Lab 9 solutions MAST90125: Bayesian Statistical learning

Variational Bayes.

In lecture 18, we looked at (mean-field) Variational Bayes as a method to find approximate posterior distributions for sub-blocks of the parameter vector $\boldsymbol{\theta}$, for the model

$$\begin{aligned}
p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{u}, \tau_e) &= \mathcal{N}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \frac{1}{\tau_e}\mathbf{I}_n) \\
p(\boldsymbol{\beta}) &\propto 1 \\
p(\mathbf{u}) &= \mathcal{N}(\mathbf{0}_q, \frac{1}{\tau_u}\mathbf{K}) \\
p(\tau_e) &= \mathrm{Ga}(\alpha_e, \gamma_e) \\
p(\tau_u) &= \mathrm{Ga}(\alpha_u, \gamma_u)
\end{aligned}$$

Instructions for lab

Download `farmdata.csv` and `Kmat.csv` from LMS.

Re-format the code given in lecture 18 for performing Variational Bayes so that rather than constructing approximate posteriors,

- $Q(\boldsymbol{\beta})$
- $Q(\mathbf{u})$
- $Q(\tau_e)$
- $Q(\tau_u)$

you determine the approximate posteriors

- $Q(\boldsymbol{\beta}, \mathbf{u})$
- $Q(\tau_e)$
- $Q(\tau_u)$.

Compare the performance of the new blocking structure to that used in class.

Solution

The difference between the code given in class and this lab is that we approximate the posterior of $\boldsymbol{\beta}, \boldsymbol{u}$ jointly. To do this, note the trick that the joint kernel of $\boldsymbol{\beta}, \mathbf{u}$ can be wirtten as

$$p(\boldsymbol{\beta}, \mathbf{u}) \propto e^{-\frac{\tau_u(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{0}_{p\times p} & \mathbf{0}_{p\times q} \\ \mathbf{0}_{q\times p} & \mathbf{K}^{-1}\end{pmatrix}\begin{pmatrix}\boldsymbol{\beta} \\ \mathbf{u}\end{pmatrix}}{2}},$$

Hence in terms of coding, updating $\boldsymbol{\beta}, \mathbf{u}$ jointly looks just like updating $\mathbf{u}$ in the code given in lecture 18, except that $\boldsymbol{\beta} = \mathbf{0}$.

As a reminder, the joint distribution, $p(y, \boldsymbol{\beta}, \mathbf{u}, \tau_e, \tau_u | \mathbf{X}, \mathbf{Z})$ is

$$\left(\frac{\tau_e}{2\pi}\right)^{\frac{n}{2}} e^{-\frac{\tau_e(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})}{2}} \times 1 \times \left(\frac{\tau_u}{2\pi}\right)^{\frac{q}{2}} \det(\mathbf{K})^{-1/2} e^{-\frac{\tau_u \mathbf{u}'\mathbf{K}^{-1}\mathbf{u}}{2}} \times \frac{\gamma_u^{\alpha_u} \tau_u^{\alpha_u - 1} e^{-\gamma_u \tau_u}}{\Gamma(\alpha_u)} \times \frac{\gamma_e^{\alpha_e} \tau_e^{\alpha_e - 1} e^{-\gamma_e \tau_e}}{\Gamma(\alpha_e)},$$

and that we need to determine the expectation of log-kernels.

- For τ_e, the kernel and log-kernel are respectively

$$\begin{aligned} \text{Kernel:} \quad & \tau_e{}^{\frac{n}{2}} e^{-\frac{\tau_e(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})}{2}} \tau_e^{\alpha_e - 1} e^{-\gamma_e \tau_e} \\ \text{Log-kernel:} \quad & (n/2 + \alpha_e - 1)\log(\tau_e) - \tau_e(\gamma_e + (\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u})/2) \end{aligned}$$

- The expected log-kernel $E_{-\tau_e}$(Log-kernel) is

$$(\frac{n}{2} + \alpha_e - 1)\log(\tau_e) - \tau_e(\gamma_e + \frac{\mathbf{y}'\mathbf{y} + E_{\boldsymbol{\beta},\mathbf{u}}\left(\begin{pmatrix} \boldsymbol{\beta}' & \mathbf{u}' \end{pmatrix} \begin{pmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ \mathbf{u} \end{pmatrix} \right)}{2} - \mathbf{y}'\mathbf{X}E_{\boldsymbol{\beta},\mathbf{u}}(\beta) - \mathbf{y}'\mathbf{Z}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u}))$$

- Using the tricks that $E(\mathbf{x}\mathbf{x}') = \text{Var}(\mathbf{x}) + E(\mathbf{x})E(\mathbf{x})'$ and that $\mathbf{a}'\mathbf{D}\mathbf{a} = \text{Tr}(\mathbf{a}'\mathbf{D}\mathbf{a}) = \text{Tr}(\mathbf{D}\mathbf{a}\mathbf{a}')$, the expected log-kernel can be written as,

$$\left(\frac{n}{2} + \alpha_e - 1\right)\log(\tau_e) - \tau_e\left(\gamma_e + \frac{(\mathbf{y} - \mathbf{X}E_{\boldsymbol{\beta},\mathbf{u}}(\boldsymbol{\beta}) - \mathbf{Z}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u}))'(\mathbf{y} - \mathbf{X}E_{\boldsymbol{\beta},\mathbf{u}}(\boldsymbol{\beta}) - \mathbf{Z}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u})) + \text{Tr}\left(\begin{pmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} \end{pmatrix} \begin{pmatrix} \text{Var}(\boldsymbol{\beta}) & \text{Var}(\boldsymbol{\beta}, \mathbf{u}) \\ \text{Var}(\mathbf{u}, \boldsymbol{\beta}) & \text{Var}(\mathbf{u}) \end{pmatrix} \right)}{2}\right)$$

- which indicates that the approximate posterior for τ_e is

$$\text{Ga}\left(\frac{n}{2} + \alpha_e, \gamma_e + \frac{(\mathbf{y} - \mathbf{X}E_{\boldsymbol{\beta},\mathbf{u}}(\boldsymbol{\beta}) - \mathbf{Z}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u}))'(\mathbf{y} - \mathbf{X}E_{\boldsymbol{\beta},\mathbf{u}}(\boldsymbol{\beta}) - \mathbf{Z}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u})) + \text{Tr}\left(\begin{pmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} \end{pmatrix} \begin{pmatrix} \text{Var}(\boldsymbol{\beta}) & \text{Var}(\boldsymbol{\beta}, \mathbf{u}) \\ \text{Var}(\mathbf{u}, \boldsymbol{\beta}) & \text{Var}(\mathbf{u}) \end{pmatrix} \right)}{2}\right).$$

- For τ_u, the kernel and log-kernel are respectively

$$\begin{aligned} \text{Kernel:} \quad & \left(\frac{\tau_u}{2\pi}\right)^{\frac{q}{2}} e^{-\frac{\tau_u \mathbf{u}'\mathbf{K}^{-1}\mathbf{u}}{2}} \tau_u^{\alpha_u - 1} e^{-\gamma_u \tau_u} \\ \text{Log-kernel:} \quad & (q/2 + \alpha_u - 1)\log(\tau_u) - \tau_u(\gamma_u + \mathbf{u}'\mathbf{K}^{-1}\mathbf{u}/2) \end{aligned}$$

- The expected log-kernel $E_{-\tau_u}$(Log-kernel) is

$$\begin{aligned} &= (q/2 + \alpha_u - 1)\log(\tau_u) - \tau_u(\gamma_u + E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u}'\mathbf{K}^{-1}\mathbf{u})/2) \\ &= (q/2 + \alpha_u - 1)\log(\tau_u) - \tau_u(\gamma_u \text{Tr}(\mathbf{K}^{-1}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u}\mathbf{u}'))/2) \\ &= (q/2 + \alpha_u - 1)\log(\tau_u) - \tau_u(\gamma_u + E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u})'\mathbf{K}^{-1}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u})/2 + \text{Tr}(\mathbf{K}^{-1}Var(\mathbf{u}))/2) \end{aligned}$$

- which indicates that the approximate posterior for τ_u is

$$\text{Ga}\left(\frac{q}{2} + \alpha_u, \gamma_u + \frac{E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u})'\mathbf{K}^{-1}E_{\boldsymbol{\beta},\mathbf{u}}(\mathbf{u}) + \text{Tr}(\mathbf{K}^{-1}Var(\mathbf{u}))}{2}\right).$$

- For $\boldsymbol{\beta}, \mathbf{u}$, the kernel and log-kernel are respectively

$$\begin{aligned}
\text{Kernel:} \quad &= e^{-\frac{\tau_e(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})}{2} - \frac{\tau_u \mathbf{u}'\mathbf{K}^{-1}\mathbf{u}}{2}} \\
&= e^{-\frac{\tau_e(\mathbf{y}-(\mathbf{X} \quad \mathbf{Z})\binom{\boldsymbol{\beta}}{\mathbf{u}})'(\mathbf{y}-(\mathbf{X} \quad \mathbf{Z})\binom{\boldsymbol{\beta}}{\mathbf{u}})}{2}} e^{-\frac{\tau_u(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{0}_{p\times p} & \mathbf{0}_{p\times q}\\ \mathbf{0}_{q\times p} & \mathbf{K}^{-1}\end{pmatrix}\binom{\boldsymbol{\beta}}{\mathbf{u}}}{2}} \\
\text{Log-kernel:} \quad & -\frac{\tau_e(\mathbf{y}-(\mathbf{X} \quad \mathbf{Z})\binom{\boldsymbol{\beta}}{\mathbf{u}})'(\mathbf{y}-(\mathbf{X} \quad \mathbf{Z})\binom{\boldsymbol{\beta}}{\mathbf{u}})}{2} - \frac{\tau_u(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{0}_{p\times p} & \mathbf{0}_{p\times q}\\ \mathbf{0}_{q\times p} & \mathbf{K}^{-1}\end{pmatrix}\binom{\boldsymbol{\beta}}{\mathbf{u}}}{2}
\end{aligned}$$

- The expected log-kernel $E_{-\boldsymbol{\beta},\mathbf{u}}(\text{Log-kernel})$ is

$$\begin{aligned}
&= -\frac{E_{\tau_e}(\tau_e)(\mathbf{y}-(\mathbf{X} \quad \mathbf{Z})\binom{\boldsymbol{\beta}}{\mathbf{u}})'(\mathbf{y}-(\mathbf{X} \quad \mathbf{Z})\binom{\boldsymbol{\beta}}{\mathbf{u}})}{2} - \frac{E_{\tau_u}(\tau_u)(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{0}_{p\times p} & \mathbf{0}_{p\times q}\\ \mathbf{0}_{q\times p} & \mathbf{K}^{-1}\end{pmatrix}\binom{\boldsymbol{\beta}}{\mathbf{u}}}{2} \\
&\propto -\frac{E_{\tau_e}(\tau_e)(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z}\\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z}\end{pmatrix}\binom{\boldsymbol{\beta}}{\mathbf{u}}}{2} + E_{\tau_e}(\tau_e)(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{X}'\mathbf{y}\\ \mathbf{Z}'\mathbf{y}\end{pmatrix} - \frac{E_{\tau_u}(\tau_u)(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{0}_{p\times p} & \mathbf{0}_{p\times q}\\ \mathbf{0}_{q\times p} & \mathbf{K}^{-1}\end{pmatrix}\binom{\boldsymbol{\beta}}{\mathbf{u}}}{2} \\
&= -\frac{(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}E_{\tau_e}(\tau_e)\mathbf{X}'\mathbf{X} & E_{\tau_e}(\tau_e)\mathbf{X}'\mathbf{Z}\\ E_{\tau_e}(\tau_e)\mathbf{Z}'\mathbf{X} & E_{\tau_e}(\tau_e)\mathbf{Z}'\mathbf{Z} + E_{\tau_u}(\tau_u)\mathbf{K}^{-1}\end{pmatrix}\binom{\boldsymbol{\beta}}{\mathbf{u}}}{2} + E_{\tau_e}(\tau_e)(\boldsymbol{\beta}' \quad \mathbf{u}')\begin{pmatrix}\mathbf{X}'\mathbf{y}\\ \mathbf{Z}'\mathbf{y}\end{pmatrix}
\end{aligned}$$

- which indicates that the approximate posterior for $\boldsymbol{\beta}, \mathbf{u}$ is

$$\mathcal{N}\left(E_{\tau_e}(\tau_e)\begin{pmatrix}E_{\tau_e}(\tau_e)\mathbf{X}'\mathbf{X} & E_{\tau_e}(\tau_e)\mathbf{X}'\mathbf{Z}\\ E_{\tau_e}(\tau_e)\mathbf{Z}'\mathbf{X} & E_{\tau_e}(\tau_e)\mathbf{Z}'\mathbf{Z} + E_{\tau_u}(\tau_u)\mathbf{K}^{-1}\end{pmatrix}^{-1}\begin{pmatrix}\mathbf{X}'\mathbf{y}\\ \mathbf{Z}'\mathbf{y}\end{pmatrix}, \begin{pmatrix}E_{\tau_e}(\tau_e)\mathbf{X}'\mathbf{X} & E_{\tau_e}(\tau_e)\mathbf{X}'\mathbf{Z}\\ E_{\tau_e}(\tau_e)\mathbf{Z}'\mathbf{X} & E_{\tau_e}(\tau_e)\mathbf{Z}'\mathbf{Z} + E_{\tau_u}(\tau_u)\mathbf{K}^{-1}\end{pmatrix}^{-1}\right).$$

```r
#Arguments are
#epsilon: accuracy cut-off.
#iter: no of iterations
#Kinv: inverse of K, where p(u) = N(0,\sigma^2_u K)
#Z: Predictor matrix for random effects
#X: Predictor matrix for fixed effects
#y: response vector
#taue_0: initial guess for residual precision.
#tauu_0: initial guess for random effect precision
#a.u, g.u: hyper-parameters of gamma prior for tauu
#a.e, g.e: hyper-parameters of gamma prior for taue

#Output are final estimates, plus iteration number when convergence was reached.
VB.mm<-function(epsilon,iter,Kinv,Z,X,y,taue_0,tauu_0,u0,beta0,a.e,g.e,a.u,g.u){
  n<-dim(X)[1]
  p<-dim(X)[2]
  q<-dim(Z)[2]
  W <-cbind(X,Z)
  WTW<-crossprod(W)
  WTY<-crossprod(W,y)
```

```
  Kinvall<-matrix(0,p+q,p+q)
  Kinvall[-c(1:p),-c(1:p)]<-Kinv

  for(i in 1:iter){
  Vub <-solve(taue_0*WTW+tauu_0*Kinvall) #update Var(b,u)
  ub  <-taue_0*Vub%*%WTY                  #update E(b,u)
  TrKinvub <- sum(diag(Kinvall%*%Vub))
  uKinvub  <- t(ub)%*%Kinvall%*%ub
  tauu    <- (a.u+0.5*q)/(g.u+0.5*as.numeric(uKinvub)+0.5*TrKinvub)
  tauu    <- as.numeric(tauu)
  err     <- y - W%*%ub
  TrWTWub  <- sum(diag(WTW%*%Vub))
  taue    <- (a.e+0.5*n)/(g.e+0.5*sum(err^2)+0.5*TrWTWub)
  taue    <- as.numeric(taue)

  if(i > 1){
  diffub  <- sqrt((ub-ub0)^2)/(abs(ub)+0.01)
  diffte <- abs(taue_0-taue)/(taue+0.01)
  difftu <- abs(tauu_0-tauu)/(tauu+0.01)
  diffvub <- sqrt((diag(Vub0) - diag(Vub))^2)/(diag(Vub))
  diff.all<-c(diffub,diffte,difftu,diffvub)
  if(max(diff.all) < epsilon) break
  }
  Vub0 <- Vub;ub0<-ub;taue_0<-taue;tauu_0<-tauu
  #Calculate relative change.
  }

  taue.param<-c((a.e+0.5*n),(g.e+0.5*sum(err^2)+0.5*TrWTWub))
  tauu.param<-c((a.u+0.5*q),(g.u+0.5*uKinvub+0.5*TrKinvub))
  param<-list(ub,Vub,taue.param,tauu.param,i)
  names(param)<-c('betau_mean','betau_var','tau_e','tau_u','iter')
  return(param)
}
```

The R code used to implement Gibbs sampling for this mixed model in lecture 18 is given below:

```
#Arguments are
#iter: no of iterations
#Z: Predictor matrix for random effects
#X: Predictor matrix for fixed effects
#y: response vector
#burnin: number of initial iterations to discard.
#taue_0: initial guess for residual precision.
#tauu_0: initial guess for random effect precision
#Kinv: inverse of K, where p(u) = N(0,\sigma^2_u K)
#a.u, b.u: hyper-parameters of gamma prior for tauu
#a.e, b.e: hyper-parameters of gamma prior for taue

normalmm.Gibbs<-function(iter,Z,X,y,burnin,taue_0,tauu_0,Kinv,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
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)
WTy <-crossprod(W,y)
library(mvtnorm)

#storing results.
par <-matrix(0,iter,p+q+2)
#Calculating log predictive densities
lppd<-matrix(0,iter,n)

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

for(i in 1:iter){
#Conditional posteriors.
  uKinvu <- t(u0)%*%Kinv%*%u0
  uKinvu <-as.numeric(uKinvu)
tauu <-rgamma(1,a.u+0.5*q,b.u+0.5*uKinvu)
#Updating component of normal posterior for beta,u
Prec <-WTW + tauu*I0/taue
P.mean<- solve(Prec)%*%WTy
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]
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_b','sigma_e')
mresult<-list(par,lppdest,pwaic2)
names(mresult)<-c('par','lppd','pwaic')
return(mresult)
}
```

Any other code you may use can be modified from the code given in Lecture 18. Possible things you may want to check include:

- convergence of Gibbs sampler
- comparing the empirical and approximate distributions using density plots.

```r
#data<-read.csv('farmdata.csv')
data<-read.csv(file.choose())
#K   <-read.csv('Kmat.csv')
K<-read.csv(file.choose())
K   <-as.matrix(K)
Kinv<-solve(K)

n<-dim(data)[1]
q<-dim(Kinv)[1]
X<-table(1:n,data$flock) #flock is fixed effect
#Indicator matrix for parents.
Z2<-table(1:n,data$sire)
Z3<-cbind(Z2,table(1:n,data$dam))
```

Importing and formatting the data

```r
system.time(chain1<-normalmm.Gibbs(iter=10000,Z=Z3,X=X,y=data$y,burnin=2000,taue_0=5,tauu_0=0.2,Kinv=Ki
```

running the Gibbs sampler, checking convergence, and calculating effective sample size

```
## Warning: package 'mvtnorm' was built under R version 4.0.5

##    user  system elapsed
##    4.68    0.17    4.87
```

```r
system.time(chain2<-normalmm.Gibbs(iter=10000,Z=Z3,X=X,y=data$y,burnin=2000,taue_0=1,tauu_0=1,Kinv=Kinv
```

```
##    user  system elapsed
##    4.56    0.06    4.63
```

```r
system.time(chain3<-normalmm.Gibbs(iter=10000,Z=Z3,X=X,y=data$y,burnin=2000,taue_0=0.2,tauu_0=3,Kinv=Ki
```

```
##    user  system elapsed
##    4.47    0.05    4.52
```

```r
library(coda)
```

```
## Warning: package 'coda' was built under R version 4.0.5
```

```r
rml1<-as.mcmc.list(as.mcmc((chain1$par[1:4000,])))
rml2<-as.mcmc.list(as.mcmc((chain2$par[1:4000,])))
rml3<-as.mcmc.list(as.mcmc((chain3$par[1:4000,])))
rml4<-as.mcmc.list(as.mcmc((chain1$par[4000+1:4000,])))
rml5<-as.mcmc.list(as.mcmc((chain2$par[4000+1:4000,])))
rml6<-as.mcmc.list(as.mcmc((chain3$par[4000+1:4000,])))
rml<-c(rml1,rml2,rml3,rml4,rml5,rml6)

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

```
##         Point est. Upper C.I.
## beta1    1.0001886   1.000618
## beta2    1.0003186   1.000664
## u1       1.0002849   1.000427
## u2       1.0004086   1.000672
```

```
## u3       1.0002609   1.000783
## u4       1.0002456   1.000754
## u5       1.0001370   1.000347
## u6       1.0001749   1.000684
## u7       1.0001805   1.000262
## u8       1.0000731   1.000364
## u9       1.0001241   1.000346
## u10      1.0000389   1.000371
## u11      0.9999831   1.000121
## u12      1.0000541   1.000331
## u13      1.0000604   1.000143
## sigma_b  1.0009851   1.002606
## sigma_e  1.0012429   1.002924
```

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

```
##     beta1     beta2        u1        u2        u3        u4        u5        u6
## 24000.000 24000.000 24000.000 21692.020 23909.883 24000.000 23848.486 24000.000
##        u7        u8        u9       u10       u11       u12       u13   sigma_b
## 24000.000 25176.790 23691.012 23376.080 24000.000 24000.000 24000.000 10917.902
##   sigma_e
##  8676.271
```

```
system.time(test1<-VB.mm(epsilon=1e-5,iter=2000,Kinv=Kinv,Z=Z3,X=X,y=data$y,taue_0=0.2,tauu_0=0.2,u0=rn
```

Estimating parameters of variational Bayes approximations

```
##    user  system elapsed
##    0.10    0.01    0.11
```

```
system.time(test2<-VB.mm(epsilon=1e-5,iter=2000,Kinv=Kinv,Z=Z3,X=X,y=data$y,taue_0=5,tauu_0=1,u0=rnorm(
```

```
##    user  system elapsed
##       0       0       0
```

```
system.time(test3<-VB.mm(epsilon=1e-5,iter=2000,Kinv=Kinv,Z=Z3,X=X,y=data$y,taue_0=1,tauu_0=5,u0=rnorm(
```

```
##    user  system elapsed
##       0       0       0
```

```
test1$iter
```

```
## [1] 20
```

```
test2$iter
```

```
## [1] 16
```

```
test3$iter
```

```
## [1] 18
```

```
#Comparing point estimates/posterior means

chain.all<-rbind(chain1$par,chain2$par,chain3$par)
```

```r
#beta, u
cbind(test1$betau_mean,test2$betau_mean,test3$betau_mean,colMeans(chain.all[,1:15]))
```

Comparing estimates obtained from Gibbs sampling and Variational Bayes

```
##            [,1]       [,2]        [,3]        [,4]
## 1   4.15443108  4.1544310  4.15443105  4.15312045
## 2   4.70775387  4.7077539  4.70775388  4.70932202
## 1   0.81178839  0.8117883  0.81178835  0.80841251
## 2  -1.09477605 -1.0947760 -1.09477600 -1.09169227
## 3   0.21037808  0.2103781  0.21037809  0.21098242
## 4   0.47799077  0.4779907  0.47799075  0.47601605
## 5  -0.53525688 -0.5352568 -0.53525681 -0.53454094
## 6  -0.70912449 -0.7091244 -0.70912442 -0.70499465
## 7   0.51809256  0.5180925  0.51809251  0.51611965
## 8  -0.16314311 -0.1631431 -0.16314311 -0.16250230
## 9  -0.14790542 -0.1479054 -0.14790543 -0.14939405
## 10  0.13369963  0.1336996  0.13369961  0.13095244
## 11  0.15439310  0.1543930  0.15439306  0.15130713
## 12 -0.07566564 -0.0756656 -0.07566562 -0.07495579
## 13  0.57227999  0.5722800  0.57227999  0.57072883
```

```r
#Variances.
sigmae2<-sigmau2<-rep(0,3)
sigmae2[1]<-test1$tau_e[2]/(test1$tau_e[1]-1)
sigmau2[1]<-test1$tau_u[2]/(test1$tau_u[1]-1)
sigmae2[2]<-test2$tau_e[2]/(test2$tau_e[1]-1)
sigmau2[2]<-test2$tau_u[2]/(test2$tau_u[1]-1)
sigmae2[3]<-test3$tau_e[2]/(test3$tau_e[1]-1)
sigmau2[3]<-test3$tau_u[2]/(test3$tau_u[1]-1)
cbind(sigmau2,mean(chain.all[,16]^2))
```

```
##        sigmau2
## [1,] 0.4304444 0.4649196
## [2,] 0.4304445 0.4649196
## [3,] 0.4304444 0.4649196
```

```r
cbind(sigmae2,mean(chain.all[,17]^2))
```

```
##         sigmae2
## [1,] 0.04876863 0.05242605
## [2,] 0.04876872 0.05242605
## [3,] 0.04876868 0.05242605
```

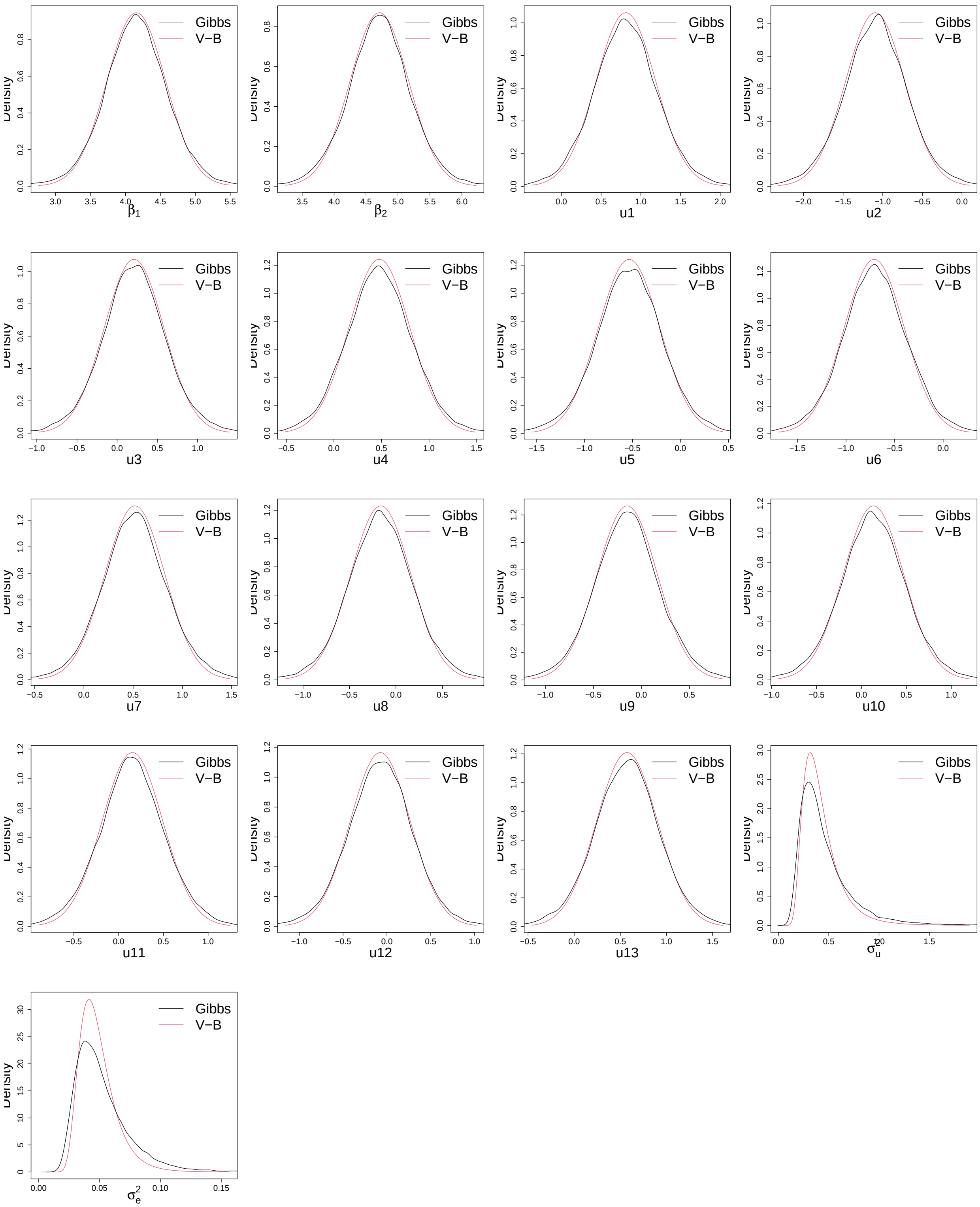
```r
#Comparing posterior distributions.
par(mfrow=c(5,4))
mlim<-quantile(chain.all[,1],c(0.005,0.995))
curve(dnorm(x,mean=test1$betau_mean[1],sd=sqrt(test1$betau_var[1,1])),ylab='Density',main='',xlim=mlim,
lines(density(chain.all[,1]))
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5)

mlim<-quantile(chain.all[,2],c(0.005,0.995))
curve(dnorm(x,mean=test1$betau_mean[2],sd=sqrt(test1$betau_var[2,2])),ylab='Density',main='',xlim=mlim,
lines(density(chain.all[,2]))
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5)
```

```
#Repeat for random effects.
for(i in 1:13){
mlim<-quantile(chain.all[,i+2],c(0.005,0.995))
curve(dnorm(x,mean=test1$betau_mean[i+2],sd=sqrt(test1$betau_var[i+2,i+2])),ylab='Density',main='',xlim=
lines(density(chain.all[,i+2]))
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5)
}

mlim<-quantile(chain.all[,16]^2,c(0.005,0.995))
curve(dgamma(1/x,shape=test1$tau_u[1],rate=test1$tau_u[2])*x^(-2),ylab='Density',main='',xlim=c(0,mlim[
lines(density(chain.all[,16]^2))
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5)

mlim<-quantile(chain.all[,17]^2,c(0.005,0.995))
curve(dgamma(1/x,shape=test1$tau_e[1],rate=test1$tau_e[2])*x^(-2),ylab='Density',main='',xlim=c(0,mlim[
lines(density(chain.all[,17]^2))
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5)
```



By determining the joint approximate posterior for $\boldsymbol{\beta}, \mathbf{u}$ rather than separate independent approximate posteriors for $\boldsymbol{\beta}$, $\mathbf{u}$ as in lecture 18, we improve the accuracy of approximate inference. This is because $\mathbf{X}$ and $\mathbf{Z}$ are not independent, and so contructing independent approximate posteriors for $\boldsymbol{\beta}$ and $\mathbf{u}$, we ignore the information contained in $\mathbf{X}'\mathbf{Z}$.