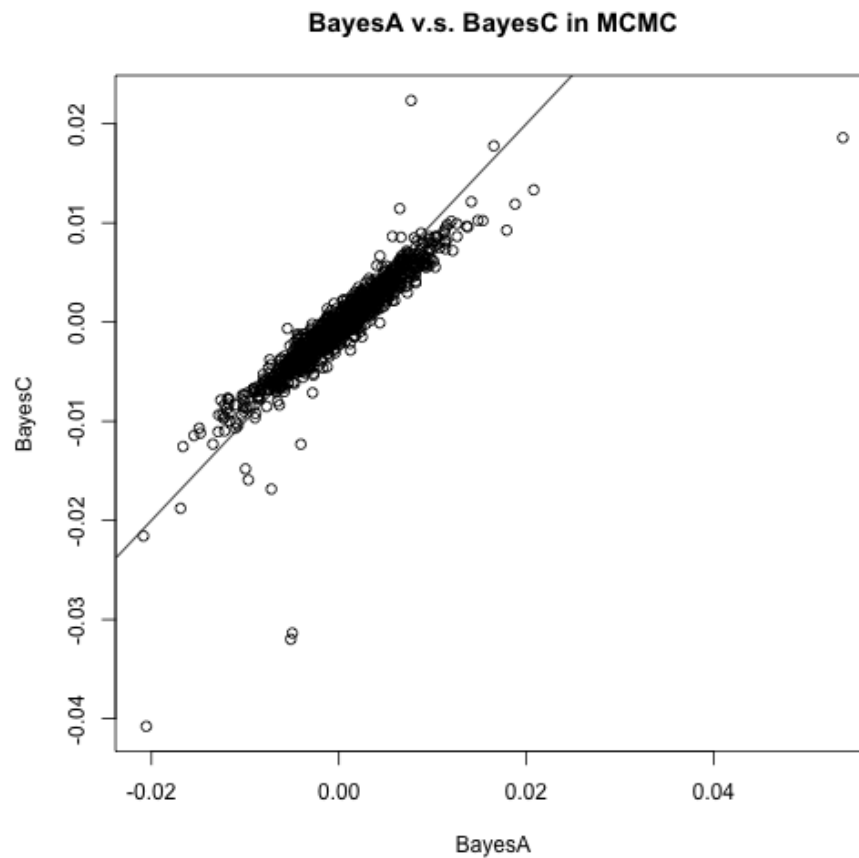
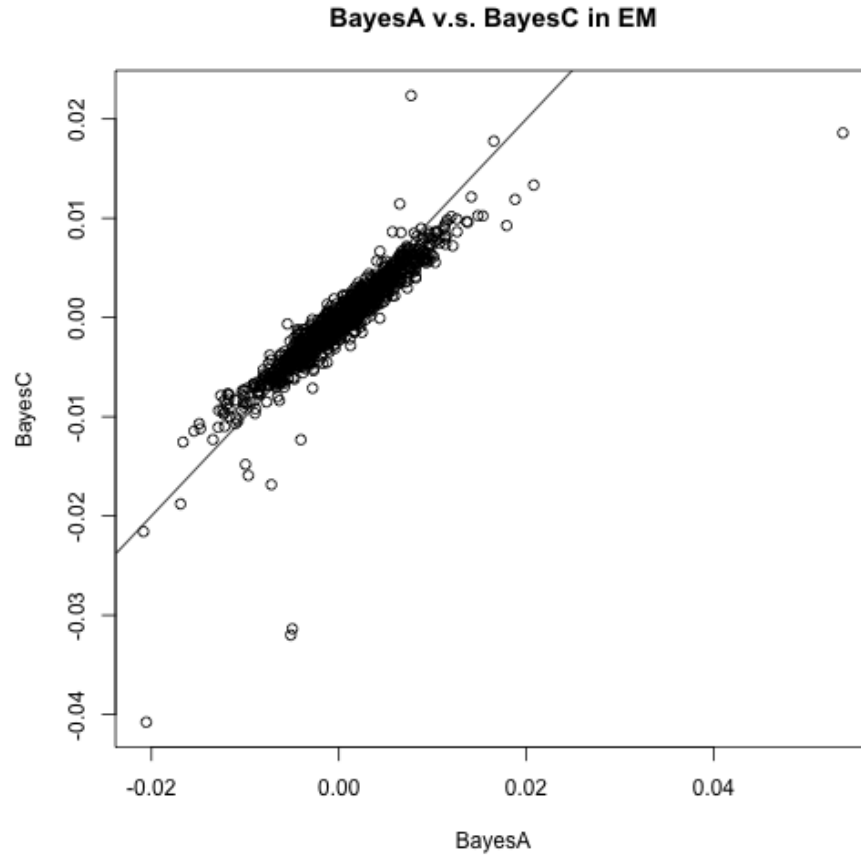


Figure 4: BayesA_C_MCMC

We can also compare the difference between BayesA and BayesC for EM:

```
plot(ba$ghat,bc$ghat,xlab="BayesA",ylab="BayesC",main="BayesA v.s. BayesC in EM")
abline(a=0,b=1)
```



```
###BayesB
```

```
init=set_init(pig,df=5,pi_snp=0.05,h2=0.5,c=NULL,model="BayesB",centered=T,trait="driploss")
run_para=list(niter=2000,burnIn=100,skip=10)
print_mcmc=list(piter=100,time_est=T,print_to="screen")
update_para=list(df=F,scale=TRUE,pi=TRUE)
op<-create.options(model="BayesB",method="MCMC",ante=FALSE,priors=NULL,init=init,
update_para=update_para,run_para=run_para,save.at="BayesB",cv=NULL,print_mcmc=print_mcmc)

bb<-bafit(dataobj=pig,op=op,trait="driploss")
```

```
## iter= 100 vare= 0.423763 scale= 0.0048747
## timepercycle= 0.001 estimated time left= 1.56 seconds
## iter= 200 vare= 0.280101 scale= 0.00287471
## timepercycle= 0.001 estimated time left= 1.67 seconds
## iter= 300 vare= 0.397197 scale= 0.00256754
```

```

## timepercycle= 0.001 estimated time left= 1.73 seconds
## iter= 400 vare= 0.292023 scale= 0.00407976
## timepercycle= 0.001 estimated time left= 1.62 seconds
## iter= 500 vare= 0.333239 scale= 0.00278575
## timepercycle= 0.001 estimated time left= 1.6 seconds
## iter= 600 vare= 0.492323 scale= 0.00230333
## timepercycle= 0.001 estimated time left= 1.48 seconds
## iter= 700 vare= 0.306483 scale= 0.00213109
## timepercycle= 0.001 estimated time left= 1.42 seconds
## iter= 800 vare= 0.498594 scale= 0.00527824
## timepercycle= 0.001 estimated time left= 1.3 seconds
## iter= 900 vare= 0.499604 scale= 0.00230064
## timepercycle= 0.001 estimated time left= 1.18 seconds
## iter= 1000 vare= 0.439679 scale= 0.00243592
## timepercycle= 0.001 estimated time left= 1.07 seconds
## iter= 1100 vare= 0.5116 scale= 0.00187912
## timepercycle= 0.001 estimated time left= 0.96 seconds
## iter= 1200 vare= 0.432131 scale= 0.0040978
## timepercycle= 0.001 estimated time left= 0.85 seconds
## iter= 1300 vare= 0.370646 scale= 0.00263223
## timepercycle= 0.001 estimated time left= 0.74 seconds
## iter= 1400 vare= 0.436532 scale= 0.00052841
## timepercycle= 0.001 estimated time left= 0.64 seconds
## iter= 1500 vare= 0.471743 scale= 0.00077792
## timepercycle= 0.001 estimated time left= 0.55 seconds
## iter= 1600 vare= 0.25665 scale= 0.00151988
## timepercycle= 0.001 estimated time left= 0.44 seconds
## iter= 1700 vare= 0.494149 scale= 0.00290529
## timepercycle= 0.001 estimated time left= 0.33 seconds
## iter= 1800 vare= 0.385592 scale= 0.0041832
## timepercycle= 0.001 estimated time left= 0.22 seconds
## iter= 1900 vare= 0.465551 scale= 0.00263691
## timepercycle= 0.001 estimated time left= 0.11 seconds
## iter= 2000 vare= 0.432472 scale= 0.00410259
## timepercycle= 0.001 estimated time left= 0 seconds

```

GWA

```

prrr1<-get_pvalues(rr)
prrr2<-get_pvalues(rr,type="fixed")

manhattan_plot(prrr1,pig$map,threshold = 0.05,main="rrBLUP random effects test")

manhattan_plot(prrr2,pig$map,threshold = 0.001,main="rrBLUP fixed effects test")

```

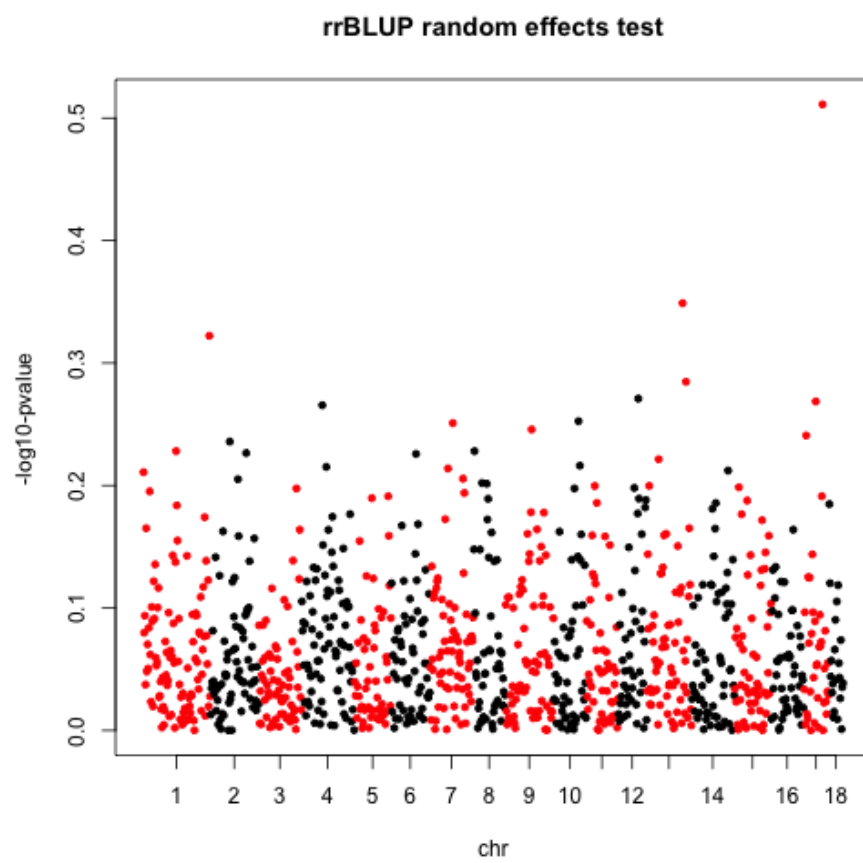


Figure 5: GWA

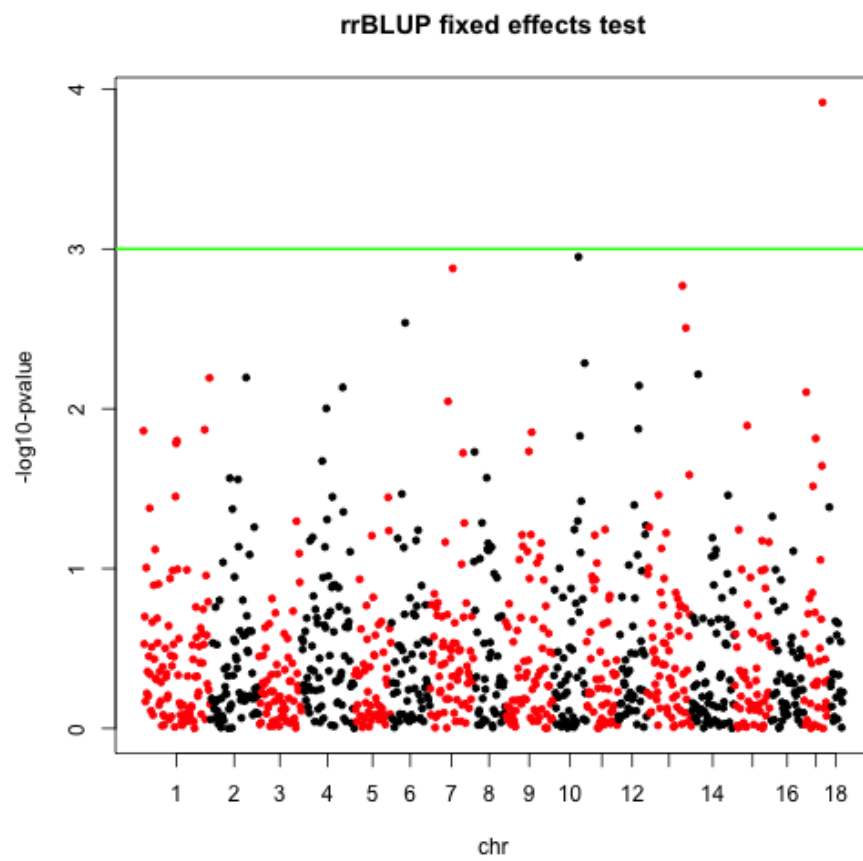


Figure 6: GWA

```
pba<-get_pvalues(ba_em,type="random")

manhattan_plot(pba,pig$map,threshold = 0.05,main="BayesA random effect test")
```

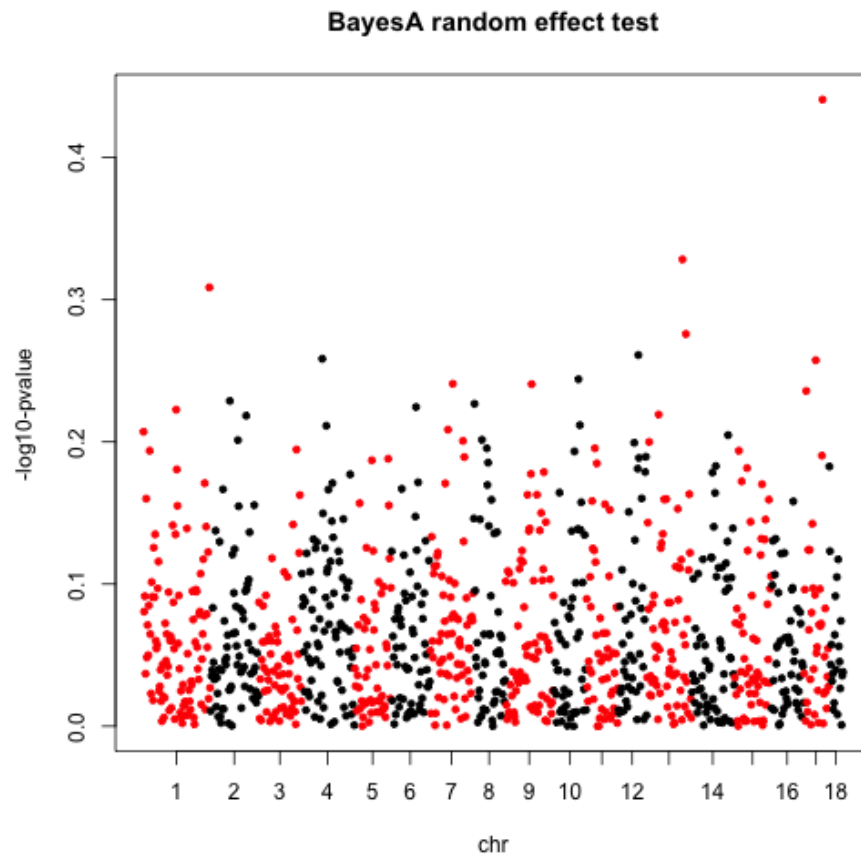


Figure 7: GWA

```
pbc<-get_pvalues(bc_em,type="random")

manhattan_plot(pbc,pig$map,threshold = 0.05,main="SSVS random effect test")

postprob_plot(bb$prob,pig$map,main="BayesB posterior probability")

postprob_plot(bc$phisave,pig$map,main="SSVS posterior probability")
```

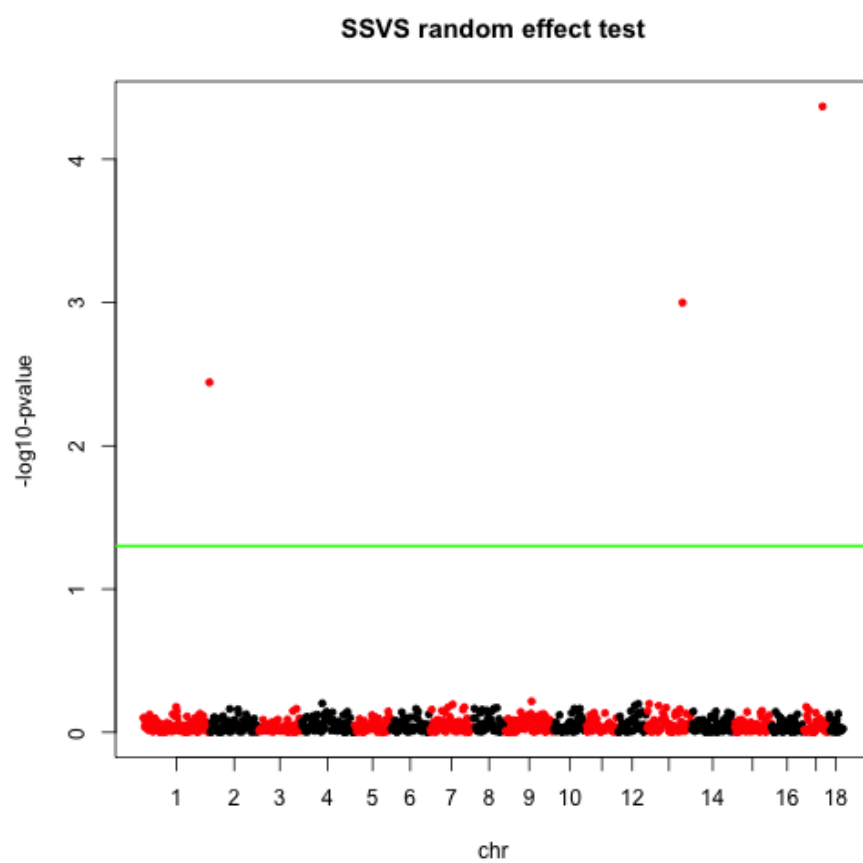


Figure 8: GWA

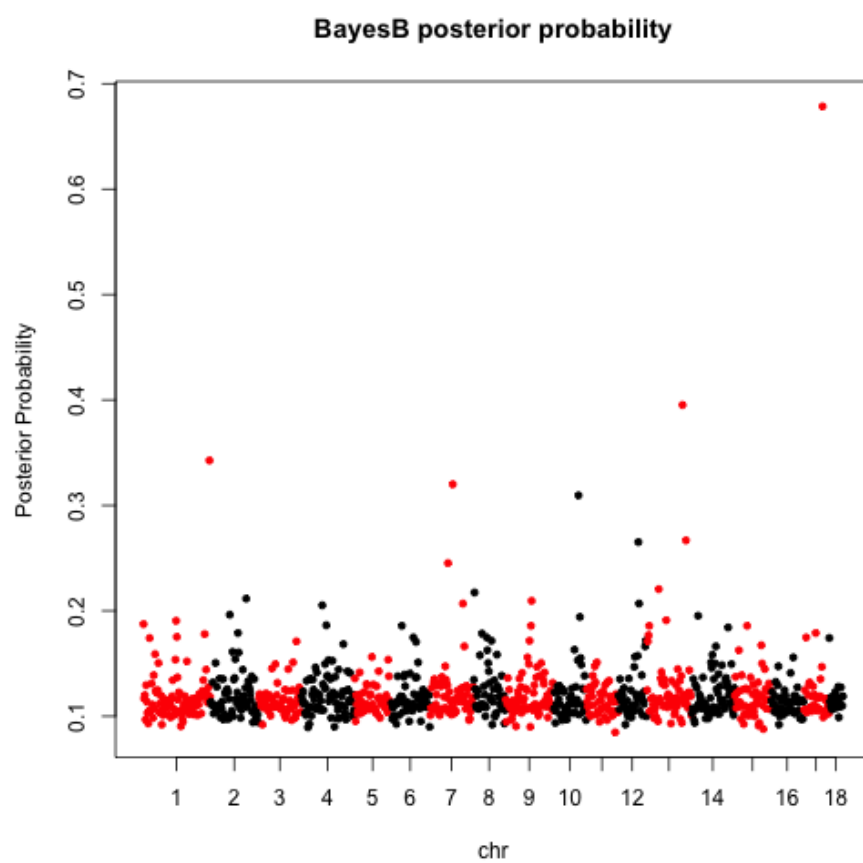


Figure 9: GWA

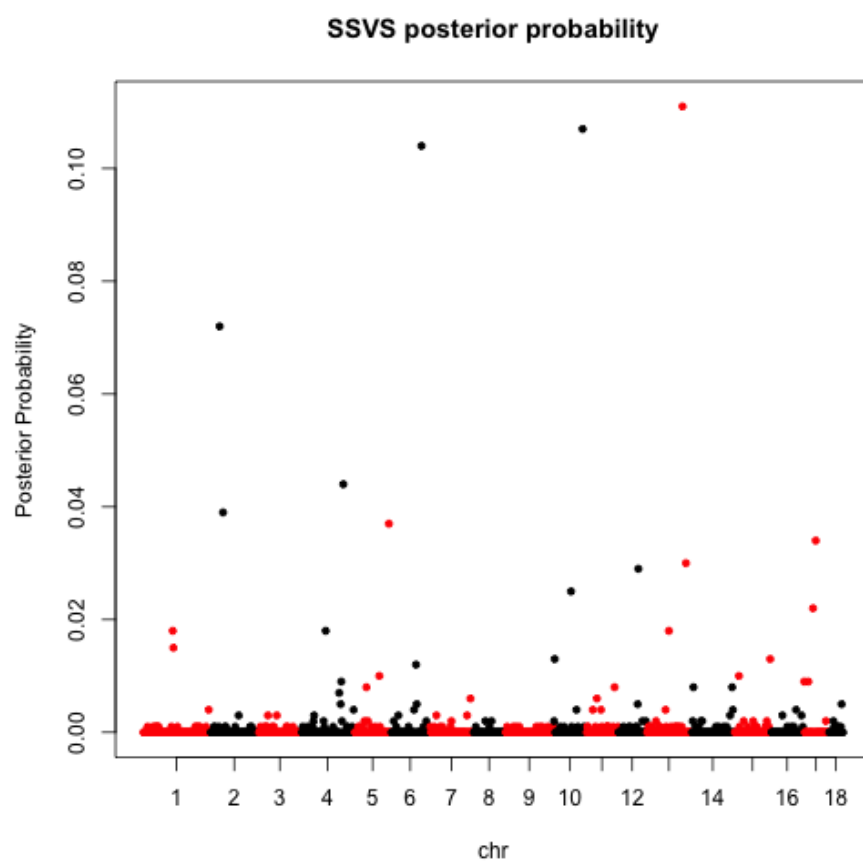


Figure 10: GWA