SCRIPT MOD6S3G: NORMAL REGRESSION WITH INDEPENDENT PRIORS
LABOR DATA APPLICATION - EFFECT OF BLOCKING

INSTRUCTOR: KLAUS MOELTNER

Load and Prepare Data

```
R> data<- read.table('c:/Klaus/AAEC5126/R/data/laborsupply.txt', sep="\t", header=FALSE)
R> #
R> #assign variable names
R> names(data)[1]<-"lfp"
R> names(data)[2]<-"whrs"
R> names(data)[3]<-"kl6"
R> names(data)[4]<-"k618"
R> names(data)[5]<-"wa"
R> names(data)[6]<-"we"
R> names(data)[7]<-"ww"
R> names(data)[8]<-"rpwg"
R> names(data)[9]<-"hhrs"
R> names(data)[10]<-"ha"
R> names(data)[11]<-"he"
R> names(data)[12]<-"hw"
R> names(data)[13]<-"faminc"
R> names(data)[14]<-"wmed"
R> names(data)[15]<-"wfed"
R> names(data)[16]<-"un"
R> names(data)[17]<-"cit"
R> names(data)[18]<-"ax"
R> #
R> save(data, file = "c:/Klaus/AAEC5126/R/data/laborsupply.rda")
```

Variable definitions:

```
% Contents of Data (columns)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1       LFP = A dummy variable = 1 if woman worked in 1975, else 0
%2       WHRS = Wife's hours of work in 1975
%3       KL6 = Number of children less than 6 years old in household
%4       K618 = Number of children between ages 6 and 18 in household
%5       WA = Wife's age
%6       WE = Wife's educational attainment, in years
%7       WW = Wife's average hourly earnings, in 1975 dollars
%8       RPWG = Wife's 1976 reported wage (not = 1975 estimated wage)
%9       HHRS = Husband's hours worked in 1975
%10      HA = Husband's age
%11      HE = Husband's educational attainment, in years
%12      HW = Husband's wage, in 1975 dollars
```

```
%13      FAMINC = Family income, in 1975 dollars
%14      WMED = Wife's mother's educational attainment, in years
%15      WFED = Wife's father's educational attainment, in years
%16      UN = Unemployment rate in county of residence, in percentage points.
%17      CIT = Dummy variable = 1 if live in large city (SMSA), else 0
%18      AX = Actual years of wife's previous labor market experience


R> #select only cases for labor market participants
R> data<-subset(data,lfp==1)
R> attach(data)
R> #
R> #define variables of interest
R> y=log(whrs*ww) #log of annual earnings
R> n<-length(y)
R> wa2<-wa^2
R> X<-cbind(rep(1,n),wa, wa2, kl6, k618, hw, ax, we)
R> k<-ncol(X)
R> #
R> bols<-solve((t(X)) %*% X) %*% (t(X) %*% y)
R> e<-y-X%*%bols
R> SSR<-(t(e)%*%e)
R> s2<-(t(e)%*%e)/(n-k)
R> s2ols<-s2 #needed for Hausman test below
R> Vb<-s2[1,1]*solve((t(X))%*%X)
R> se=sqrt(diag(Vb))
R> tval=bols/se
R> #
R> tt<-data.frame(col1=c("constant","wa","wa2","kl6","k618","hw","ax","we"),
                 col2=bols,
                 col3=se,
                 col4=tval)
R> colnames(tt)<-c("variable","estimate","s.e.","t")
```

Starting values, priors, and tuners

I contrast to script `mod6_3e` we implement the GS with 4 blocks.

```
R> X1<-X[,1:3]
R> X2<-X[,4:6]
R> X3=X[,7:8]
R> k1<-ncol(X1)
R> k2<-ncol(X2)
R> k3<-ncol(X3)
R> k<-k1+k2+k3
R> #TUNERS
R> ###############
R> r1<-2000  #burn-ins
R> r2<-3000 #keepers
R> R<-r1+r2
R> # of Gibbs Sampler
R> #
```

```
R> #PRIORS:
R> ###################
R> #for beta:
R> mu01<-rep(0,k1)
R> V01<-100*diag(k1)
R> mu02<-rep(0,k2)
R> V02<-100*diag(k2)
R> mu03<-rep(0,k3)
R> V03<-100*diag(k3)
R> #for sig2:
R> v0<-1/2
R> tau0<-1/2
R> #
R> # STARTING VALUES
R> ######################
R> beta1draw<-bols[1:k1]
R> beta2draw<-bols[(k1+1):(k1+k2)]
R> beta3draw<-bols[(k1+k2+1):k]
R> sig2draw<-as.vector(s2)
```

Gibbs Sampler

```
R> beta1mat<-matrix(0,k1,r2)  #will collect draws of beta1
R> beta2mat<-matrix(0,k2,r2)  #will collect draws of beta2
R> beta3mat<-matrix(0,k3,r2)  #will collect draws of beta3
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 beta1
 ###############
     V1<-solve(solve(V01)+(1/sig2draw)*t(X1)%*%X1)
     mu1<-V1 %*% (solve(V01) %*% mu01 + (1/sig2draw)* t(X1) %*%
     (y-X2 %*% beta2draw - X3 %*% beta3draw))
     beta1draw<-mvrnorm(n=1,mu1,V1)
     if (i>r1)  {
        beta1mat[,(i-r1)]<-beta1draw
     }
 # draw beta2
 ###############
     V1<-solve(solve(V02)+(1/sig2draw)*t(X2)%*%X2)
     mu1<-V1 %*% (solve(V02) %*% mu02 + (1/sig2draw)* t(X2) %*%
     (y-X1%*%beta1draw-X3%*%beta3draw))
     beta2draw<-mvrnorm(n=1,mu1,V1)
     if (i>r1)  {
        beta2mat[,(i-r1)]<-beta2draw
     }
```

```
 # draw beta3
 ##############
     V1<-solve(solve(V03)+(1/sig2draw)*t(X3)%*%X3)
     mu1<-V1 %*% (solve(V03) %*% mu03 + (1/sig2draw)* t(X3) %*%
     (y-X2%*%beta2draw-X1%*%beta1draw))
     beta3draw<-mvrnorm(n=1,mu1,V1)
     if (i>r1)  {
        beta3mat[,(i-r1)]<-beta3draw
     }
 # draw sig2
 ##############
     betadraw<-c(beta1draw,beta2draw,beta3draw)
     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(beta1mat,beta2mat,beta3mat,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 Tables

Table 1. OLS output

variable	estimate	s.e.	t
constant	4.287	1.683	2.547
wa	0.151	0.079	1.911
wa2	-0.002	0.001	-2.326
kl6	-0.620	0.148	-4.192
k618	-0.116	0.047	-2.475
hw	0.005	0.016	0.328
ax	0.052	0.008	6.614
we	0.067	0.025	2.673

The OLS-estimated error variance is 1.2104.

Table 2. Posterior results

variable	post. mean	post.std	nse	IEF	M*	CD
constant	4.190	1.665	0.034	1.239	2421.055	0.538
wa	0.153	0.079	0.002	1.177	2549.144	-0.459
wa2	-0.002	0.001	0.000	1.192	2517.394	0.466
kl6	-0.625	0.149	0.003	1.343	2233.487	-0.520
k618	-0.114	0.047	0.002	7.761	386.547	-1.864
hw	0.005	0.015	0.001	9.882	303.584	-1.085
ax	0.053	0.008	0.000	4.491	667.995	-0.838
we	0.071	0.022	0.002	35.566	84.350	0.315
sig2	1.215	0.085	0.002	1.064	2819.165	0.080

ACplots

```
R> #Focus on "kl6" coefficient
R> ##############################
R> d<-50; # we're only interested into lags up to 50
R> int<-t(beta2mat[1,])
R> m<-length(int)
R> lagcorrkl6<-matrix(0,1,d)
```

```
R> j<-1
R> for (j in 1:d)    {
     int1<-int[1:(m-j)]
     int2<-int[(j+1):m]
     int3<-cor(int1,int2)
     lagcorrkl6[j]<-int3
 }
R> #
R> #
R> #Focus on "we" coefficient
R> ##############################
R> d<-50; # we're only interested into lags up to 50
R> int<-t(beta3mat[2,])
R> m<-length(int)
R> lagcorrwe<-matrix(0,1,d)
R> j<-1
R> for (j in 1:d)    {
     int1<-int[1:(m-j)]
     int2<-int[(j+1):m]
     int3<-cor(int1,int2)
     lagcorrwe[j]<-int3
 }
R> #
R> #
R> #Focus on "sig2" coefficient
R> ##############################
R> d<-50; # we're only interested into lags up to 50
R> int<-t(sig2mat)
R> m<-length(int)
R> lagcorrsig2<-matrix(0,1,d)
R> j<-1
R> for (j in 1:d)    {
     int1<-int[1:(m-j)]
     int2<-int[(j+1):m]
     int3<-cor(int1,int2)
     lagcorrsig2[j]<-int3
 }
```

Gibbs Sampler, keep all

```
R> r1<-0
R> r2<-5000
R> R<-r1+r2
R> beta1mat<-matrix(0,k1,r2)  #will collect draws of beta1
R> beta2mat<-matrix(0,k2,r2)  #will collect draws of beta2
R> beta3mat<-matrix(0,k3,r2)  #will collect draws of beta3
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> #
```

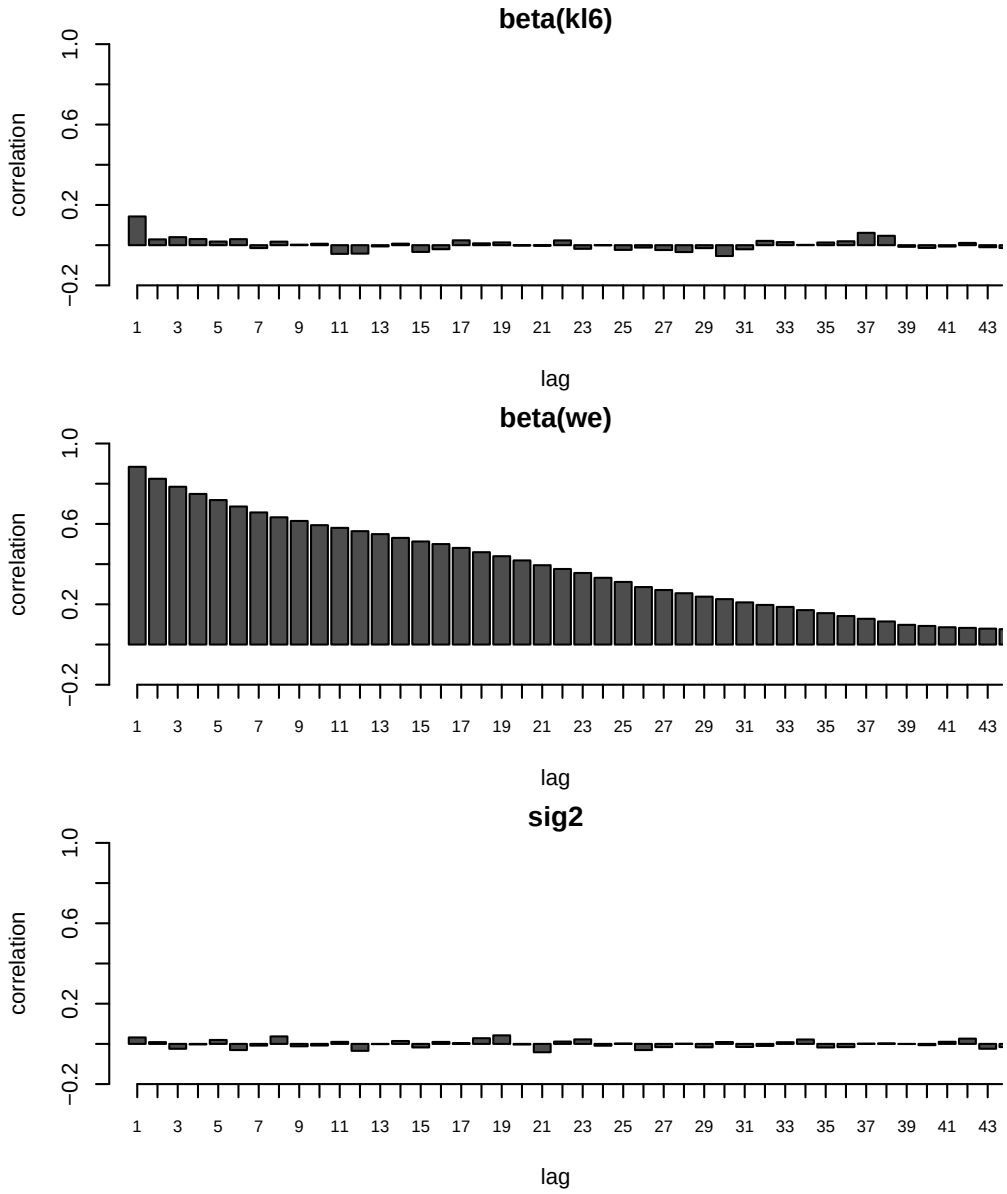

FIGURE 1. AC plots

```
R> for (i in 1:R) {
    setWinProgressBar(pb,i,title=paste(round((i/R)*100,0),"% done"))
 #
 # draw beta1
 ###############
     V1<-solve(solve(V01)+(1/sig2draw)*t(X1)%*%X1)
     mu1<-V1 %*% (solve(V01) %*% mu01 + (1/sig2draw)* t(X1) %*%
     (y-X2 %*% beta2draw - X3 %*% beta3draw))
     beta1draw<-mvrnorm(n=1,mu1,V1)
     if (i>r1)  {
```

```
        beta1mat[,(i-r1)]<-beta1draw
    }
 # draw beta2
 ###############
    V1<-solve(solve(V02)+(1/sig2draw)*t(X2)%*%X2)
    mu1<-V1 %*% (solve(V02) %*% mu02 + (1/sig2draw)* t(X2) %*%
    (y-X1%*%beta1draw-X3%*%beta3draw))
    beta2draw<-mvrnorm(n=1,mu1,V1)
    if (i>r1)  {
       beta2mat[,(i-r1)]<-beta2draw
    }
 # draw beta3
 ###############
    V1<-solve(solve(V03)+(1/sig2draw)*t(X3)%*%X3)
    mu1<-V1 %*% (solve(V03) %*% mu03 + (1/sig2draw)* t(X3) %*%
    (y-X2%*%beta2draw-X1%*%beta1draw))
    beta3draw<-mvrnorm(n=1,mu1,V1)
    if (i>r1)  {
       beta3mat[,(i-r1)]<-beta3draw
    }
 # draw sig2
 ###############
    betadraw<-c(beta1draw,beta2draw,beta3draw)
    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 Plots

```
R> proc.time()-tic
   user  system elapsed
  16.82    3.51   23.17
```

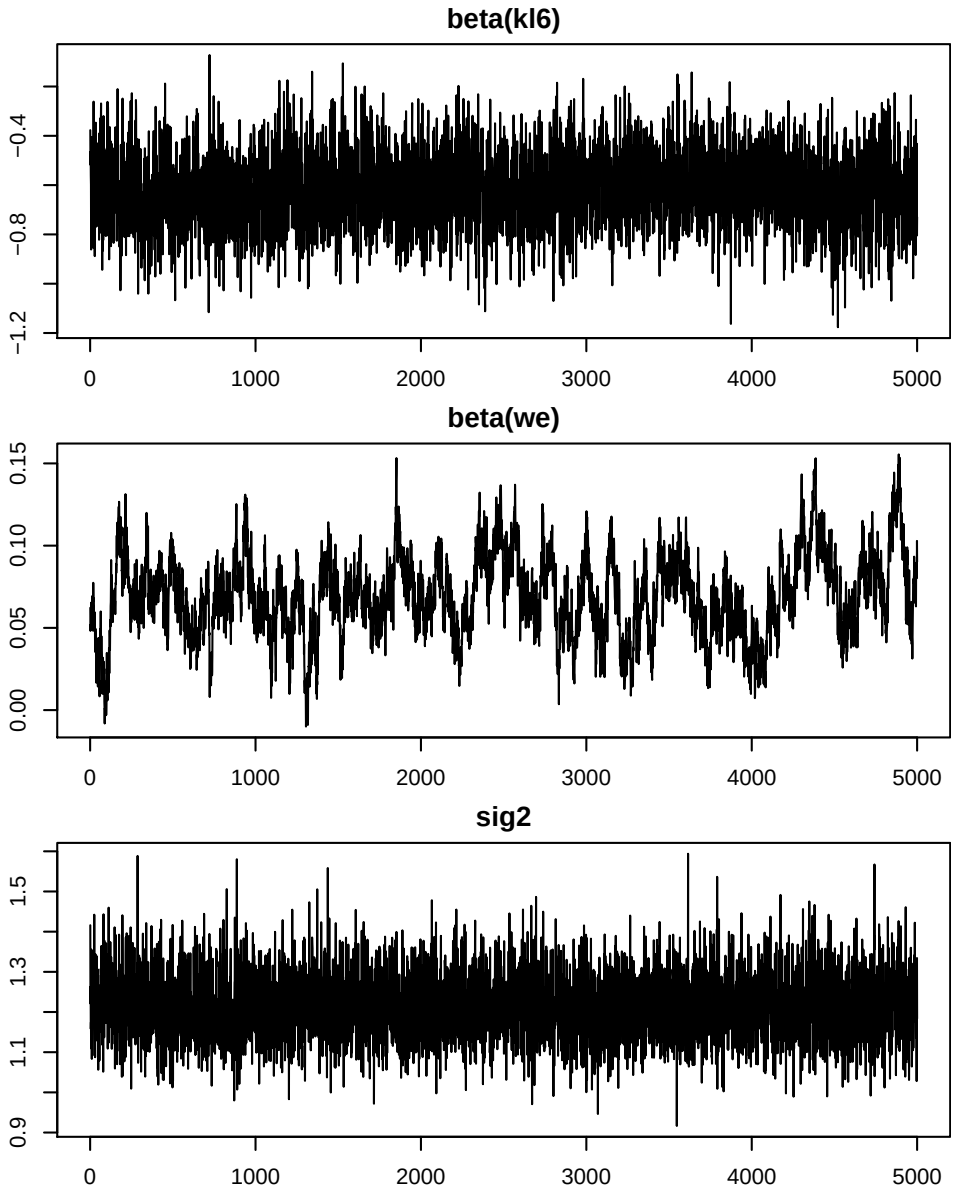

FIGURE 2. Convergence plots under inefficient blocking