Week 9 Lab – MAST90125: Bayesian Statistical learning

Writing Gibbs samplers for linear models with proper priors for $\boldsymbol{\beta}$.

In this lab, we will continue discussing how to write code for Gibbs sampling of linear models with proper priors. We will use the data in `USJudgeRatings.csv`, which is available on Canvas. We will assume the variable `RTEN` is the response and the other variables are predictors. Meanwhile, how to analyze the generated chains will be included. In addition, we will show another example to see the difference between empirical Bayes and full Bayes.

Download `USJudgeRatings.csv` from Canvas.

Comment on the code below that purports to perform Gibbs sampling for a variety of linear models. See if you can determine what the code is doing. You may find referring back to Lectures 14 and 15 useful. Try comparing the posterior distributions to see what differences the priors cause.

Comment: Running this code and seeing it does not produce warning messages do not prove anything. You still need to check convergence. Remember in the assignment, you were given the following:

```
Processing chains for calculation of Gelman-Rubin diagnostics. Imagine you have 4 chains of
a multi-parameter problem, and thinning already completed, called par1,par2,par3,par4

Step one: Converting the chains into mcmc lists.
library(coda)
par1<-as.mcmc.list(as.mcmc((par1)))
par2<-as.mcmc.list(as.mcmc((par2)))
par3<-as.mcmc.list(as.mcmc((par3)))
par4<-as.mcmc.list(as.mcmc((par4)))

Step two: Calculating diagnostics

par.all<-c(par1,par2,par3,par4)
gelman.diag(par.all)

Step Three: Calculating effective sample size
effectiveSize(estml)
```

You may find this useful to check the performance of the codes given.

Examples of Gibbs samplers for linear models

First, consider the following two functions. What are they in Lecture 14?

```r
Gibbs.lm1<-function(X,y,tau0,iter,burnin){
p <- dim(X)[2]
XTX <- crossprod(X)
XTXinv <-solve(XTX)
XTY <- crossprod(X,y)
betahat<-solve(XTX,XTY)
tau    <-tau0
library(mvtnorm)

par<-matrix(0,iter,p+1)
for( i in 1:iter){
  beta <- rmvnorm(1,mean=betahat,sigma=XTXinv/tau)
  beta <-as.numeric(beta)
  err  <- y-X%*%beta
  tau  <- rgamma(1,0.5*n,0.5*sum(err^2))
  par[i,] <-c(beta,tau)
}

par <-par[-c(1:burnin),]
return(par)
}
```

```
Gibbs.lm2<-function(X,y,tau0,iter,burnin){
p <- dim(X)[2]
svdX <-svd(X)
U    <-svdX$u
Lambda<-svdX$d
V    <-svdX$v
Vbhat <- crossprod(U,y)/Lambda
tau <-tau0

vbeta<-rnorm(p)
par<-matrix(0,iter,p+1)
for( i in 1:iter){
  sqrttau<-sqrt(tau)
  vbeta <- rnorm(p,mean=Vbhat,sd=1/(sqrttau*Lambda) )
  beta <-V%*%vbeta
  err  <- y-X%*%beta
  tau  <- rgamma(1,0.5*n,0.5*sum(err^2))
  par[i,] <-c(beta,tau)
}

par <-par[-c(1:burnin),]
return(par)
}
```

What is the different between the above functions?

Then, we focus on things we talked about in Lecture 15.

- Linear mixed model/ ridge regression (flat prior for $\boldsymbol{\beta}_0$, $p(\tau_e) = \text{Ga}(\alpha_e, \gamma_e)$ with $\tau_e = (\sigma_e^2)^{-1}$, and $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \sigma_\beta^2 \mathbf{I})$ with $(\sigma_\beta^2)^{-1} = \tau_\beta \sim \text{Ga}(\alpha_\beta, \gamma_\beta)$.)

```
normalmm.Gibbs<-function(iter,Z,X,y,burnin,taue_0,tauu_0,a.u,b.u,a.e,b.e){
  n   <-length(y) #no. observations
  p   <-dim(X)[2] #no of fixed effect predictors.
  q   <-dim(Z)[2] #no of random effect levels
  tauu<-tauu_0
```

```r
  taue<-taue_0
  beta0<-rnorm(p)
  u0   <-rnorm(q,0,sd=1/sqrt(tauu))

  #Building combined predictor matrix.
  W<-cbind(X,Z)
  WTW <-crossprod(W)
  library(mvtnorm)

  #storing results.
  par <-matrix(0,iter,p+q+2)

  #Create modified identity matrix for joint posterior.
  I0  <-diag(p+q)
  diag(I0)[1:p]<-0

  for(i in 1:iter){
    #Conditional posteriors.
    tauu <-rgamma(1,a.u+0.5*q,b.u+0.5*sum(u0^2))
    #Updating component of normal posterior for beta,u
    Prec <-WTW + tauu*I0/taue
    P.mean <- solve(Prec)%*%crossprod(W,y)
    P.var  <-solve(Prec)/taue
    betau <-rmvnorm(1,mean=P.mean,sigma=P.var)
    betau <-as.numeric(betau)
    err   <- y-W%*%betau
    taue  <-rgamma(1,a.e+0.5*n,b.e+0.5*sum(err^2))
    #storing iterations.
    par[i,]<-c(betau,1/sqrt(tauu),1/sqrt(taue))
    beta0  <-betau[1:p]
    u0     <-betau[p+1:q]
  }

par <-par[-c(1:burnin),]
colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),'sigma_b','sigma_e')
 return(par)
}
```

Example: LASSO with either γ fixed or estimated from the data.

In Week 8 lab, you were given a function to fit a Bayesian LASSO, assuming the penalty, γ, was fixed. You would have noted that unlike frequentist LASSO, coefficient estimates in a Bayesian LASSO were never exactly zero.

In order to estimate γ, we return to the conditional posteriors we outlined in lecture 15 for Bayesian LASSO:

$$\begin{aligned}
p(\beta|\mathbf{y},\mathbf{X},\sigma_1^2,\dots\sigma_p^2,\tau_e) &= \mathcal{N}(\tau_e(\tau_e\mathbf{X}'\mathbf{X}+\mathbf{K}^{-1})^{-1}\mathbf{X}'\mathbf{y},(\tau_e\mathbf{X}'\mathbf{X}+\mathbf{K}^{-1})^{-1}), \text{where } \mathbf{K}_{jj}=\sigma_j^2\\
p(\tau_e|\mathbf{y},\boldsymbol{\beta},\mathbf{X}) &= \text{Ga}(\alpha_e+n/2,\gamma_e+(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})/2),\\
p((\sigma_j^2)^{-1}|\gamma,\boldsymbol{\beta}) &= \text{InvGaussian}(\gamma/|\boldsymbol{\beta}_j|,\gamma^2).
\end{aligned}$$

However we want to estimate γ as well. We know that the only place γ appeared in the joint distribution $p(\mathbf{y},\boldsymbol{\beta},\tau_e,\sigma_1^2,\dots\sigma_p^2,\gamma)$ is in the expansion of the Laplace (or double exponential) prior for $\boldsymbol{\beta}$,

$$\prod_{j=1}^p \frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{\beta_j^2}{2\sigma_j^2}}\frac{\gamma^2}{2}e^{-\frac{\gamma^2\sigma_j^2}{2}} \propto (\gamma^2)^p e^{-\frac{\gamma^2\sum_{j=1}^p\sigma_j^2}{2}}.$$

Looking at this kernel, we see that γ^2 (note not γ) corresponds to a kernel of a Gamma distribution. We also know that a Gamma prior and Gamma likelihood leads to a Gamma posterior. Therefore if we assume $\gamma^2 \sim \text{Ga}(\alpha,\delta)$ *a priori*, then the conditional posterior of γ^2 is,

$$p(\gamma^2|\sigma_1^2,\dots\sigma_p^2) = \text{Ga}(\alpha+p,\delta+0.5\sum_{j=1}^p\sigma_j^2)$$

As $\gamma\in(0,\infty)$, squaring γ is a one to one transformation. Therefore, there is no problem sampling γ^2 from the conditional posterior, taking the square root to obtain a draw for γ and then cycling through the remaining conditional posteriors.

Code for implementing LASSO with either γ fixed or estimated Note: To run this, you need to install R package *LaplacesDemon*

```r
#LASSO with fixed gamma
#Arguments are
#iter: no of iterations
#Z: Predictor matrix for effects with LASSO penalty
#X: Predictor matrix for effects without LASSO penalty (typically only intercept)
#y: response vector
#burnin: number of initial iterations to discard.
#taue_0: initial guess for residual precision.
#gamma: prior parameter for Laplace (double exponential) prior for u
#a.e, b.e: hyper-parameters of gamma prior for taue

normallassofixed.Gibbs<-function(iter,Z,X,y,burnin,taue_0,gamma,a.e,b.e){
  library(LaplacesDemon)
  n   <-length(y) #no. observations
  p   <-dim(X)[2] #no of fixed effect predictors.
  q   <-dim(Z)[2] #no of random effect levels
  taue<-taue_0
  tauu <-rinvgaussian(q,gamma/abs(rnorm(q)),gamma^2)

  #Building combined predictor matrix.
  W<-cbind(X,Z)
  WTW <-crossprod(W)
  library(mvtnorm)

  #storing results.
  par <-matrix(0,iter,p+q+1)

  #Calculating log predictive densities
  lppd<-matrix(0,iter,n)

  for(i in 1:iter){
    #Conditional posteriors.

    #Updating component of normal posterior for beta,u
    Kinv  <-diag(p+q)
    diag(Kinv)[1:p]<-0
    diag(Kinv)[p+1:q]<-tauu

    Prec <-taue*WTW + Kinv
    P.var  <-solve(Prec)
    P.mean <- taue*P.var%*%crossprod(W,y)
    betau <-rmvnorm(1,mean=P.mean,sigma=P.var)
    betau <-as.numeric(betau)
    err   <- y-W%*%betau
    taue  <-rgamma(1,a.e+0.5*n,b.e+0.5*sum(err^2))
    tauu <-rinvgaussian(q,gamma/abs(betau[-c(1:p)]),gamma^2)

    #storing iterations.
    par[i,]<-c(betau,1/sqrt(taue))
    #Storing log=predictive density
    lppd[i,]= dnorm(y,mean=as.numeric(W%*%betau),sd=1/sqrt(taue))
  }

  lppd      = lppd[-c(1:burnin),]
  lppdest   = sum(log(colMeans(lppd)))        #Estimating lppd for whole dataset.
  pwaic2    = sum(apply(log(lppd),2,FUN=var)) #Estimating effective number of parameters.
  par <-par[-c(1:burnin),]
  colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),'sigma_e')
  mresult<-list(par,lppdest,pwaic2)
  names(mresult)<-c('par','lppd','pwaic')
  return(mresult)
}

#####################################################################################
#Now the function where gamma is updated in the Gibbs sampler.
#Arguments are
#iter: no of iterations
#Z: Predictor matrix for effects with LASSO penalty
#X: Predictor matrix for effects without LASSO penalty (typically only intercept)
#y: response vector
```

```r
#burnin: number of initial iterations to discard.
#taue_0: initial guess for residual precision.
#a.l, b.l: hyper-parameters of gamma prior for (gamma^2),
#where gamma is parameter of Laplace for u.
#a.e, b.e: hyper-parameters of gamma prior for taue

normallassoestimated.Gibbs<-function(iter,Z,X,y,burnin,taue_0,a.l,b.l,a.e,b.e){
  library(LaplacesDemon)
  n   <-length(y) #no. observations
  p   <-dim(X)[2] #no of fixed effect predictors.
  q   <-dim(Z)[2] #no of random effect levels
  taue<-taue_0
  gamma2<-rgamma(1,a.l,b.l)
  gamma <-sqrt(gamma2)
  tauu <-rinvgaussian(q,gamma/abs(rnorm(q)),gamma^2)

  #Building combined predictor matrix.
  W<-cbind(X,Z)
  WTW <-crossprod(W)
  library(mvtnorm)

  #storing results.
  par <-matrix(0,iter,p+q+1+1)

  #Calculating log predictive densities
  lppd<-matrix(0,iter,n)

  for(i in 1:iter){
    #Conditional posteriors.

    #Updating component of normal posterior for beta,u
    Kinv  <-diag(p+q)
    diag(Kinv)[1:p]<-0
    diag(Kinv)[p+1:q]<-tauu

    Prec <-taue*WTW + Kinv
    P.var  <-solve(Prec)
    P.mean <- taue*P.var%*%crossprod(W,y)
    betau <-rmvnorm(1,mean=P.mean,sigma=P.var)
    betau <-as.numeric(betau)
    err   <- y-W%*%betau
    taue  <-rgamma(1,a.e+0.5*n,b.e+0.5*sum(err^2))
    tauu <-rinvgaussian(q,gamma/abs(betau[-c(1:p)]),gamma^2)
    gamma2 <-rgamma(1,a.l+q,b.l+0.5*sum(1/tauu))
    gamma  <-sqrt(gamma2)
    #storing iterations.
    par[i,]<-c(betau,1/sqrt(taue),gamma)
    lppd[i,]= dnorm(y,mean=as.numeric(W%*%betau),sd=1/sqrt(taue))
  }

  lppd      = lppd[-c(1:burnin),]
  lppdest   = sum(log(colMeans(lppd)))        #Estimating lppd for whole dataset.
  pwaic2    = sum(apply(log(lppd),2,FUN=var)) #Estimating effective number of parameters.
  par <-par[-c(1:burnin),]
  colnames(par)<-c(paste('beta',1:p,sep=''),paste('u',1:q,sep=''),'sigma_e','gamma')
  mresult<-list(par,lppdest,pwaic2)
  names(mresult)<-c('par','lppd','pwaic')
  return(mresult)
}
```

Fitting the two LASSO Gibbs samplers to the US Judge Ratings data

```r
data<-read.csv('./USJudgeRatings.csv')
# data<-read.csv(file.choose())
response<-data$RTEN  #response variable
n<-dim(data)[1]
intercept <-matrix(1,n,1) #Intercept (to be estimated without penalty)
Pred<-data[,2:12]         #Predictor variables.
Pred<-as.matrix(scale(Pred))
check1<-normallassoestimated.Gibbs(iter=10000,Z=Pred,X=intercept,y=response,burnin=2000,
                                   taue_0=1,a.l=0.1,b.l=0.1,a.e=0.01,b.e=0.01)
```

```
## Warning: package 'mvtnorm' was built under R version 4.3.1

##
## Attaching package: 'mvtnorm'

## The following objects are masked from 'package:LaplacesDemon':
##
##     dmvt, rmvt
```

```r
check2<-normallassoestimated.Gibbs(iter=10000,Z=Pred,X=intercept,y=response,burnin=2000,
                                   taue_0=5,a.l=0.1,b.l=0.1,a.e=0.01,b.e=0.01)
check3<-normallassoestimated.Gibbs(iter=10000,Z=Pred,X=intercept,y=response,burnin=2000,
                                   taue_0=0.2,a.l=0.1,b.l=0.1,a.e=0.01,b.e=0.01)


library(coda)
```

```
## Warning: package 'coda' was built under R version 4.3.1
```

```r
#Estimating Gelman -Rubin diagnostics.
#Note 8000 iterations were retained, so 50:50 split is iteration 1:4000 and iteration 4001:8000

#However first we must convert the output into mcmc lists for coda to interpret.
ml1<-as.mcmc.list(as.mcmc((check1$par[1:4000,])))
ml2<-as.mcmc.list(as.mcmc((check2$par[1:4000,])))
ml3<-as.mcmc.list(as.mcmc((check3$par[1:4000,])))
ml4<-as.mcmc.list(as.mcmc((check1$par[4000+1:4000,])))
ml5<-as.mcmc.list(as.mcmc((check2$par[4000+1:4000,])))
ml6<-as.mcmc.list(as.mcmc((check3$par[4000+1:4000,])))
estml<-c(ml1,ml2,ml3,ml4,ml5,ml6)

#Gelman-Rubin diagnostic.
gelman.diag(estml)[[1]]
```

```
##         Point est. Upper C.I.
## beta1    1.0000862  1.0003528
## u1       0.9998213  0.9998895
## u2       1.0001800  1.0006624
## u3       1.0002569  1.0010133
## u4       1.0005268  1.0014490
## u5       1.0000397  1.0004620
## u6       1.0001882  1.0008292
## u7       1.0004854  1.0014724
## u8       1.0000294  1.0002437
## u9       1.0003781  1.0012732
## u10      0.9999764  1.0000732
## u11      1.0004831  1.0015013
## sigma_e  1.0005302  1.0013329
## gamma    1.0018676  1.0047429
```

```
#effective sample size.
effectiveSize(estml)
```

```
##     beta1        u1        u2        u3        u4        u5        u6        u7
## 23687.235 22263.704 15337.907 15420.864 20755.509 15002.665 15822.395 21369.798
##        u8        u9       u10       u11   sigma_e     gamma
## 19468.515 11976.700 19460.104 16772.086 14818.606  3803.943
```

```
#For Empirical Bayes, we will fix gamma to the posterior mean of
#gamma from the chains fitted above. Note draws of gamma were stored in column 14.
#as we have intercept (column 1), 11 predictors (columns 2:12) and one standard deviation (column 13).
gamm.est<-mean(c(check1$par[,14],check2$par[,14],check3$par[,14]));gamm.est
```

```
## [1] 4.21387
```

```
check4<-normallassofixed.Gibbs(iter=10000,Z=Pred,X=intercept,y=response,burnin=2000,
                               taue_0=1,gamma=gamm.est,a.e=0.01,b.e=0.01)
check5<-normallassofixed.Gibbs(iter=10000,Z=Pred,X=intercept,y=response,burnin=2000,
                               taue_0=5,gamma=gamm.est,a.e=0.01,b.e=0.01)
check6<-normallassofixed.Gibbs(iter=10000,Z=Pred,X=intercept,y=response,burnin=2000,
                               taue_0=0.2,gamma=gamm.est,a.e=0.01,b.e=0.01)

library(coda)
#Estimating Gelman -Rubin diagnostics.
#Note 8000 iterations were retained, so 50:50 split is iteration 1:4000 and iteration 4001:8000

#However first we must convert the output into mcmc lists for coda to interpret.
fml1<-as.mcmc.list(as.mcmc((check4$par[1:4000,])))
fml2<-as.mcmc.list(as.mcmc((check5$par[1:4000,])))
fml3<-as.mcmc.list(as.mcmc((check6$par[1:4000,])))
fml4<-as.mcmc.list(as.mcmc((check4$par[4000+1:4000,])))
fml5<-as.mcmc.list(as.mcmc((check5$par[4000+1:4000,])))
fml6<-as.mcmc.list(as.mcmc((check6$par[4000+1:4000,])))
fixml<-c(fml1,fml2,fml3,fml4,fml5,fml6)

#Gelman-Rubin diagnostic.
gelman.diag(fixml)[[1]]
```

```
##         Point est. Upper C.I.
## beta1    1.0007979   1.002246
## u1       1.0003962   1.001277
## u2       1.0002264   1.000879
## u3       1.0004644   1.001534
## u4       1.0000186   1.000348
## u5       1.0002160   1.000762
## u6       1.0004887   1.001480
## u7       1.0001113   1.000415
## u8       1.0002674   1.000988
## u9       0.9999507   1.000140
## u10      1.0001722   1.000597
## u11      1.0004194   1.001044
## sigma_e  1.0010133   1.002849
```

```
#effective sample size.
effectiveSize(fixml)
```

```
##    beta1       u1       u2       u3       u4       u5       u6       u7
## 25513.12 22495.61 15560.86 15314.90 21050.51 19580.02 16681.71 23014.64
##       u8       u9      u10      u11  sigma_e
## 19487.83 12807.65 18003.08 18828.81 14856.80
```

```
#Combining all chains from each model
check.all1<-rbind(check1$par,check2$par,check3$par) #all chains where gamma was estimated.
check.all2<-rbind(check4$par,check5$par,check6$par) #all chains where gamma was fixed.

#Plots of results.
par(mfrow=c(5,3))
#Intercept
plot(density(check.all1[,1]),col=1,lty=1,xlab=expression(beta[0]),main='Comparison of
     posteriors for intercepts',cex.lab=1.5)
lines(density(check.all2[,1]),col=2,lty=2)
legend('topright',legend=c(expression(paste(gamma,' estimated')),
                expression(paste(gamma,' fixed'))),col=1:2,lty=1,bty='n',cex=1.5)

#co-officients
for(i in 1:11){
  plot(density(check.all1[,i+1]),col=1,lty=1,xlab=paste('u',i,sep=''),main='Comparison
       of posteriors for coefficents',cex.lab=1.5)
  lines(density(check.all2[,i+1]),col=2,lty=2)
  curve(0.5*gamm.est*exp(-gamm.est*abs(x)),col=3,lty=1,add=TRUE)
  legend('topright',legend=c(expression(paste(gamma,' estimated')),
      expression(paste(gamma,' fixed')),'prior'),col=1:3,lty=1,bty='n',cex=1.5)
}

#Standard deviation
plot(density(check.all1[,13]),col=1,lty=1,xlab=expression(sigma),main='Comparison
     of posteriors for std deviation',cex.lab=1.5)
lines(density(check.all2[,13]),col=2,lty=2)
legend('topright',legend=c(expression(paste(gamma,' estimated')),
    expression(paste(gamma,' fixed'))),col=1:2,lty=1,bty='n',cex=1.5)
```

```
#Comparing variances for Empirical Bayes versus fully Bayesian.

#Empirical Bayes
apply(check.all2,2,FUN=var) #Empirical Bayes
```

```
##        beta1           u1           u2           u3           u4           u5
## 0.0003393520 0.0005099926 0.0090171917 0.0086602040 0.0091643125 0.0110051488
##           u6           u7           u8           u9          u10          u11
## 0.0100644717 0.0201513690 0.0216581544 0.0489492041 0.0301592462 0.0029368563
##      sigma_e
## 0.0002295644
```

```
apply(check.all1,2,FUN=var) #Fully Bayesian
```

```
##        beta1           u1           u2           u3           u4           u5
## 0.0003452926 0.0005073989 0.0091736911 0.0085995952 0.0095151673 0.0113413212
##           u6           u7           u8           u9          u10          u11
## 0.0101843203 0.0212370276 0.0227977232 0.0510661207 0.0314368670 0.0029795685
##      sigma_e        gamma
## 0.0002340547 1.3578485882
```

```
#Fully Bayesian lppd estimate
check1$lppd
```

```
## [1] 34.5582
```

```
check2$lppd
```

```
## [1] 34.51095
```

```
check3$lppd
```

```
## [1] 34.53152
```

```
#Empirical Bayes lppd estimate
check4$lppd
```

```
## [1] 34.54434
```

```
check5$lppd
```

```
## [1] 34.59685
```

```
check6$lppd
```

```
## [1] 34.60497
```

```r
#Fully Bayesian effective number of parameters
check1$pwaic
```

```
## [1] 9.154188
```

```r
check2$pwaic
```

```
## [1] 9.069061
```

```r
check3$pwaic
```

```
## [1] 9.24785
```

```r
#Empirical Bayes effective number of parameters
check4$pwaic
```

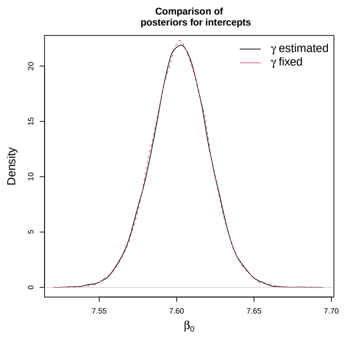
```
## [1] 9.020075
```

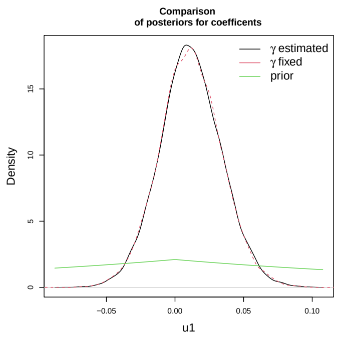
```r
check5$pwaic
```

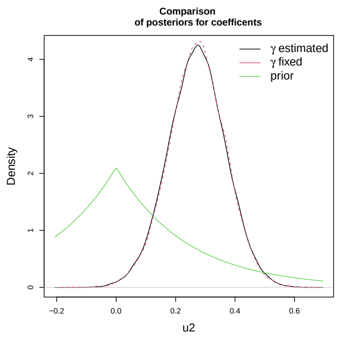
```
## [1] 9.224122
```

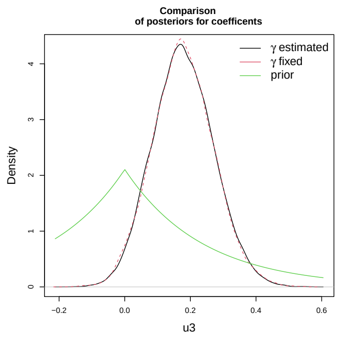
```r
check6$pwaic
```

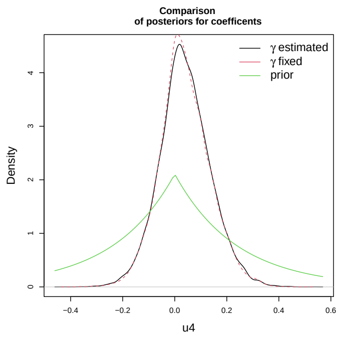
```
## [1] 9.072524
```

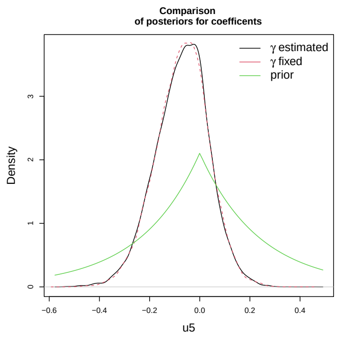

Comparison of posteriors for intercepts
γ estimated
γ fixed
Density
β_0

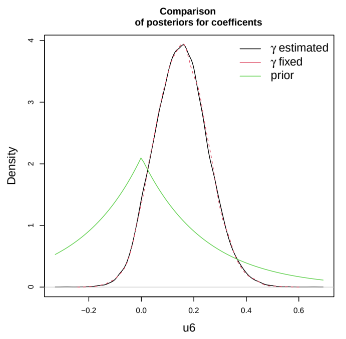

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u1

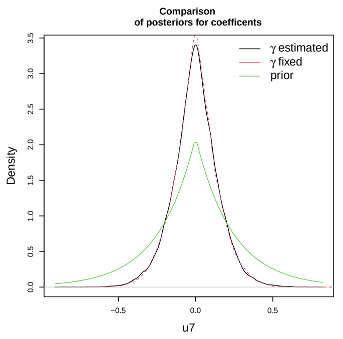

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u2

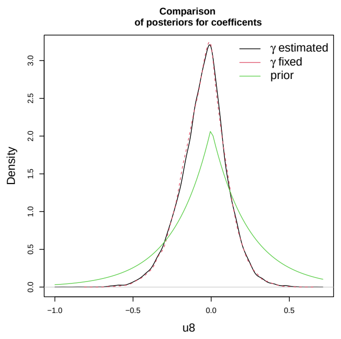

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u3

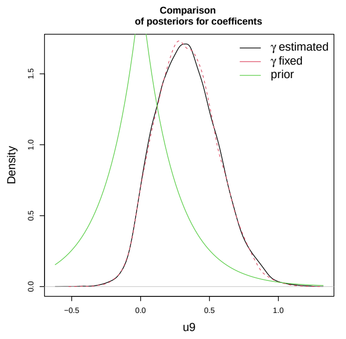

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u4

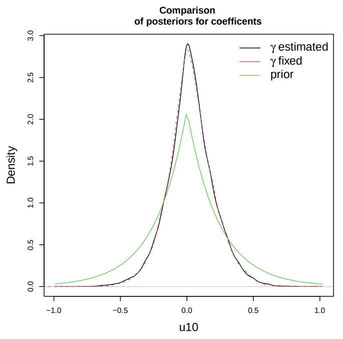

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u5

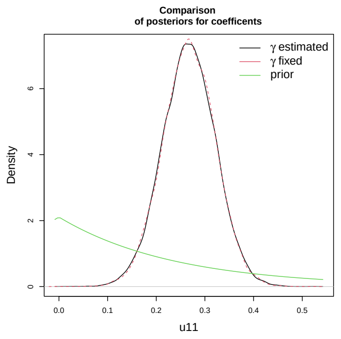

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u6

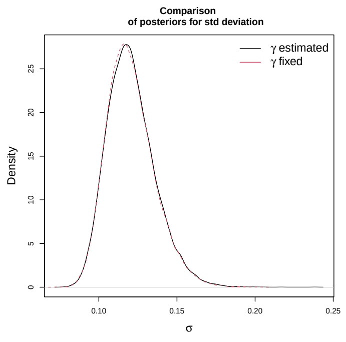

Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u7


Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u8


Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u9


Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u10


Comparison of posteriors for coefficents
γ estimated
γ fixed
prior
Density
u11


Comparison of posteriors for std deviation
γ estimated
γ fixed
Density
σ