SCRIPT MOD6S3B: NORMAL REGRESSION WITH INDEPENDENT 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
```

Starting values, priors, and tuners

For now consider "tuners" any other values that have to be set before running the Gibbs Sampler, such as the number of burn-ins and the number of keepers.

```
R> #TUNERS
R> ###############
R> r1<-5000  #burn-ins
R> r2<-10000 #keepers
R> R<-r1+r2
R> # of Gibbs Sampler
R> #
R> #PRIORS:
R> ####################
R> #for beta:
R> mu0<-rep(0,k)
R> V0<-100*diag(k)
R> #for sig2:
R> v0<-0.5
R> tau0<-0.5
R> #
R> # STARTING VALUES
R> ######################
R> bols<-solve(t(X) %*% X) %*% (t(X) %*% y)
R> e<-y-X%*%bols
R> s2<-(t(e)%*%e)/(n-k)
```

```
R> betadraw<-bols
R> sig2draw<-as.vector(s2)
```

Gibbs Sampler

```
R> betamat<-matrix(0,k,r2)  #will collect draws of beta
R> sig2mat<-matrix(0,1,r2)  #will collect draws of sig2
R> #
R> # Call for a progress bar to monitor progress of Gibbs Sampler
R> pb<-winProgressBar(title="progress bar", min=0,max=R,width= 300)
R> #
R> for (i in 1:R) {
    setWinProgressBar(pb,i,title=paste(round((i/R)*100,0),"% done"))
 #
 # draw betas
 ###############
     V1<-solve(solve(V0)+(1/sig2draw)*t(X)%*%X)
     mu1<-V1 %*% (solve(V0) %*% mu0 + (1/sig2draw)* t(X) %*% y)
     betadraw<-mvrnorm(n=1,mu1,V1)
     if (i>r1)  {
        betamat[,(i-r1)]<-betadraw
     }
 # draw sig2
 ###############
     v1<-(n+2*v0)/2
     tau1<-(1/2)*(t(y-X %*% betadraw) %*% (y-X %*% betadraw)+2*tau0)
     sig2draw<-rinvgamma(1,v1,scale=tau1)
     if (i>r1)  {
        sig2mat[i-r1]<-sig2draw
     }
 }
R> close (pb)  #close progress bar
```

Convergence Diagnostics

```
R> #stack all results
R> M<-rbind(betamat,sig2mat)
R> k<-nrow(M)
R> R<-ncol(M)
R> # Autocorrelation, nse, and efficiency
R> ########################################
R> IEF<-matrix(0,k,1)
R> nse<-matrix(0,k,1)
R> #
R> for (i in 1:k) {
    int<-t(M[i,]) # pick draws for a single parameter
    intsum<-0
    j=1
    for (j in 1:(R-2)) {
```

```
        int1<-int[1:(R-j)]
        int2<-int[(j+1):R]
        int3=cor(int1,int2)
        intsum<-intsum+(1-j/R)*int3
        if (abs(int3)<0.05)  {
            break # exit innermost loop
             }
    }
    intsum<-max(0,intsum)#for situations when the loop immediately cuts off
    # with a slight negative value for the correlation
    IEF[i]=1+2*intsum
 }
R> #
R> nse<-sqrt(((1/R)*(diag(var(t(M)))*IEF)))
R> mstar<-R/IEF
R> #
R> # CD diagnostics
R> ##################
R> g1<-round(0.1*R)
R> g2=round(0.6*R)+1
R> M1<-M[,1:g1]
R> M2=M[,g2:R]
R> R1<-ncol(M1)
R> R2=ncol(M2)
R> mM1=rowMeans(M1)
R> mM2=rowMeans(M2)
R> #
R> IEF1<-matrix(0,k,1)
R> nse1<-matrix(0,k,1)
R> #
R> for (i in 1:k) {
    int<-t(M1[i,]) # pick draws for a single parameter
    intsum<-0
    j=1
    for (j in 1:(R1-2)) {
        int1<-int[1:(R1-j)]
        int2<-int[(j+1):R1]
        int3=cor(int1,int2)
        intsum<-intsum+(1-j/R1)*int3
        if (abs(int3)<0.05)  {
            break # exit innermost loop
             }
    }
    intsum<-max(0,intsum)#for situations when the loop immediately cuts off
    # with a slight negative value for the correlation
    IEF1[i]=1+2*intsum
 }
```

```
R> nse1<-sqrt(((1/R1)*(diag(var(t(M1)))*IEF1)))
R> #
R> #
R> IEF2<-matrix(0,k,1)
R> nse2<-matrix(0,k,1)
R> #
R> for (i in 1:k) {
    int<-t(M2[i,]) # pick draws for a single parameter
    intsum<-0
    j=1
    for (j in 1:(R2-2)) {
        int1<-int[1:(R2-j)]
        int2<-int[(j+1):R2]
        int3=cor(int1,int2)
        intsum<-intsum+(1-j/R2)*int3
        if (abs(int3)<0.05)  {
            break # exit innermost loop
             }
    }
    intsum<-max(0,intsum)#for situations when the loop immediately cuts off
    # with a slight negative value for the correlation
    IEF2[i]=1+2*intsum
 }
R> nse2<-sqrt(((1/R2)*(diag(var(t(M2)))*IEF1)))
R> #
R> CD<-(mM1-mM2)/(sqrt(nse1^2+nse2^2))
R> diagnostics<-cbind(rowMeans(M), apply(M,1,sd), nse, IEF, mstar, CD)
```

Output Table

```
R> ttpost<-data.frame(col1=c("constant","x2","x3","sig2"),
                col2=diagnostics[,1],
                col3=diagnostics[,2],
                col4=diagnostics[,3],
                col5=diagnostics[,4],
                col6=diagnostics[,5],
                col7=diagnostics[,6])
R> colnames(ttpost)<-c("variable","post. mean","post.std","nse","IEF","M*","CD")
```

Table 1. Posterior results

variable	post. mean	post.std	nse	IEF	M*	CD
constant	1.195	0.027	0.000	1.023	9776.960	-0.190
x2	0.401	0.012	0.000	1.002	9981.514	-0.984
x3	0.800	0.006	0.000	1.000	10000.000	-0.665
sig2	1.456	0.020	0.000	1.035	9662.433	0.985

Plots

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

```
R> R<-r2
R> # Priors:
R> ##################
R> sig2prior<-rinvgamma(R,v0,scale=tau0)
R> betaprior<-mvrnorm(n=R,mu0,V0)
R> #
R> # evaluate kernel densities (for plot)
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> sig2postdens<-density(sig2mat,kernel="epanechnikov",n=1000)
R> beta2postdens<-density(betamat[2,],kernel="epanechnikov",n=1000)
R> beta3postdens<-density(betamat[3,],kernel="epanechnikov",n=1000)

R> proc.time()-tic
   user  system elapsed
  13.91    3.01   18.91
```

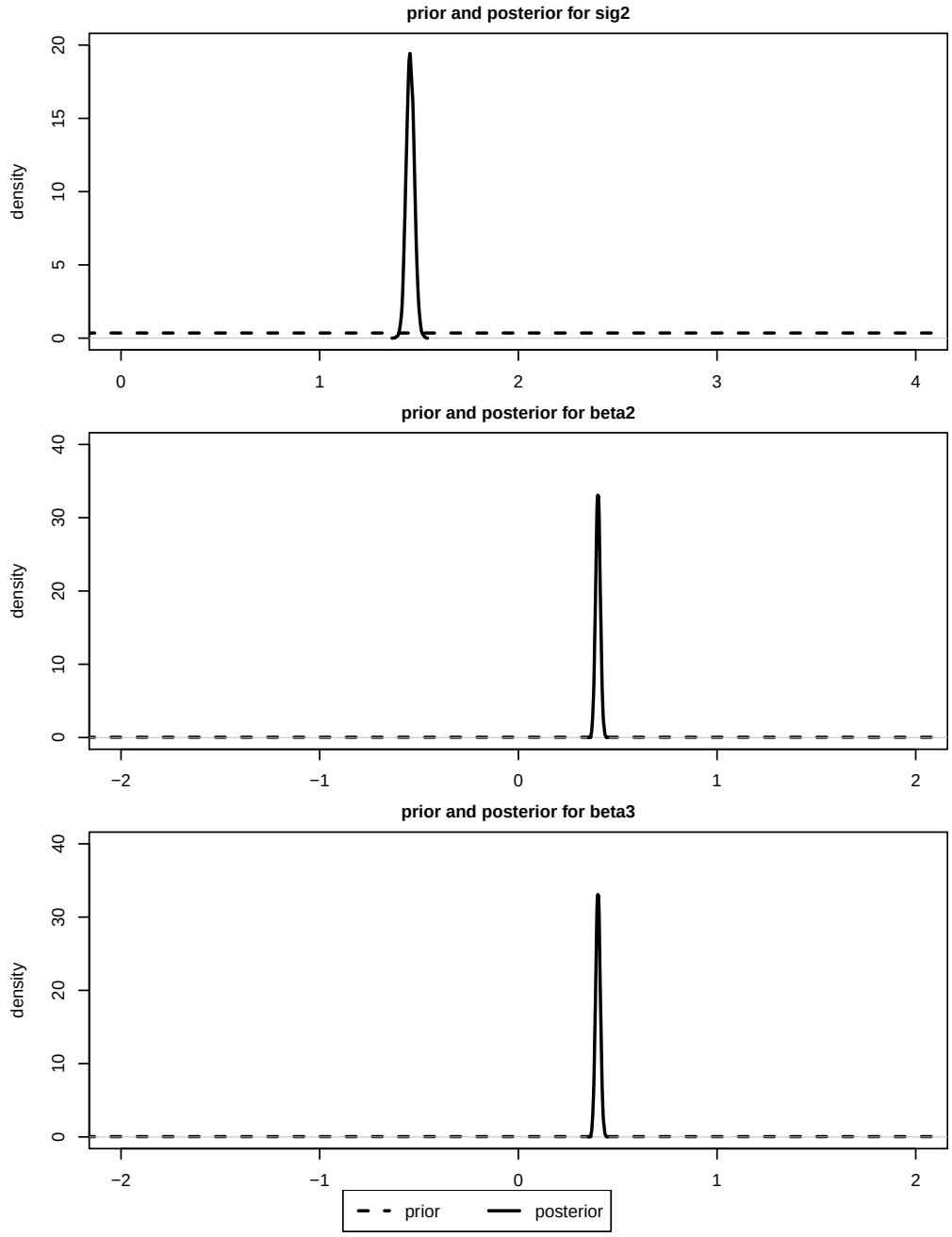

FIGURE 1. Prior vs. Posterior plots