Lecture 20 R scripts

Example: Variational Bayes applied to farm data.

To illustrate Variational Bayes, we consider an agricultural dataset, which can be downloaded as `farmdata.csv` (for weaning weights) and `Kmat.csv` (for the kinship matrix) from Canvas. To benchmark Variational Bayes, we will compare results to those obtained using Gibbs sampling.

Writing a Gibbs sampler for this problem The Gibbs sampler for this problem is just a modification of the Gibbs sampler written in lab 7. Now however we can work with arbitrary variance-covariance matrices in the normal prior by including the argument Kinv. Kinv $= \mathbf{K}^{-1}$ is the inverse of the prior variance-covariance matrix up to proportionality.

```r
#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.
IO  <-diag(p+q)
diag(IO)[1:p]<-0
IO[-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*IO/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)
}
```

```
data<-read.csv('./farmdata.csv')
# data<-read.csv(file.choose()) #Choose farmdata.csv as downloaded from Canvas
K   <-read.csv('./Kmat.csv')
# K<-read.csv(file.choose()) #Choose farmdata.csv as downloaded from Canvas
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

```
system.time(chain1<-normalmm.Gibbs(iter=10000,Z=Z3,X=X,y=data$y,burnin=2000,taue_0=5,tauu_0=0.2,
                                   Kinv=Kinv,a.u=0.001,b.u=0.001,a.e=0.001,b.e=0.001))
```

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

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

```
##    user  system elapsed
##   1.741   0.029   1.772
```

```
system.time(chain2<-normalmm.Gibbs(iter=10000,Z=Z3,X=X,y=data$y,burnin=2000,taue_0=1,tauu_0=1,
                                   Kinv=Kinv,a.u=0.001,b.u=0.001,a.e=0.001,b.e=0.001))
```

```
##    user  system elapsed
##   1.683   0.017   1.700
```

```
system.time(chain3<-normalmm.Gibbs(iter=10000,Z=Z3,X=X,y=data$y,burnin=2000,taue_0=0.2,tauu_0=3,
                                   Kinv=Kinv,a.u=0.001,b.u=0.001,a.e=0.001,b.e=0.001))
```

```
##    user  system elapsed
##   1.692   0.019   1.712
```

```
library(coda)
```

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

```
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
```

Writing CAVI algorithm to find V-B estimates To determine the parameters of the Variational Bayes approximation, we will use an CAVI algorithm. This means we will succesively update $E(\boldsymbol{\beta}), \text{Var}(\boldsymbol{\beta}), E(\mathbf{u}), \text{Var}(\mathbf{u}), E(\tau_u), E(\tau_e)$ until convergence has been reached. In this code, convergence is defined as when $\sqrt{(\theta^{(t)} - \theta^{(t-1)})^2}/(|\theta^{(t)}| + 0.01) \leq \epsilon$ for all elements of $\boldsymbol{\theta}$. The addition of 0.01 is to avoid problems when $\theta^{(t)} = 0$

```
#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]
  ZTZ<-crossprod(Z)
  ZTY<-crossprod(Z,y)
  XTY<-crossprod(X,y)
  ZTX<-crossprod(Z,X)
  XTX<-crossprod(X)
  XTXinv<-solve(XTX)

  for(i in 1:iter){
  Vu <-solve(taue_0*ZTZ+tauu_0*Kinv) #update Var(u)
  u  <-taue_0*Vu%*%(ZTY-ZTX%*%beta0) #update E(u)
  Vb <-XTXinv/taue_0                 #update Var(beta)
  b  <-XTXinv%*%(XTY-t(ZTX)%*%u)     #update E(beta)
  TrKinvu <- sum(diag(Kinv%*%Vu))
  uKinvu  <- t(u)%*%Kinv%*%u
  tauu    <- (a.u+0.5*q)/(g.u+0.5*as.numeric(uKinvu)+0.5*TrKinvu)
  tauu    <- as.numeric(tauu)
  err     <- y - X%*%b - Z%*%u
  TrXTXb  <- sum(diag(XTX%*%Vb))
  TrZTZu  <- sum(diag(ZTZ%*%Vu))
  taue    <- (a.e+0.5*n)/(g.e+0.5*sum(err^2)+0.5*TrXTXb+0.5*TrZTZu)
  taue    <- as.numeric(taue)

  if(i > 1){
  diffb  <- sqrt((b-beta0)^2)/(abs(b)+0.01)
  diffu  <- sqrt((u-u0)^2)/(abs(u)+0.01)
  diffte <- abs(taue_0-taue)/(taue+0.01)
  difftu <- abs(tauu_0-tauu)/(tauu+0.01)
  diffvb <- sqrt((diag(Vb0) - diag(Vb))^2)/(diag(Vb))
  diffvu <- sqrt((diag(Vu0) - diag(Vu))^2)/(diag(Vu))
  diff.all<-c(diffb,diffu,diffte,difftu,diffvb,diffvu)
  if(max(diff.all) < epsilon) break
  }
  Vu0 <- Vu;u0<-u;Vb0<-Vb;beta0<-b;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*TrXTXb+0.5*TrZTZu))
  tauu.param<-c((a.u+0.5*q),(g.u+0.5*uKinvu+0.5*TrKinvu))
  param<-list(b,Vb,u,Vu,taue.param,tauu.param,i)
  names(param)<-c('beta_mean','beta_var','u_mean','u_var','tau_e','tau_u','iter')
  return(param)
}
```

```
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=rnorm(13),beta0=rnorm(2),a.e=0.001,g.e=0.001,a.u=0.001,g.u=0.001))
```

Estimating parameter of variational Bayes approximations

```
##    user  system elapsed
##   0.093   0.000   0.094
```

```
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(13),beta0=rnorm(2),a.e=0.001,g.e=0.001,a.u=0.001,g.u=0.001))
```

```
##    user  system elapsed
##   0.074   0.001   0.075
```

```
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(13),beta0=rnorm(2),a.e=0.001,g.e=0.001,a.u=0.001,g.u=0.001))
```

```
##    user  system elapsed
##   0.055   0.001   0.056
```

```
test1$iter
```

```
## [1] 1070
```

```
test2$iter
```

```
## [1] 1152
```

```
test3$iter
```

```
## [1] 870
```

```
#Comparing point estimates/posterior means

chain.all<-rbind(chain1$par,chain2$par,chain3$par)
#beta
cbind(test1$beta_mean,test2$beta_mean,test3$beta_mean,colMeans(chain.all[,1:2]))
```

Comparing estimates obtained from Gibbs sampling and Variational Bayes

```
##       [,1]     [,2]     [,3]     [,4]
## 1 4.149860 4.149861 4.149861 4.153120
## 2 4.708765 4.708766 4.708767 4.709322
```

```
#u
cbind(test1$u_mean,test2$u_mean,test3$u_mean,colMeans(chain.all[,3:15]))
```

```
##           [,1]        [,2]        [,3]        [,4]
## 1   0.80578542  0.80578469  0.80578421  0.80841251
## 2  -1.08621855 -1.08621928 -1.08621975 -1.09169227
## 3   0.21245313  0.21245245  0.21245200  0.21098242
## 4   0.47527965  0.47527943  0.47527928  0.47601605
## 5  -0.52479170 -0.52479204 -0.52479226 -0.53454094
## 6  -0.69870349 -0.69870374 -0.69870390 -0.70499465
## 7   0.50963975  0.50963940  0.50963918  0.51611965
## 8  -0.16302215 -0.16302238 -0.16302253 -0.16250230
## 9  -0.15022031 -0.15022065 -0.15022087 -0.14939405
## 10  0.12955976  0.12955948  0.12955930  0.13095244
## 11  0.14797212  0.14797175  0.14797150  0.15130713
## 12 -0.07160937 -0.07160964 -0.07160982 -0.07495579
## 13  0.57213111  0.57213069  0.57213042  0.57072883
```

```
#means of Variance parameters: E(\sigma_e^2), E(\sigma_u^2)
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,] 0.352975 0.4649196
## [2,] 0.352975 0.4649196
## [3,] 0.352975 0.4649196
```

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

```
##         sigmae2
## [1,] 0.05642792 0.05242605
## [2,] 0.05642792 0.05242605
## [3,] 0.05642792 0.05242605
```

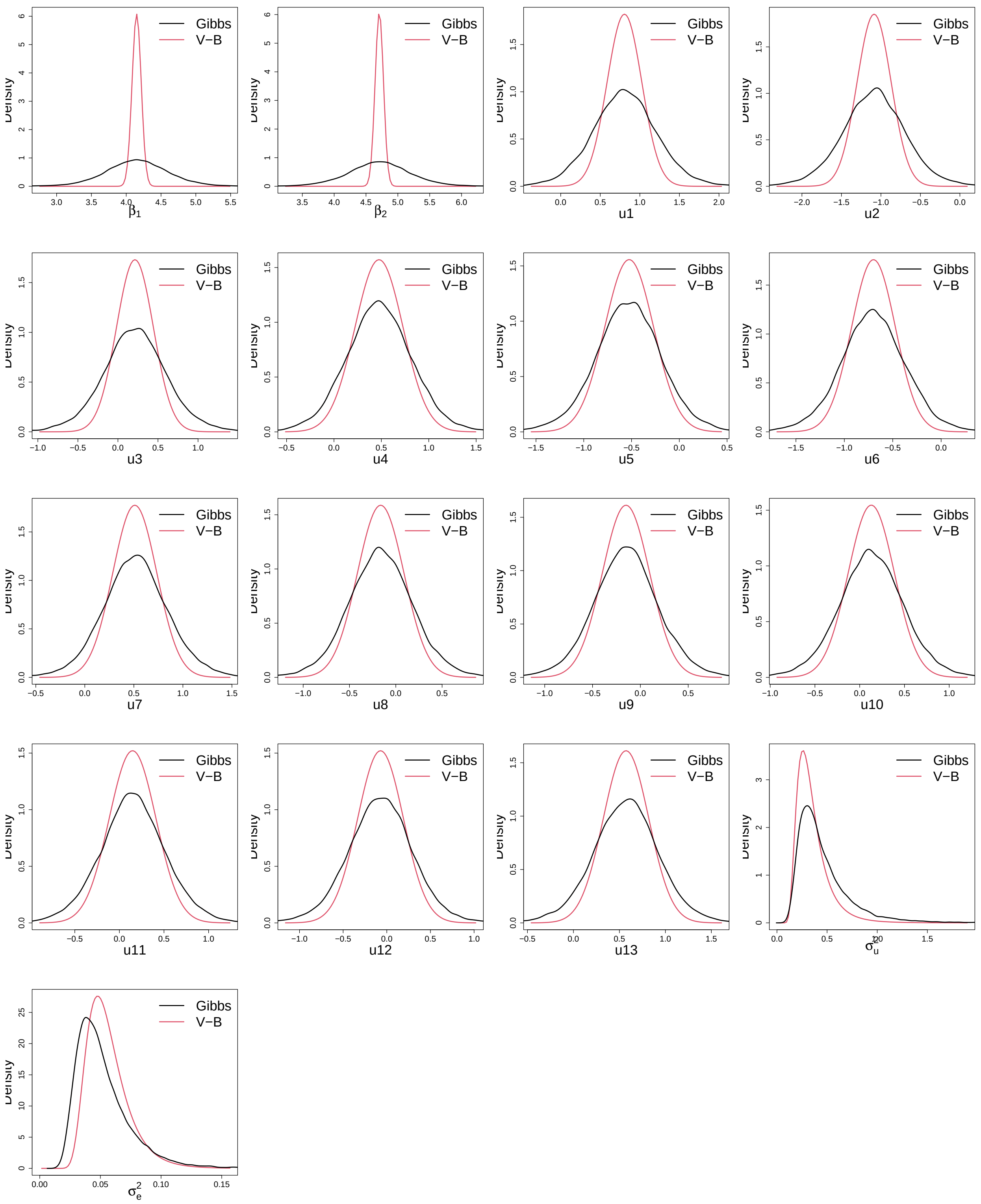
```
#Comparing posterior distributions.
par(mfrow=c(5,4))
mlim<-quantile(chain.all[,1],c(0.005,0.995))
curve(dnorm(x,mean=test1$beta_mean[1],sd=sqrt(test1$beta_var[1,1])),ylab='Density',main='',
      xlim=mlim,col=2,lty=1,xlab=expression(beta[1]),cex.lab=2.5,cex.axis=1.5, lwd=2)
lines(density(chain.all[,1]), lwd=2)
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5, lwd=2)

mlim<-quantile(chain.all[,2],c(0.005,0.995))
curve(dnorm(x,mean=test1$beta_mean[2],sd=sqrt(test1$beta_var[2,2])),ylab='Density',main='',
      xlim=mlim,col=2,lty=1,xlab=expression(beta[2]),cex.lab=2.5,cex.axis=1.5,lwd=2)
lines(density(chain.all[,2]),lwd=2)
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5,lwd=2)


#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$u_mean[i],sd=sqrt(test1$u_var[i,i])),ylab='Density',main='',
      xlim=mlim,col=2,lty=1,xlab=paste('u',i,sep=''),cex.lab=2.5,cex.axis=1.5,lwd=2)
lines(density(chain.all[,i+2]),lwd=2)
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5,lwd=2)
}

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[2]),col=2,lty=1,xlab=expression(sigma[u]^2),cex.lab=2.5,cex.axis=1.5,lwd=2)
lines(density(chain.all[,16]^2),lwd=2)
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5,lwd=2)

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='',lwd=2,
      xlim=c(0,mlim[2]),col=2,lty=1,xlab=expression(sigma[e]^2),cex.lab=2.5,cex.axis=1.5,cex.axis=1.5)
lines(density(chain.all[,17]^2),lwd=2)
legend('topright',legend=c('Gibbs','V-B'),col=1:2,lty=1,bty='n',cex=2.5,lwd=2)
```


Gibbs
V−B
Density
β_1
β_2
u1
u2
u3
u4
u5
u6
u7
u8
u9
u10
u11
u12
u13
σ_u^2
σ_e^2