SCRIPT MOD6S2B: NORMAL REGRESSION WITH CONJUGATE PRIORS

INSTRUCTOR: KLAUS MOELTNER

Simulate data

```
R> n<-10000 #sample size
R> x1<-rep(1,n)
R> x2<-rnorm(n,-1.4,1)
R> x3<-rnorm(n,3,2)
R> btrue<-c(1.2,0.4,0.8)
R> X<-cbind(x1,x2,x3)
R> k<-ncol(X)
R> sig2true<-1.44
R> eps<-rnorm(n,0,sqrt(sig2true))
R> y<-X %*% btrue + eps
```

Estimation

Define priors, derive posteriors, and compute the log-marginal likelihood:

```
R> #PRIORS:
R> ####################
R> #for beta:
R> mu0<-rep(0,k)
R> V0<-100*diag(k)
R> #for h:
R> v0<-10
R> tau0<-1
R> # get OLS results for convenient expressions of posterior
R> bols<-solve(t(X) %*% X) %*% (t(X) %*% y)
R> e<-y-X%*%bols
R> SSE<-(t(e)%*%e) #previously called "SSR"
R> #
R> # POSTERIORS:
R> ###################
R> V1<-solve(solve(V0)+ t(X) %*% X) # conditional posterior variance of beta
R> mu1<-V1 %*% (solve(V0) %*% mu0 + t(X) %*% y)# conditional posterior mean of beta
R> # last term is equivalent to X'X*bols
R> v1<-v0+n # posterior D.o.F. of h
R> tau1<-v1/(v0*(1/tau0)+SSE+t(bols-as.matrix(mu0))%*% solve(V0+solve(t(X) %*% X))
 %*%(bols-as.matrix(mu0)))
R> #
R> # log-p(y)
R> ###################
R> logpy<-lgamma(.5*v1) + .5*v0*log(v0*(1/tau0))- lgamma(.5*v0) -.5*n*log(pi)+
 .5*log(det(V1)/det(V0)) - .5*v1*log(v1*(1/tau1))
```

```
R> #
R> ttpost<-data.frame(col1=c("constant","x2","x3","h"),
                col2=c(btrue,1/sig2true),
                col3=c(mu1,tau1),
                col4=c(sqrt(2*tau1^2/v1),sqrt(diag((1/tau1[1,1])*V1))))
R> colnames(ttpost)<-c("variable","true values","post.mean","post.std.")
```

TABLE 1. Posterior results

variable	true values	post.mean	post.std.
constant	1.200	1.195	0.010
x2	0.400	0.401	0.027
x3	0.800	0.800	0.012
h	0.694	0.687	0.006

The log-marginal likelihood value is -16091.167.

Save results for later:

```
R> save(mu1,V1,tau1,v1,logpy,v0,tau0,mu0,V0,
 file = "c:/Klaus/AAEC5126/module6/NormalConjM1.rda")
```

PLOTS

Let's plot the prior vs. posterior for all parameters.

```
R> R<-10000
R> # Priors:
R> #################
R> hprior<-rgamma(R,v0/2,scale=(2*tau0/v0))
R> sig2prior<-1/hprior #get implied prior for error variance
R> betaprior<-matrix(0,k,R)
R> for (i in 1:R) {
 betaprior[,i]<-mvrnorm(n=1,mu0,sig2prior[i]*V0)
 }
R> #
R> # evaluate kernel densities (for plot)
R> hdens<-density(hprior,kernel="epanechnikov",n=1000)
R> sig2dens<-density(sig2prior,kernel="epanechnikov",n=1000)
R> beta2dens<-density(betaprior[2,],kernel="epanechnikov",n=1000)
R> beta3dens<-density(betaprior[3,],kernel="epanechnikov",n=1000)
R> # Posteriors
R> #################
R> hpost<-rgamma(R,v1/2,scale=(2*tau1/v1))
R> sig2post<-1/hpost #get implied prior for error variance
R> # we need draws from the multivariate t-density for beta
R> ##########################################################
R> # useful function to replicate a vector or matrix multiple times - used further
R> # below
R> repmat = function(X,m,n){
   ##R equivalent of repmat (matlab)
```

```
   mx = dim(X)[1]
   nx = dim(X)[2]
   matrix(t(matrix(X,mx,nx*n)),mx*m,nx*n,byrow=T)
 }
R> Vbeta<-(1/tau1[1,1])*V1     #posterior variance-cov. matrix of beta
R> #C1<-cov2cor(Vbeta)          #convert to correlation matrix
R> #std1<-matrix(sqrt(diag(Vbeta)))     #collect standard deviations
R> mu1mat<-repmat(t(mu1),R,1) #R by k matrix with mu1 on each row
R> betapost<-mu1mat + rmvt(R, sigma=Vbeta, df=v1)
R> #rmvt draws from a mvt with mean vector zero, so we need to add mu1
R> #for each draw
R> # this produces an R by k matrix of draws of beta, one draw per row
R> # you can verify, using "colMeans(betapost)" and "cov(betapost)" that
R> # empirical mean vector is (almost) identical to the analytical
R> # solution for mu1, and the empirical covariance matrix is (virtually)
R> # identical to the ananlytical solution (1/tau1)V1
R> # evaluate kernel densities (for plot)
R> hpostdens<-density(hpost,kernel="epanechnikov",n=1000)
R> sig2postdens<-density(sig2post,kernel="epanechnikov",n=1000)
R> beta2postdens<-density(betapost[,2],kernel="epanechnikov",n=1000)
R> beta3postdens<-density(betapost[,3],kernel="epanechnikov",n=1000)

R> proc.time()-tic
   user  system elapsed
   3.07    0.13    3.29
```

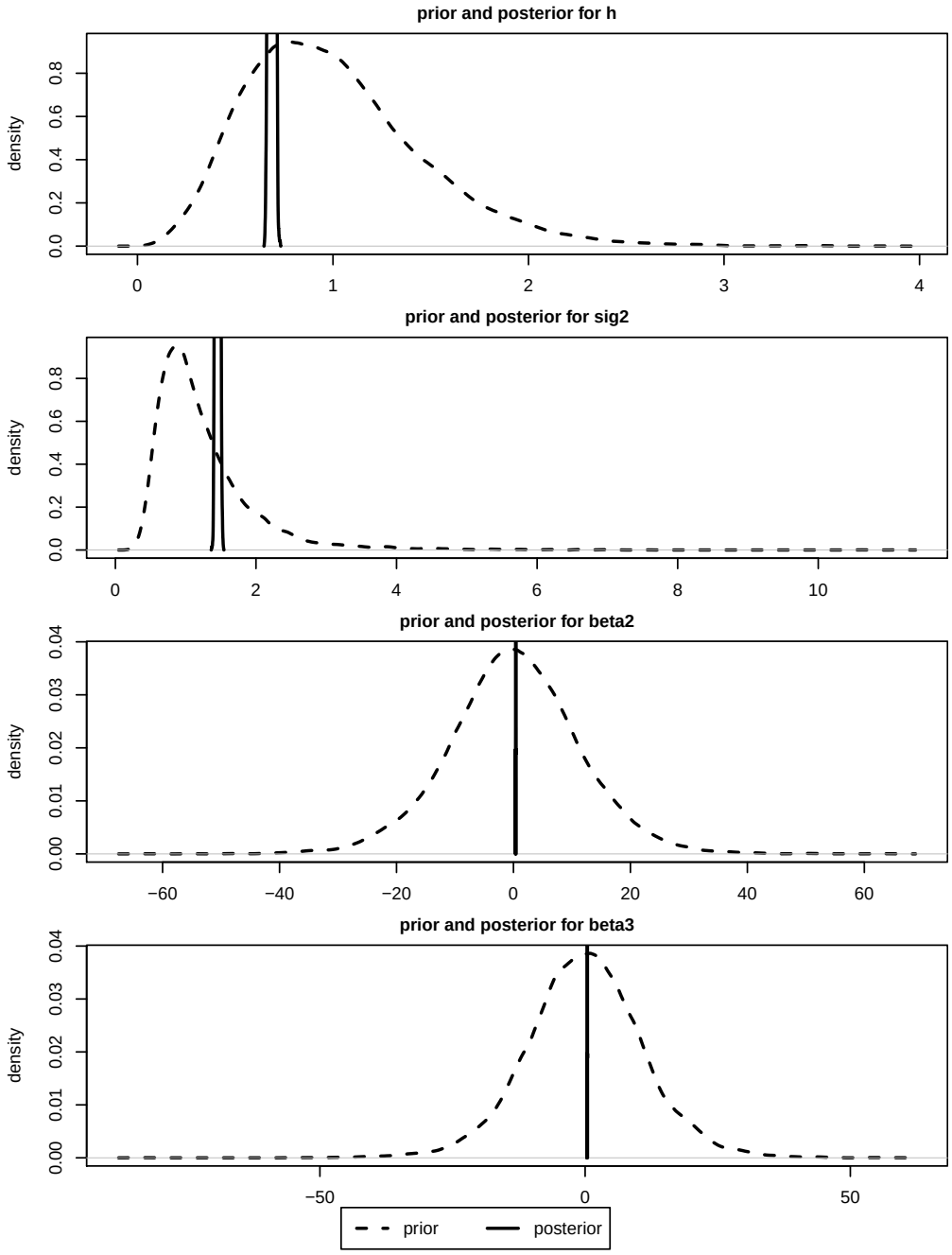

FIGURE 1. Prior vs. Posterior plots