MATH 598: TOPICS IN STATISTICS

MCMC FOR HIERARCHICAL MODELS

The following Bayesian hierarchical model is used to describe the variation of health outcomes of four different types of surgery in six hospitals. Let $i = 1, \ldots, 6$ denote hospital, and $j = 1, \ldots, 4$ denote surgery type. Then, let m_{ij} denote the number of surgeries of type j at hospital i, and y_{ij} denote the number of these surgeries that were successful (that is, surgeries for which the patient was not re-hospitalized in the thirty days after the surgery took place. The following data are to be studied.

		Surgery Type							
		1		2		3		4	
		y	m	y	m	y	m	y	m
Hospital	1	1	49	7	30	1	29	3	26
	2	11	35	2	17	6	22	1	32
	3	14	26	30	43	15	29	15	31
	4	16	26	23	25	22	34	32	37
	5	29	34	21	25	22	28	16	30
	6	15	30	20	33	13	34	12	39

- STAGE 1: Observed data

$$Y_{ij} \sim Binomial(m_{ij}, \theta_{ij}) \qquad i = 1, \ldots, 6, j = 1, \ldots, 4 \text{ independently}$$

- STAGE 2: Log-odds model

$$\psi_{ij} = \log\left(\frac{\theta_{ij}}{1-\theta_{ij}}\right) \sim Normal(\alpha_i + \mu_j, \sigma_j^2) \qquad i = 1, \ldots, 6, j = 1, \ldots, 4 \text{ independently}$$

- STAGE 3: Prior model

$$\alpha_i \sim Normal(0, 5^2) \qquad i = 1, \ldots, 6 \text{ independently}$$

$$\mu_j \sim Normal(0, 5^2) \qquad j = 1, \ldots, 4 \text{ independently}$$

$$\sigma_j^2 \sim InvGamma(5, 2.5) \qquad j = 1, \ldots, 4 \text{ independently}$$

In this model, the parameter vector comprises 38 parameters

$$\{\psi_{ij}, i = 1, \ldots, 6, j = 1, \ldots, 4\}, \{\alpha_i, i = 1, \ldots, 6\}, \{\mu_j, j = 1, \ldots, 4\}, \{\sigma_j^2, j = 1, \ldots, 4\}.$$

and it can be seen that the posterior, up to proportionality, factorizes as follows

$$\pi_n(\psi, \alpha, \mu, \sigma^2) \propto \prod_{i=1}^{6}\prod_{j=1}^{4} f(y_{ij}; m_{ij}, \psi_{ij}) \qquad \text{Likelihood}$$

$$\times \prod_{i=1}^{6}\prod_{j=1}^{4} \pi_0^{(1)}(\psi_{ij}|\alpha_i, \mu_j, \sigma_j^2) \qquad \text{First stage prior}$$

$$\times \prod_{i=1}^{6} \pi_0^{(2)}(\alpha_i) \qquad \text{Second stage prior}$$

$$\times \prod_{j=1}^{4} \pi_0^{(2)}(\mu_j) \qquad \text{Second stage prior}$$

$$\times \prod_{j=1}^{4} \pi_0^{(2)}(\sigma_j^2) \qquad \text{Second stage prior}$$

This is high-dimensional posterior distribution that cannot be analyzed directly. However, using a Gibbs sampler strategy, the posterior can be sampled and summarized.

Algorithm 1: Single parameter Gibbs sampler.
In this algorithm, the 38 full conditional posterior distributions are sampled individually in a sweep. However, the intrinsic conditional independence structure of the model simplifies the sampling

- $\psi_{ij}, i = 1, \ldots, 6, j = 1, \ldots 4$: The full conditional for ψ_{ij} is not available in closed form, but up to proportionality is given by

$$\pi_n(\psi_{ij}|-) \propto \exp\{y_{ij}\psi_{ij} - m_{ij}\log(1+\exp\{\psi_{ij}\})\exp\left\{-\frac{1}{2\sigma_j^2}(\psi_{ij}-\alpha_i-\mu_j)^2\right\}$$

  For the Gibbs sampler update of ψ_{ij} we use a Metropolis-within-Gibbs step, and propose an update using a Normal proposal mechanism with variance $\sigma_\psi = 0.5$. Note that, conditional on the collection of α, μ and σ^2 parameters, the ψ_{ij} parameters are independent, so that they can be updated in parallel.

- $\alpha_i, i = 1, \ldots, 6$: At the second stage of the hierarchy, notice that the second stage parameters are **conditionally independent of the data** given the ψ parameters. We have that for $i = 1, \ldots, 6$

$$\begin{aligned}\pi_n(\alpha_i|-) &\propto \prod_{j=1}^{4}\pi_0^{(1)}(\psi_{ij}|\alpha_i,\mu_j,\sigma_j^2)\times\pi_0^{(2)}(\alpha_i)\\ &\propto \exp\left\{-\frac{1}{2}\sum_{j=1}^{4}\frac{1}{\sigma_j^2}(\psi_{ij}-\alpha_i-\mu_j)^2\right\}\exp\left\{-\frac{\alpha_i^2}{2\times 5^2}\right\}\end{aligned}$$

  After some algebra, we see that the full conditional for α_i takes the form

$$\pi_n(\alpha_i|-) \equiv Normal(m_{\alpha_i}, M_{\alpha_i})$$

  where

$$m_{\alpha_i} = \frac{\sum_{j=1}^{4}\frac{(\psi_{ij}-\mu_j)}{\sigma_j^2}}{\sum_{j=1}^{4}\frac{1}{\sigma_j^2}+\frac{1}{5^2}} \qquad M_{\alpha_i} = \frac{1}{\sum_{j=1}^{4}\frac{1}{\sigma_j^2}+\frac{1}{5^2}}.$$

- $\mu_j, j = 1, \ldots, 4$: As above, we have that for $j = 1, \ldots, 4$

$$\begin{aligned}\pi_n(\mu_j|-) &\propto \prod_{i=1}^{6}\pi_0^{(1)}(\psi_{ij}|\alpha_i,\mu_j,\sigma_j^2)\times\pi_0^{(2)}(\mu_j)\\ &\propto \exp\left\{-\frac{1}{2\sigma_j^2}\sum_{i=1}^{6}(\psi_{ij}-\alpha_i-\mu_j)^2\right\}\exp\left\{-\frac{\mu_j^2}{2\times 5^2}\right\}\end{aligned}$$

  After some algebra, we see that the full conditional for μ_j takes the form

$$\pi_n(\mu_j|-) \equiv Normal(m_{\mu_j}, M_{\mu_j})$$

  where

$$m_{\mu_j} = \frac{\frac{1}{\sigma_j^2}\sum_{i=1}^{6}(\psi_{ij}-\alpha_i)}{\frac{6}{\sigma_j^2}+\frac{1}{5^2}} \qquad M_{\mu_j} = \frac{1}{\frac{6}{\sigma_j^2}+\frac{1}{5^2}}.$$

- $\sigma_j^2, j = 1, \ldots, 4$: We have that for $j = 1, \ldots, 4$

$$\pi_n(\sigma_j^2|-) \propto \prod_{i=1}^{6} \pi_0^{(1)}(\psi_{ij}|\alpha_i, \mu_j, \sigma_j^2) \times \pi_0^{(2)}(\sigma_j^2)$$

$$\propto \left(\frac{1}{\sigma_j^2}\right)^{6/2} \exp\left\{-\frac{1}{2\sigma_j^2}\sum_{i=1}^{6}(\psi_{ij} - \alpha_i - \mu_j)^2\right\} \left(\frac{1}{\sigma_j^2}\right)^{10/2+1} \exp\left\{-\frac{5}{2\sigma_j^2}\right\}$$

We then see that the full conditional for σ_j^2 takes the form

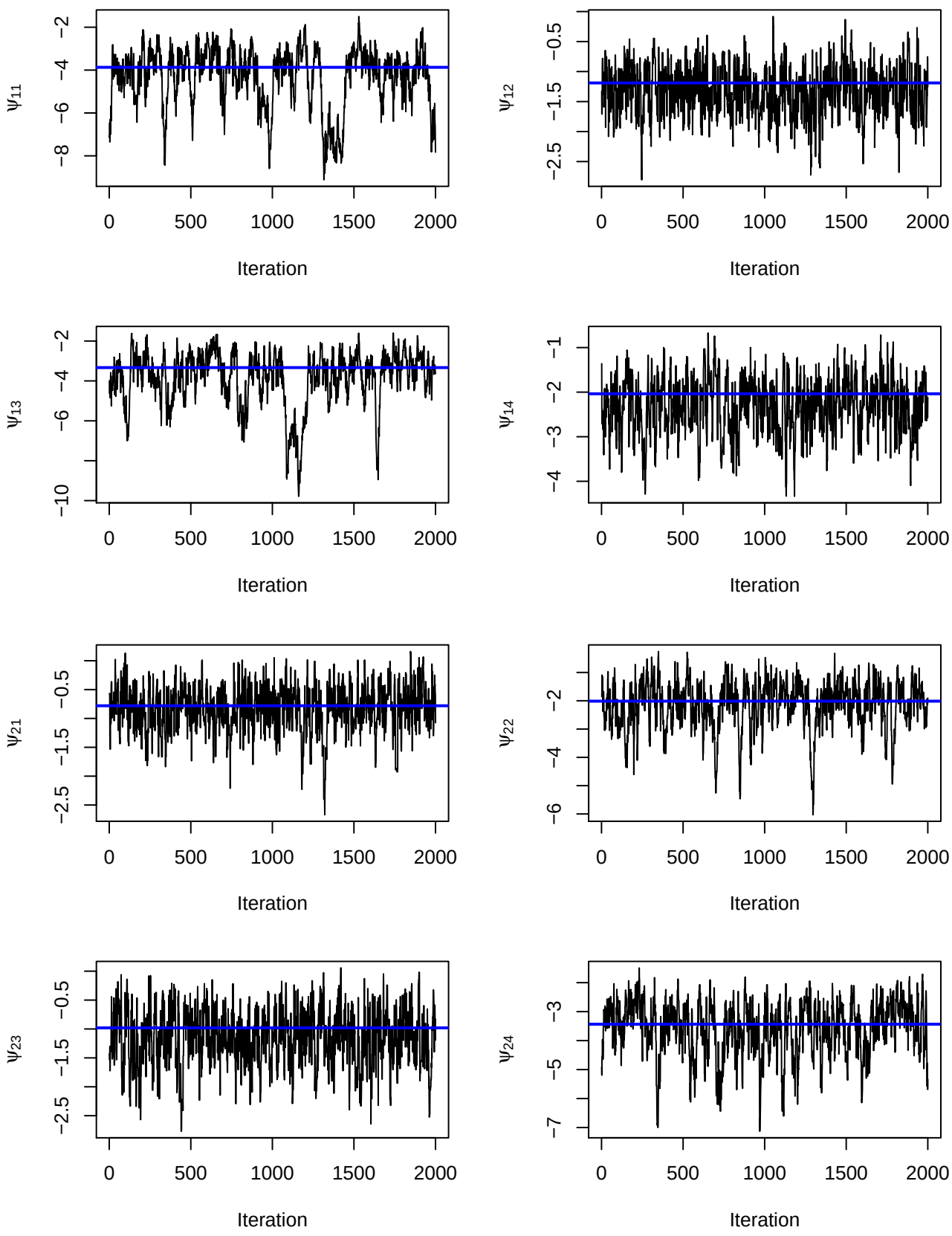
$$\pi_n(\sigma_j^2|-) \equiv InvGamma\left(\frac{16}{2}, \frac{\sum_{i=1}^{6}(\psi_{ij} - \alpha_i - \mu_j)^2 + 5}{2}\right)$$

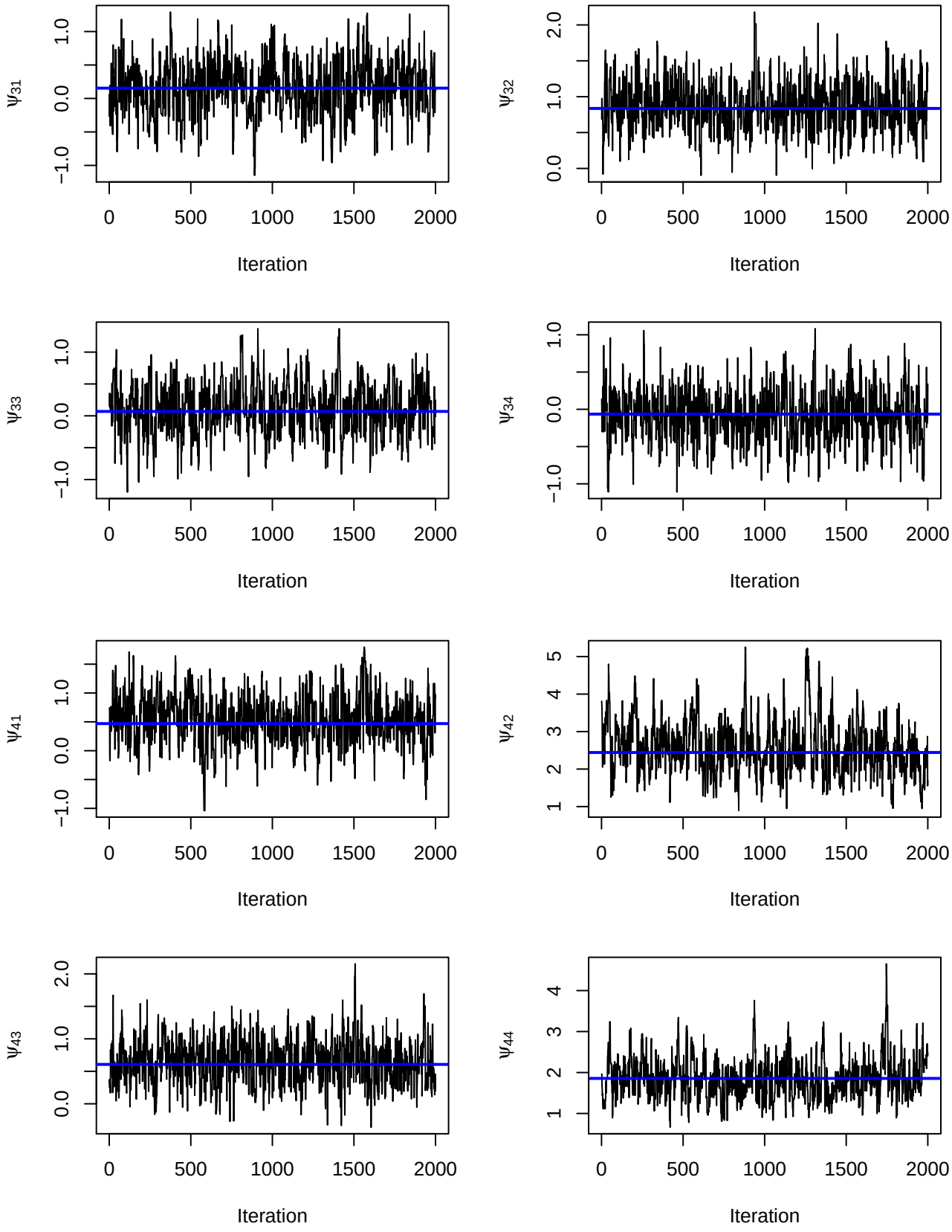
```
set.seed(3234434)

#Data
Irow<-6
Jcol<-4
ymdat<-c(1,49,7,30,1,29,3,26,11,35,2,17,6,22,1,32,14,26,30,43,15,29,15,31,
16,26,23,25,22,34,32,37,29,34,21,25,22,28,16,30,15,30,20,33,13,34,12,39)

ym.mat<-matrix(ymdat,nrow=6,byrow=T)
y.mat<-ym.mat[,c(1,3,5,7)]
m.mat<-ym.mat[,c(1,3,5,7)+1]
z.mat<-m.mat-y.mat

#############################################################################
#MCMC: Starting values
psi.hat<-log(y.mat/z.mat)
old.psi<-new.psi<-log(y.mat/z.mat)
old.al<-apply(old.psi,1,mean)
old.mu<-apply(old.psi,2,mean)
old.sigsq<-1/rgamma(Jcol,10,5)

#MCMC settings
nsamp<-2000  #Number of samples to be collected
nburn<-500   #Number of burn-in iteratons
nthin<-1     #Thinning rate: take a sample after every nthin iterations

nits<-nburn+nsamp*nthin  #Total number of iterations

sq<-0.5  #MH proposal standard deviation
ico<-0

#Matrices to store the samples
psi.samp<-th.samp<-array(0,c(nsamp,Irow,Jcol))
al.samp<-matrix(0,nrow=nsamp,ncol=Irow)
mu.samp<-sig.samp<-matrix(0,nrow=nsamp,ncol=Jcol)

#Set the prior parameters
prior.var<-5^2
prior.a<-5;prior.b<-2.5

#Run the Gibbs sampler
for(iter in 1:nits){
```

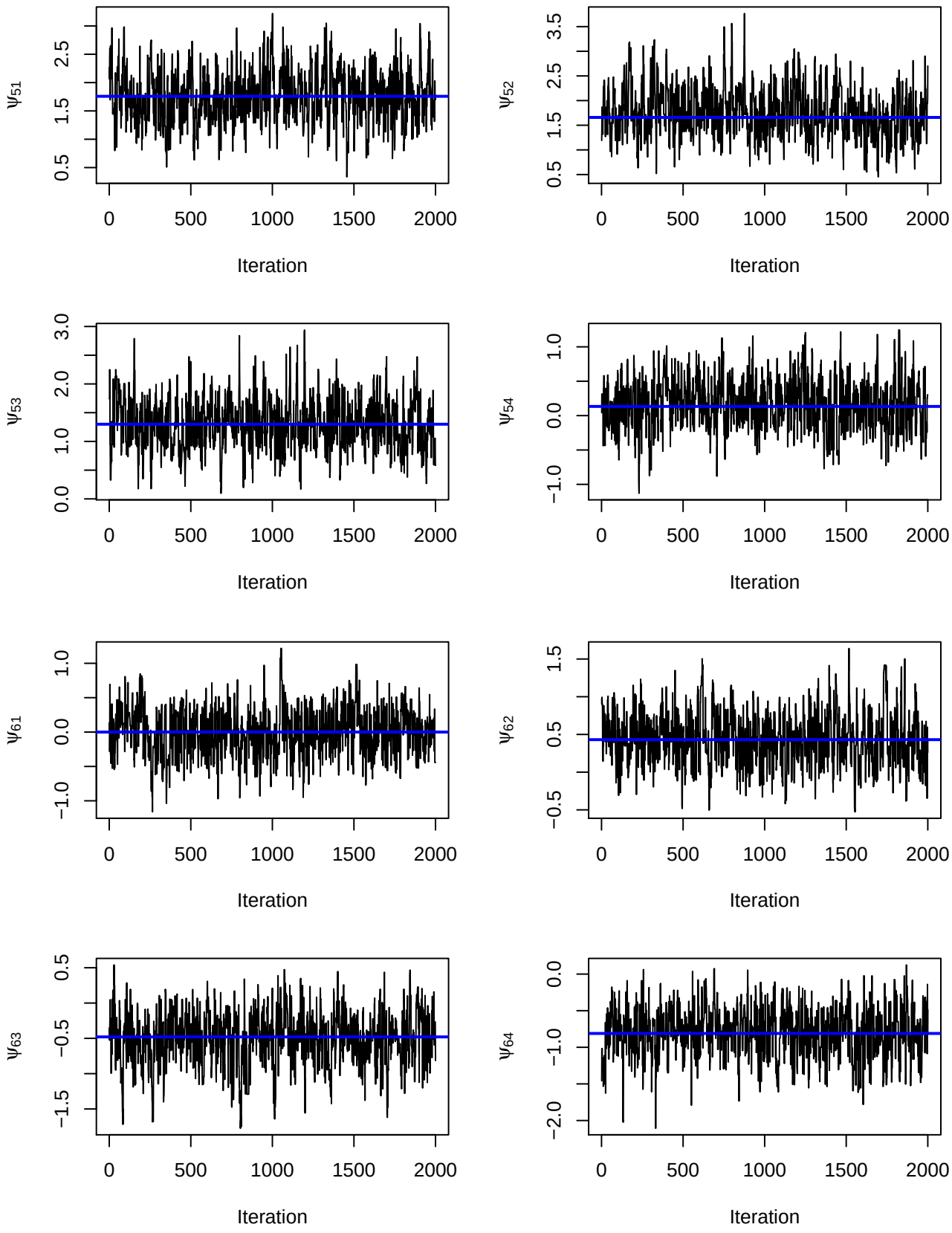
```
    #Metropolis-Hastings for the log-odds parameters
        for(i in 1:Irow){
                for(j in 1:Jcol){
                        old.pij<-1/(1+exp(-old.psi[i,j]))
                        old.like<-y.mat[i,j]*log(old.pij)+z.mat[i,j]*log(1-old.pij)
                        old.prior<-dnorm(old.psi[i,j],old.al[i]+old.mu[j],sqrt(old.sigsq[j]))
                        new.psi[i,j]<-old.psi[i,j]+rnorm(1)*sq
                        new.pij<-1/(1+exp(-new.psi[i,j]))
                        new.like<-y.mat[i,j]*log(new.pij)+z.mat[i,j]*log(1-new.pij)
                        new.prior<-dnorm(new.psi[i,j],old.al[i]+old.mu[j],sqrt(old.sigsq[j]))
                        if(log(runif(1)) < new.like+new.prior-old.like-old.prior){
                                old.psi[i,j]<-new.psi[i,j]
                        }
                }
        }

    #Exact updates for the alphas, mus and sigmas from their full conditionals.
        for(i in 1:Irow){
                old.al[i]<-rnorm(1,
                    sum((old.psi[i,]-old.mu)/old.sigsq)/(sum(1/old.sigsq)+1/prior.var),
                    sqrt(1/(sum(1/old.sigsq)+1/prior.var)))
        }
        for(j in 1:Jcol){
                old.mu[j]<-rnorm(1,
                    sum((old.psi[,j]-old.al)/old.sigsq[j])/(Irow/old.sigsq[j]+1/prior.var),
                    sqrt(1/(Irow/old.sigsq[j]+1/prior.var)))
        }

        mean.mat<-outer(old.al,old.mu,"+")
        for(j in 1:Jcol){
                ssq.j<-sum((old.psi[,j]-mean.mat[,j])^2)
                old.sigsq[j]<-1/rgamma(1,Irow/2+prior.a,ssq.j/2+prior.b)
        }

    #Store the samples
        if(iter > nburn & iter %% nthin ==0){
                ico<-ico+1
                psi.samp[ico,,]<-old.psi
                th.samp[ico,,]<-1/(1+exp(-old.psi))
                al.samp[ico,]<-old.al
                mu.samp[ico,]<-old.mu
                sig.samp[ico,]<-old.sigsq
        }
}
```

Having run the chain for a total of 2500 iterations, we can inspect the output in the form of trace plots and acf plots to assess how well the chain is performing. We do this for the post burn-in samples. In the plots, the blue lines represent the maximum likelihood estimates from a non-hierarchical analysis.

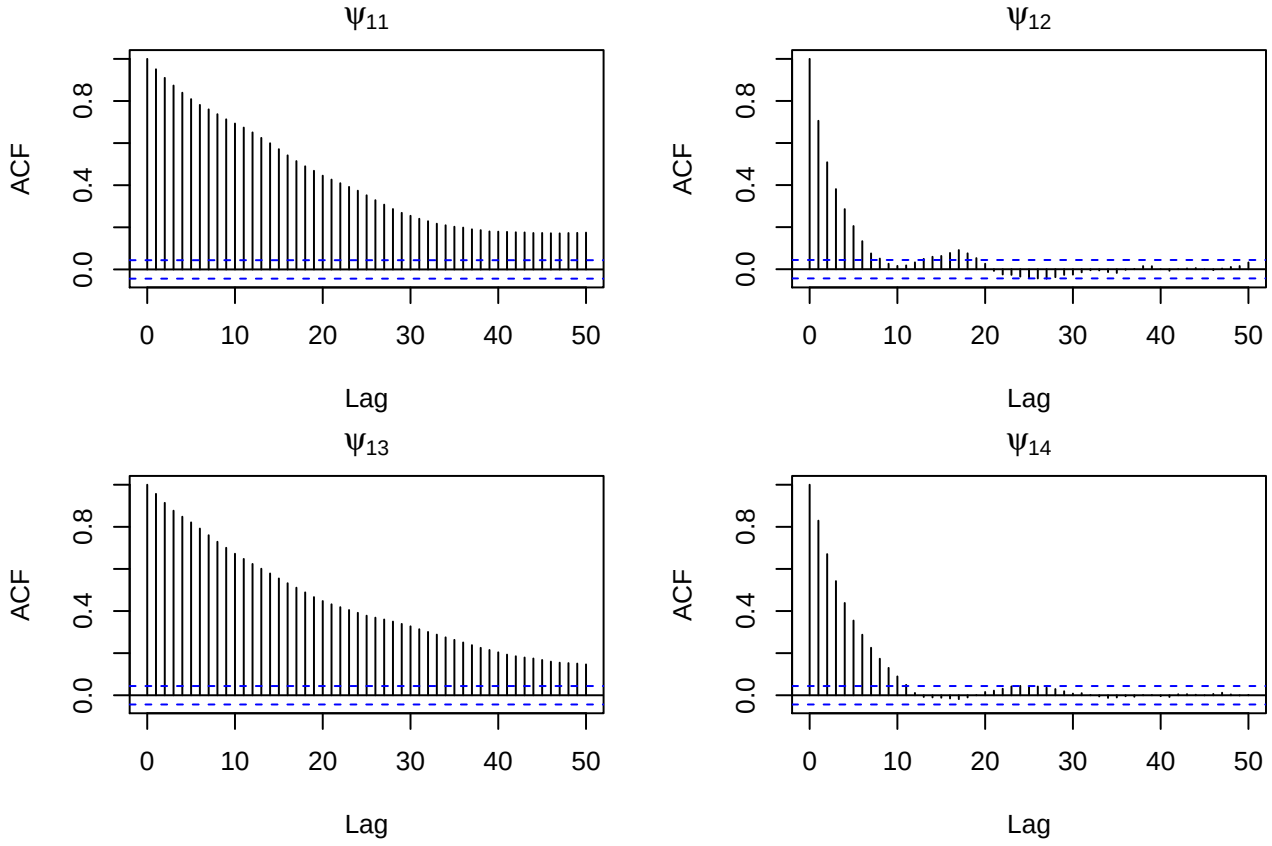
```
par(mfrow=c(2,2),mar=c(4,4,2,2))
for(i in 1:Irow){
        for(j in 1:Jcol){
                plot(psi.samp[,i,j],type="l",ylab=substitute(psi[ival][jval],
            list(ival=i,jval=j)),xlab="Iteration")
                abline(h=psi.hat[i,j],col="blue",lwd=2)
        }
}
```

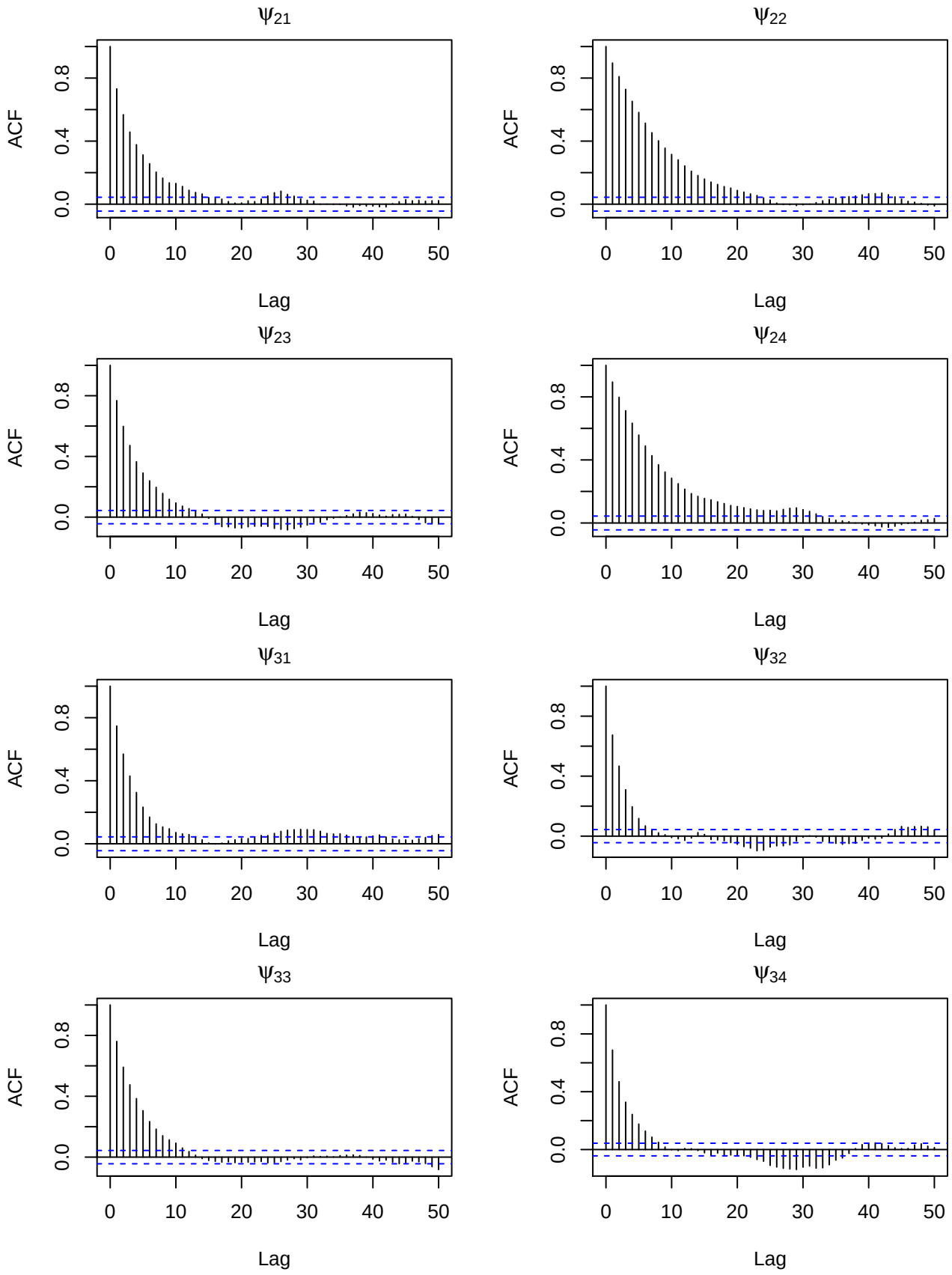

ψ11
−2
−4
−6
−8
0
500
1000
1500
2000
Iteration
ψ12
−0.5
−1.5
−2.5
0
500
1000
1500
2000
Iteration
ψ13
−2
−4
−6
−10
0
500
1000
1500
2000
Iteration
ψ14
−1
−2
−3
−4
0
500
1000
1500
2000
Iteration
ψ21
−0.5
−1.5
−2.5
0
500
1000
1500
2000
Iteration
ψ22
−2
−4
−6
0
500
1000
1500
2000
Iteration
ψ23
−0.5
−1.5
−2.5
0
500
1000
1500
2000
Iteration
ψ24
−3
−5
−7
0
500
1000
1500
2000
Iteration

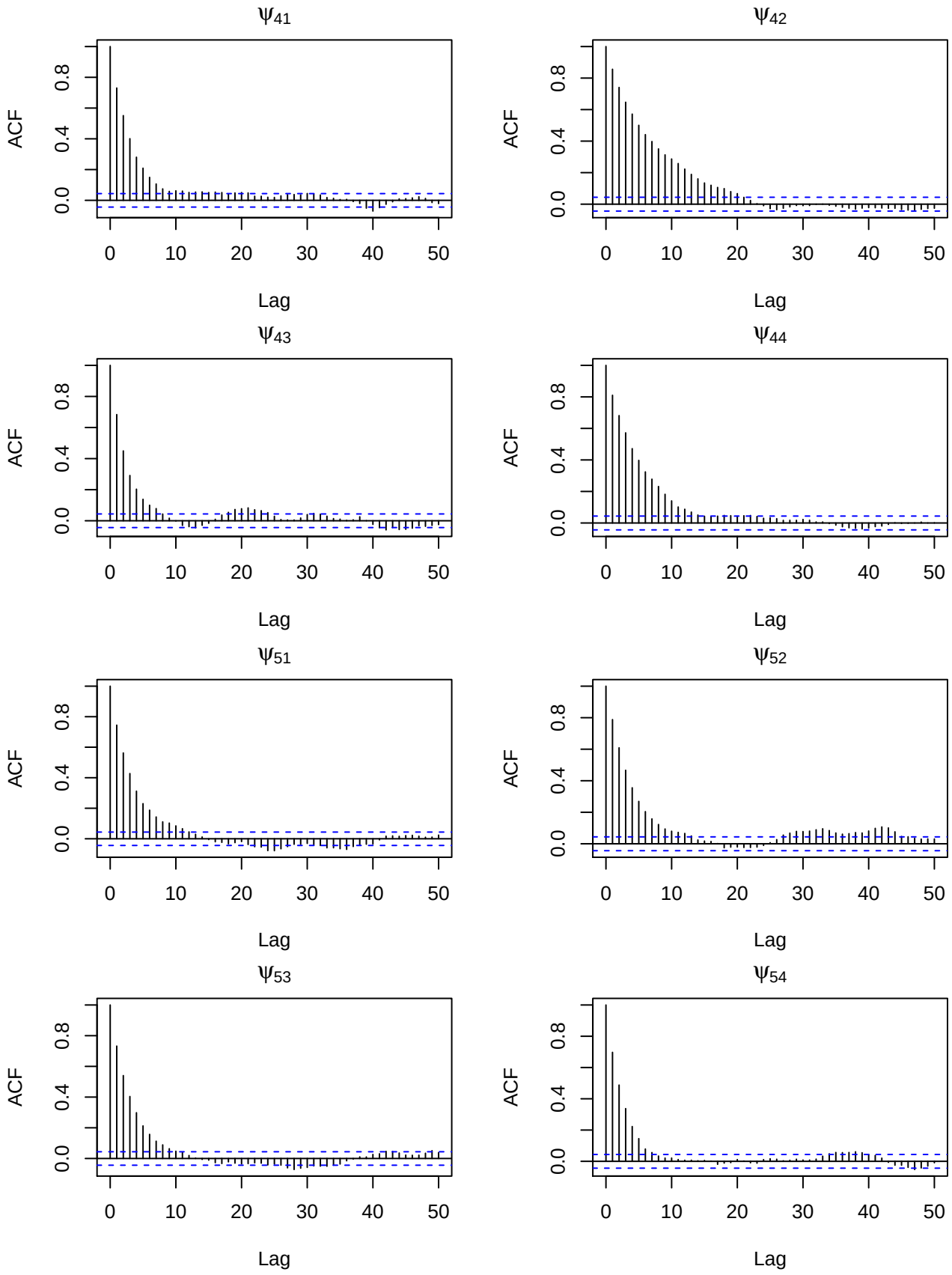

ψ31
ψ32
ψ33
ψ34
ψ41
ψ42
ψ43
ψ44
Iteration
0
500
1000
1500
2000

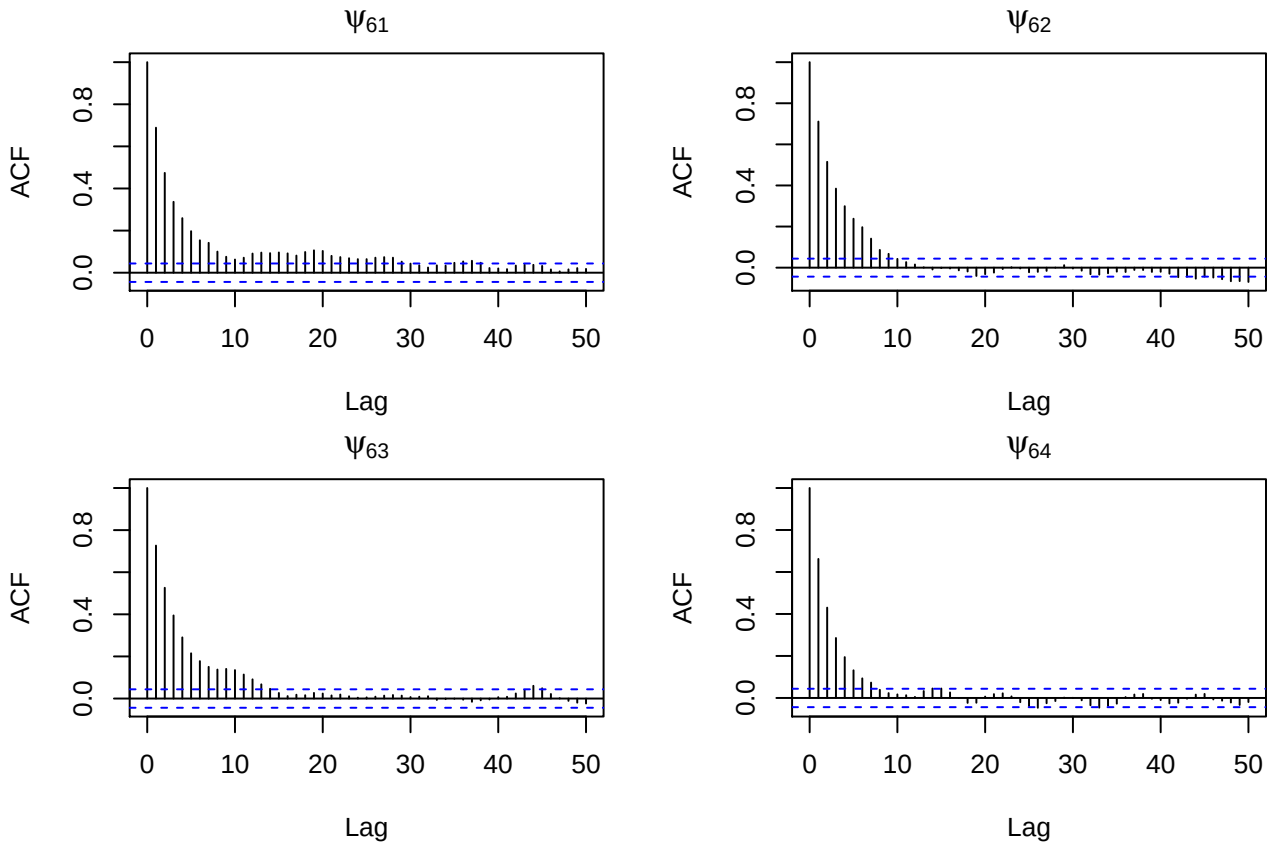

ψ51
ψ52
ψ53
ψ54
ψ61
ψ62
ψ63
ψ64
Iteration
0
500
1000
1500
2000

```
par(mfrow=c(2,2),mar=c(4,4,2,2))
for(i in 1:Irow){
        for(j in 1:Jcol){
                acf(psi.samp[,i,j],main=' ',lag.max=50)
        title(substitute(psi[ival][jval],list(ival=i,jval=j)),line=1)
        }
}
```



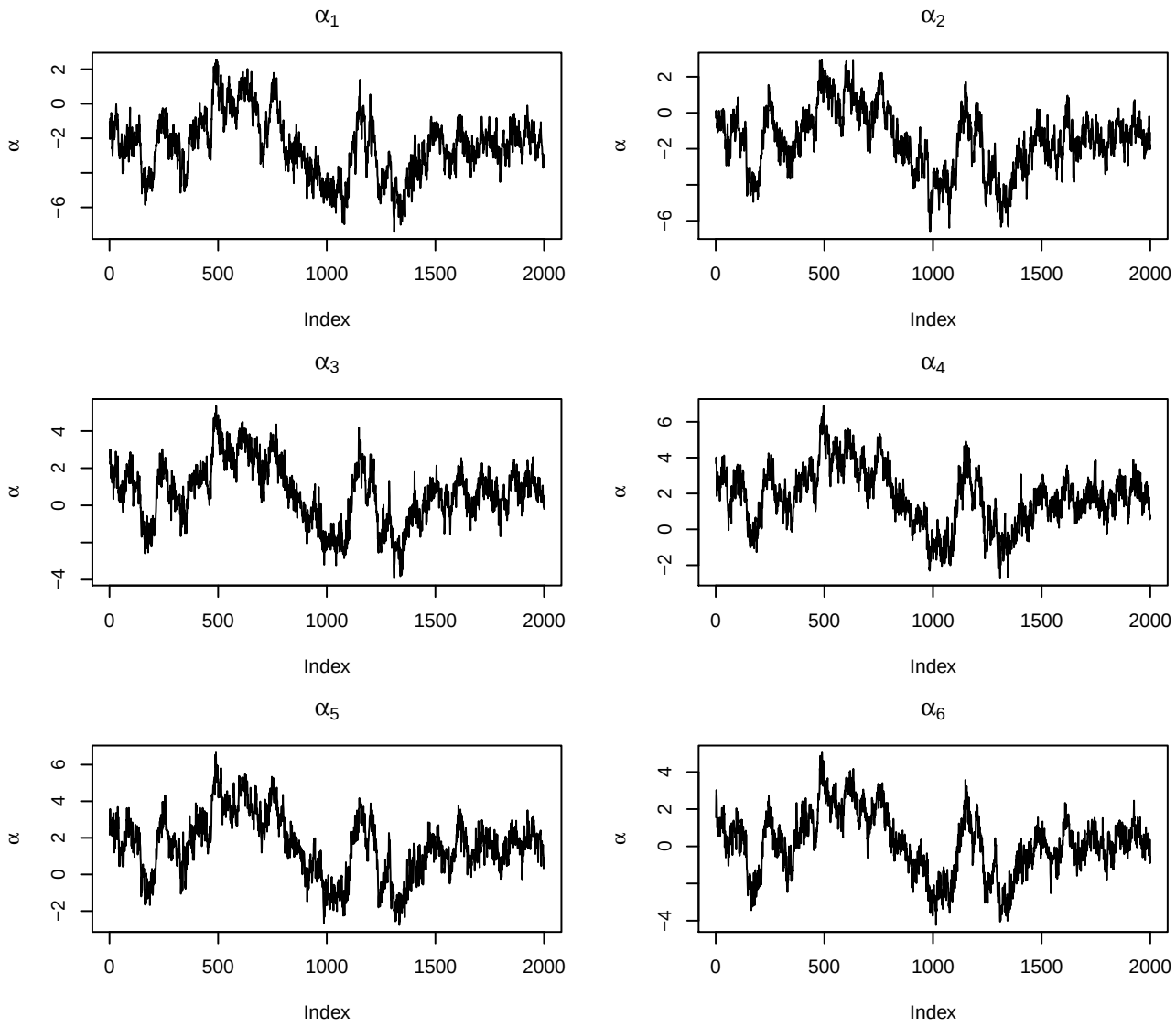

ψ21
ψ22
ψ23
ψ24
ψ31
ψ32
ψ33
ψ34
ACF
Lag
0.0
0.4
0.8
0
10
20
30
40
50


ψ41
ψ42
ψ43
ψ44
ψ51
ψ52
ψ53
ψ54
ACF
Lag
0.0
0.4
0.8
0
10
20
30
40
50

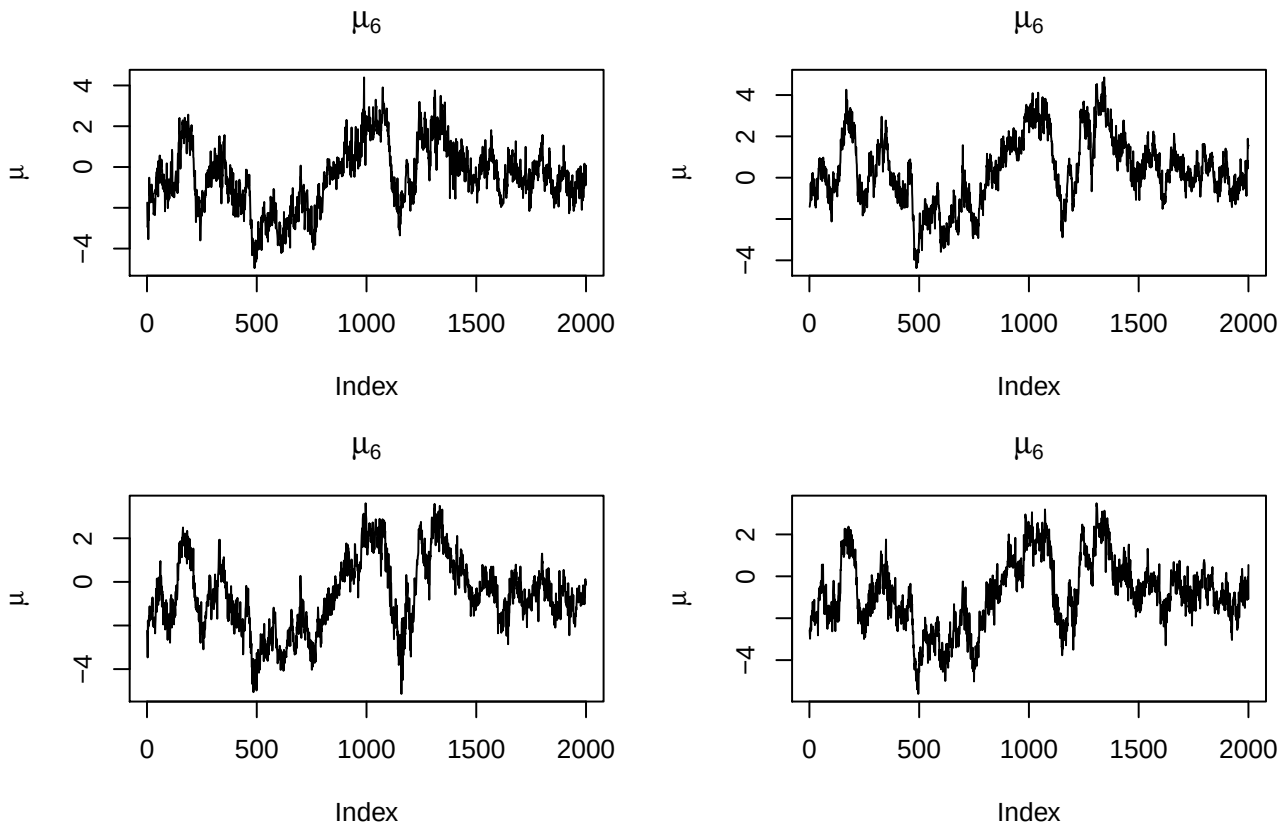


Evidence from these plots for the ψ_{ij} parameters indicates reasonable performance, despite the fact that there is a degree of autocorrelation in the chains. Now we inspect the α_i and μ_j parameter samples.

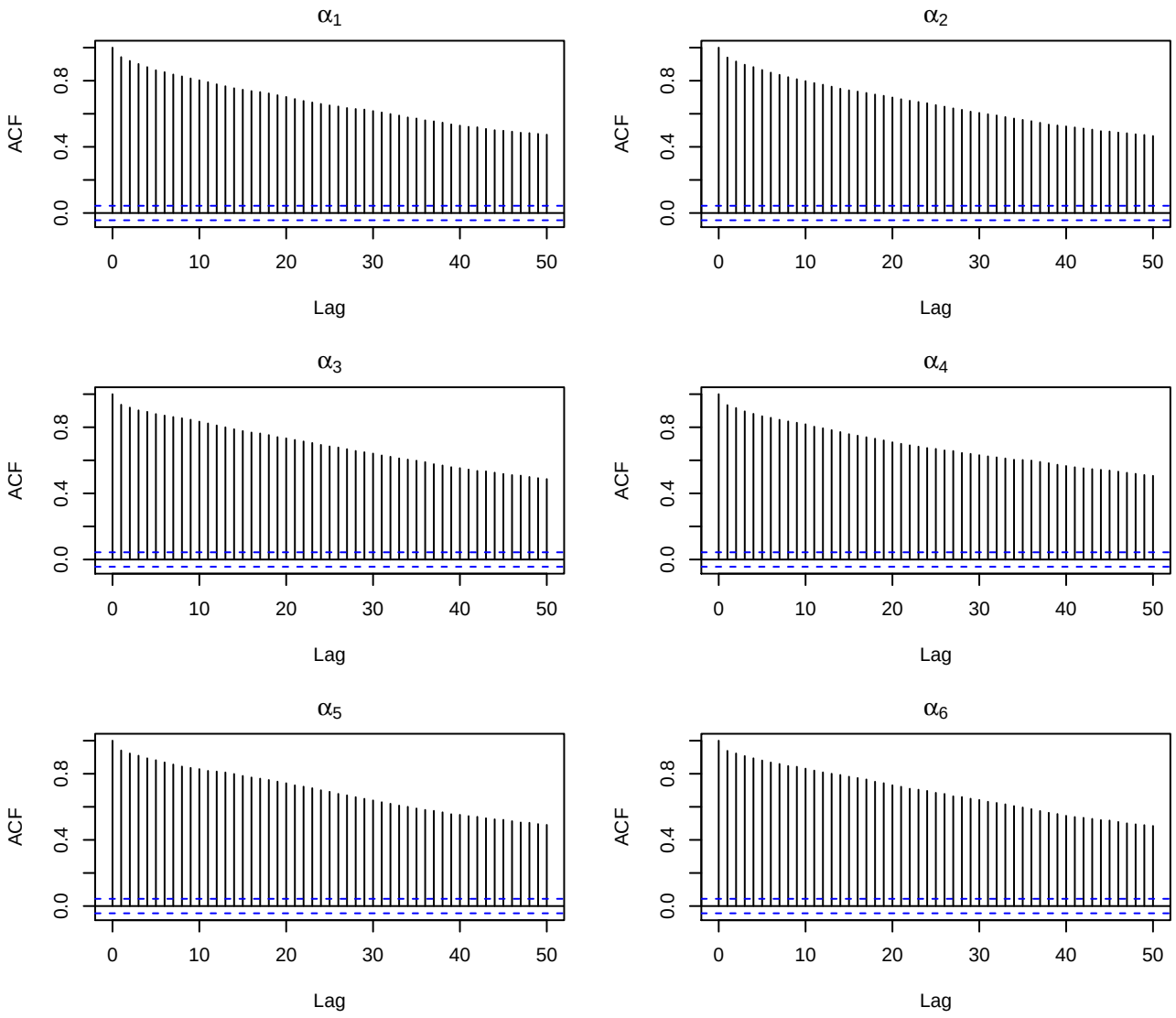
```
par(mfrow=c(3,2),mar=c(4,4,3,2))
for(i in 1:Irow){
        plot(al.samp[,i],type="l",ylab=expression(alpha), main=substitute(alpha[ival],
            list(ival=i)))
}
```



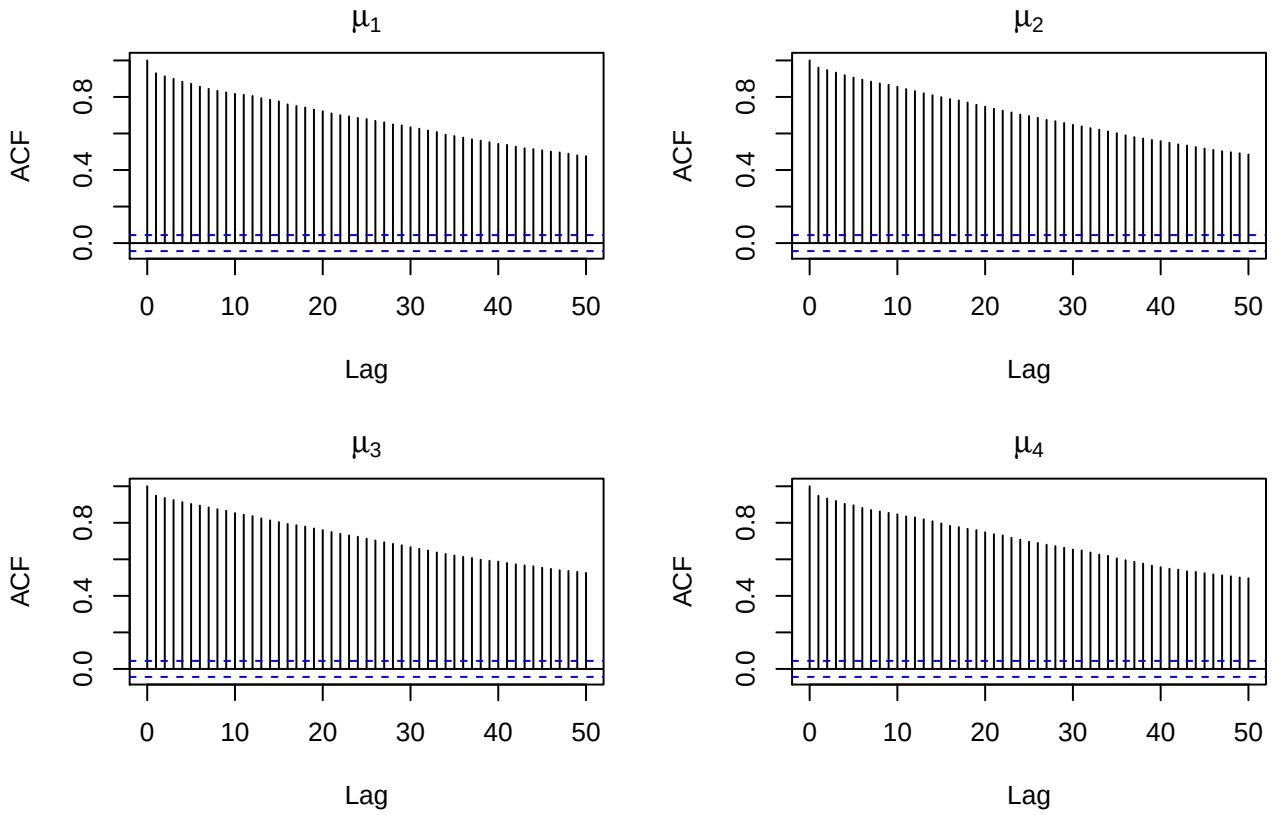

```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        plot(mu.samp[,j],type="l",ylab=expression(mu),main=substitute(mu[jval],
            list(jval=i)))
}
```




```
par(mfrow=c(3,2),mar=c(4,4,3,2))
for(i in 1:Irow){
        acf(al.samp[,i],main=' ',lag.max=50)
    title(substitute(alpha[ival], list(ival=i)),line=1)
}
```




```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        acf(mu.samp[,j],main=' ',lag.max=50)
    title(substitute(mu[jval], list(jval=j)),line=1)
}
```

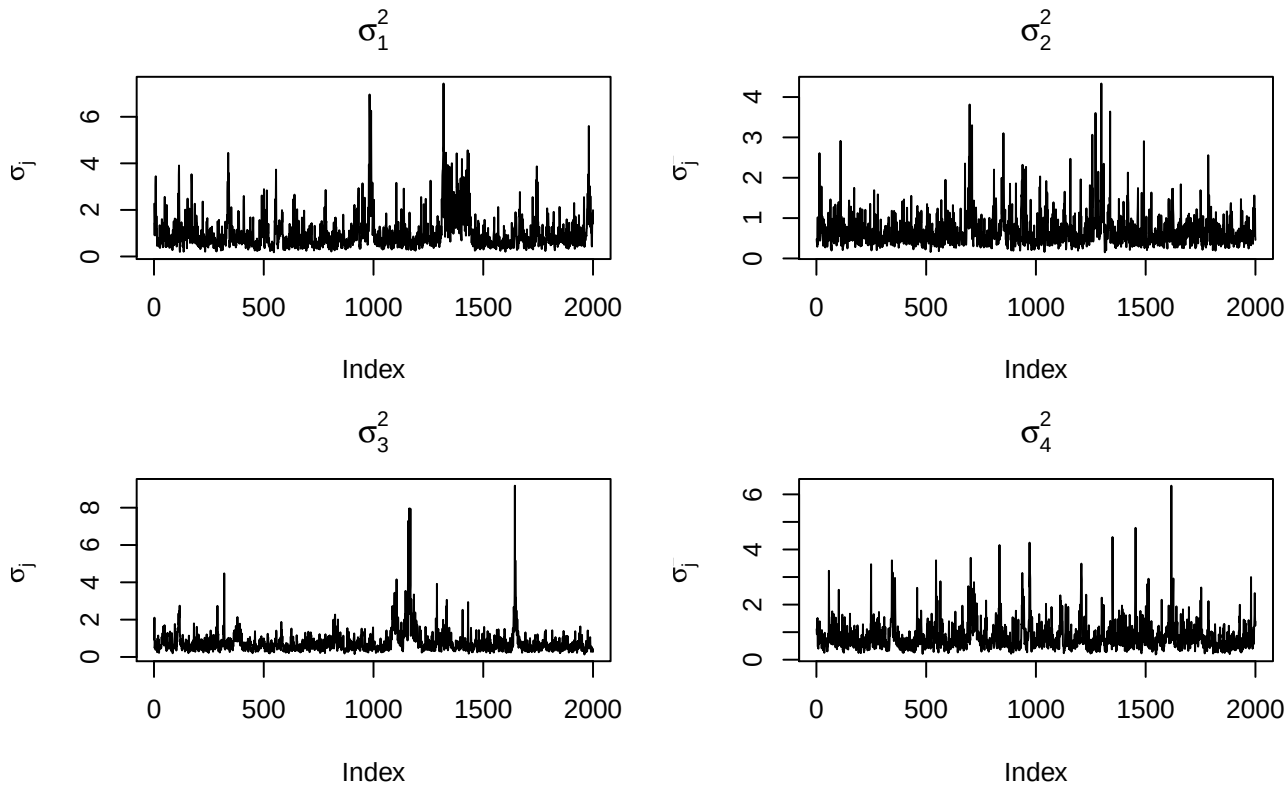


These plots exhibit a very high degree of dependence within the chains. If we compute the covariances between these samples, we see that they are very large in magnitude.

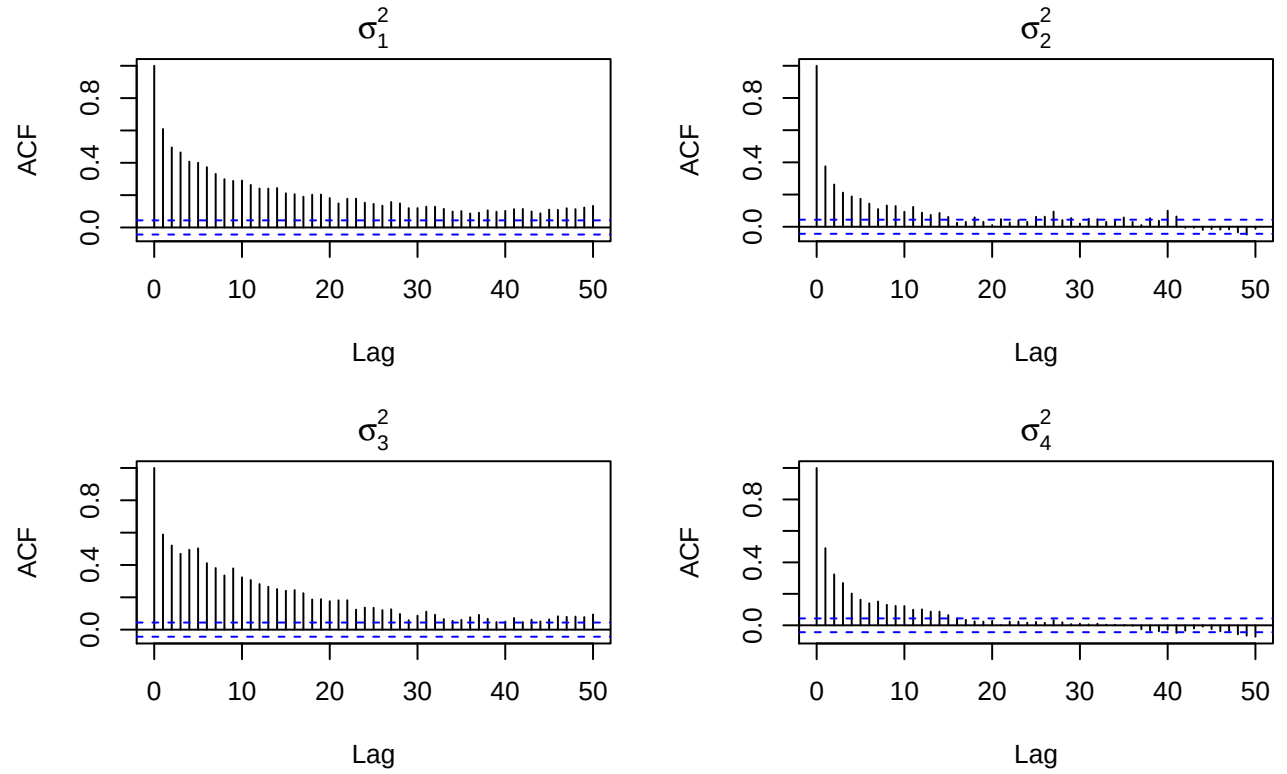
```
comb.samp<-cbind(al.samp,mu.samp)
colnames(comb.samp)<-c("al1","al2","al3","al4","al5","al6","mu1","mu2","mu3","mu4")
round(cor(comb.samp),3)

+        al1    al2    al3    al4    al5    al6    mu1    mu2    mu3    mu4
+ al1  1.000  0.864  0.903  0.900  0.893  0.903 -0.898 -0.927 -0.898 -0.913
+ al2  0.864  1.000  0.898  0.869  0.902  0.902 -0.902 -0.910 -0.904 -0.907
+ al3  0.903  0.898  1.000  0.911  0.923  0.925 -0.923 -0.940 -0.928 -0.933
+ al4  0.900  0.869  0.911  1.000  0.901  0.910 -0.912 -0.921 -0.922 -0.921
+ al5  0.893  0.902  0.923  0.901  1.000  0.924 -0.917 -0.936 -0.929 -0.934
+ al6  0.903  0.902  0.925  0.910  0.924  1.000 -0.920 -0.943 -0.932 -0.938
+ mu1 -0.898 -0.902 -0.923 -0.912 -0.917 -0.920  1.000  0.921  0.909  0.918
+ mu2 -0.927 -0.910 -0.940 -0.921 -0.936 -0.943  0.921  1.000  0.932  0.938
+ mu3 -0.898 -0.904 -0.928 -0.922 -0.929 -0.932  0.909  0.932  1.000  0.929
+ mu4 -0.913 -0.907 -0.933 -0.921 -0.934 -0.938  0.918  0.938  0.929  1.000
```

We finally inspect the σ_j^2 parameters.

```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        plot(sig.samp[,j],type="l",ylab=expression(sigma[j]^2),
            main=substitute(sigma[jval]^2, list(jval=j)))
}
```




```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        acf(sig.samp[,j],main='',lag.max=50);title(substitute(sigma[jv]^2,list(jv=j)),line=1)
}
```



The sampling for these parameters also exhibits a relatively high degree of auto correlation. Finally, we can check the effective sample sizes

```
library(coda)
colnames(al.samp)<-c("al1","al2","al3","al4","al5","al6")
colnames(mu.samp)<-c("mu1","mu2","mu3","mu4")
colnames(sig.samp)<-c("sig1","sig2","sig3","sig4")
effectiveSize(al.samp);effectiveSize(mu.samp);effectiveSize(sig.samp)

+      al1      al2      al3      al4      al5      al6
+ 19.62556 22.18742 12.56394 14.66654 12.06203 12.63256
+      mu1      mu2      mu3      mu4
+ 14.29796 13.03691 12.31649 11.93431
+      sig1      sig2      sig3      sig4
+ 150.09566 307.25451  94.95639 342.91374
```

These results indicate a very high degree of autocorrelation in sampling of the parameters at the third stage of the hierarchical model.

Algorithm 2: Blocked Gibbs sampler.
In a blocked Gibbs sampler, we place several of the parameters together in a "block" and attempt to update them together. In this model, we choose to form a block by defining the 10-dimensional vector

$$\varphi = (\alpha_1, \alpha_2, \ldots, \alpha_6, \mu_1, \ldots, \mu_4)^\top,$$

and exploit the fact that the joint full conditional distribution for this block of parameters, given the other parameters, is multivariate Normal. We have that if Ψ is the vector defined by

$$\Psi = (\psi_{11}, \psi_{12}, \psi_{13}, \psi_{14}, \psi_{21}, \ldots, \psi_{64})^\top$$

then

$$\pi_n(\varphi|-) \propto \exp\left\{-\frac{1}{2}(\Psi - \mathbf{A}\varphi)^\top \mathbf{D}^{-1}(\Psi - \mathbf{A}\varphi)\right\} \exp\left\{-\frac{1}{2}\varphi^\top \mathbf{V}^{-1}\varphi\right\}$$

where $\mathbf{D}$ is the 24×24 diagonal matrix with diagonal

$$(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_1^2, \ldots, \sigma_4^2)$$

and $\mathbf{V}$ is the 10×10 diagonal matrix with 5^2 everywhere on the diagonal. The matrix $\mathbf{A}$ is the design matrix for the Second Stage model: it is a 24×10 matrix of zeros and ones. By a standard calculation, we have that

$$\pi_n(\varphi|-) \equiv Normal_{10}(\mathbf{m}, \mathbf{M})$$

where

$$\mathbf{m} = \left(\mathbf{A}^\top \mathbf{D}^{-1}\mathbf{A} + \mathbf{V}^{-1}\right)^{-1} \mathbf{A}^\top \mathbf{D}^{-1}\Psi \qquad \mathbf{M} = \left(\mathbf{A}^\top \mathbf{D}^{-1}\mathbf{A} + \mathbf{V}^{-1}\right)^{-1}$$

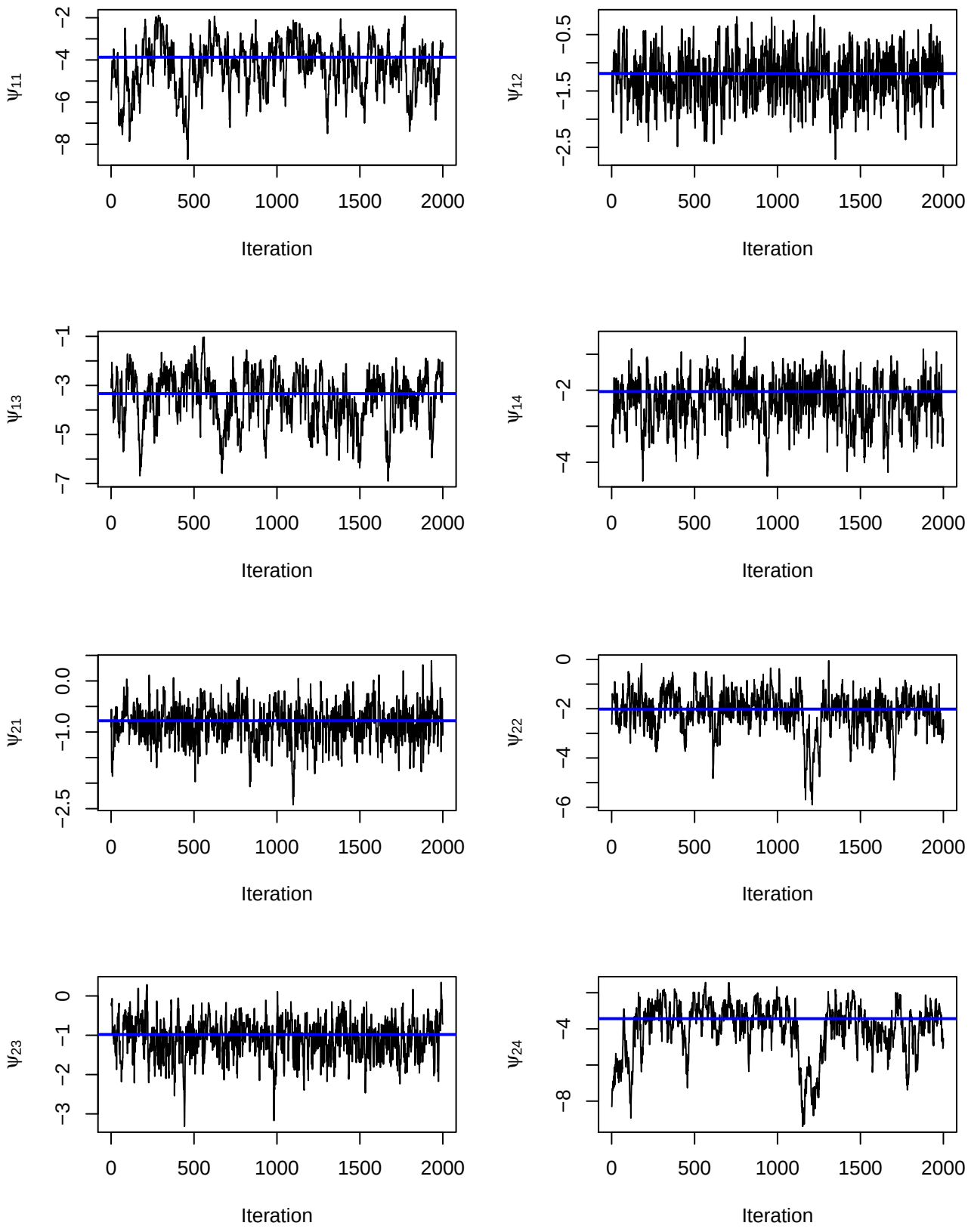
```
set.seed(3234434)
Irow<-6
Jcol<-4

ymdat<-c(1,49,7,30,1,29,3,26,11,35,2,17,6,22,1,32,14,26,30,43,15,29,15,31,
16,26,23,25,22,34,32,37,29,34,21,25,22,28,16,30,15,30,20,33,13,34,12,39)
ym.mat<-matrix(ymdat,nrow=6,byrow=T)

y.mat<-ym.mat[,c(1,3,5,7)]
m.mat<-ym.mat[,c(1,3,5,7)+1]
z.mat<-m.mat-y.mat

#The design matrix
A<-matrix(0,nrow=24,ncol=10)
A[1:4,1]<-A[5:8,2]<-A[9:12,3]<-A[13:16,4]<-A[17:20,5]<-A[21:24,6]<-1
A[(0:5)*4+1,7]<-A[(0:5)*4+2,8]<-A[(0:5)*4+3,9]<-A[(0:5)*4+4,10]<-1
A
```

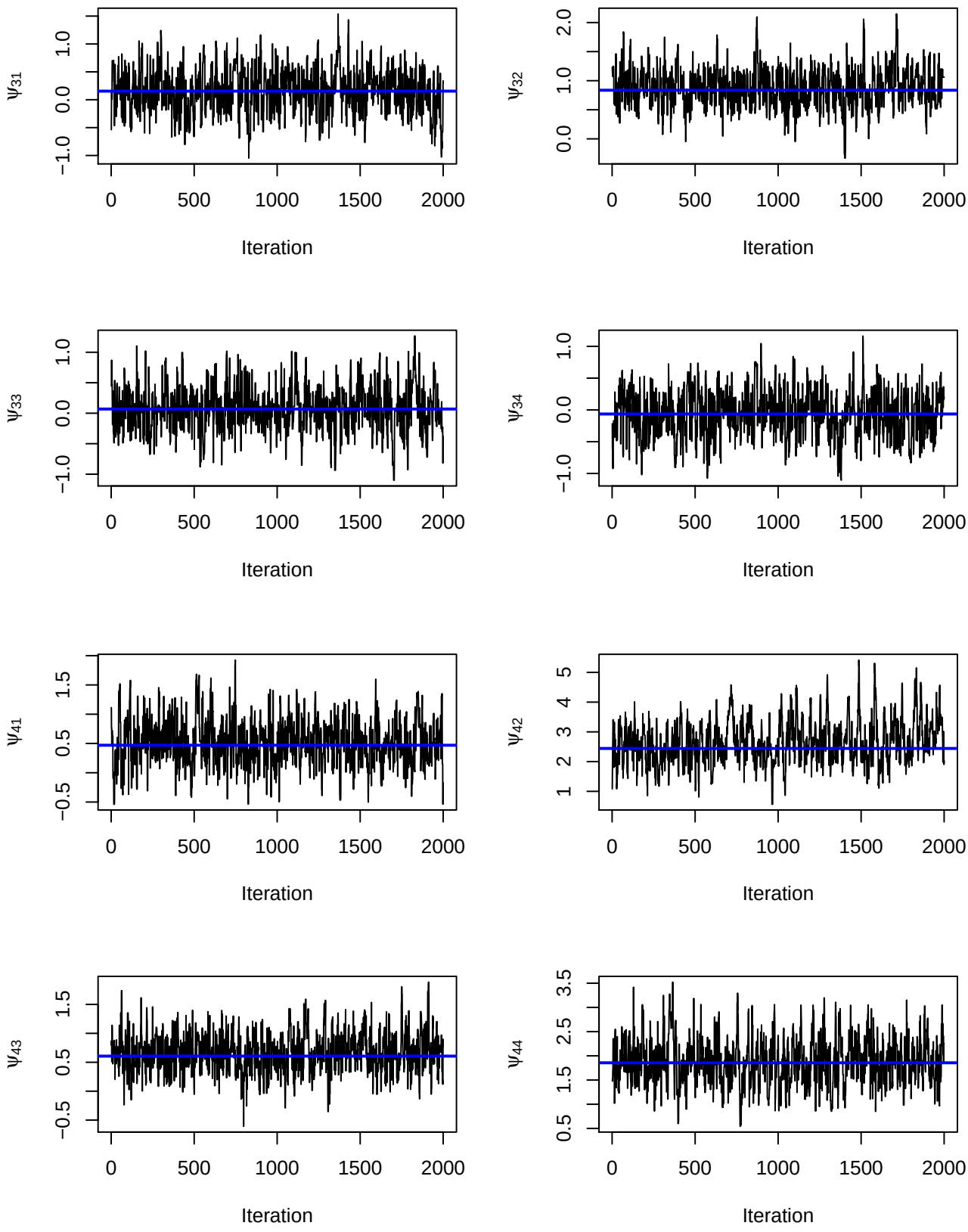
```
+       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
+  [1,]    1    0    0    0    0    0    1    0    0     0
+  [2,]    1    0    0    0    0    0    0    1    0     0
+  [3,]    1    0    0    0    0    0    0    0    1     0
+  [4,]    1    0    0    0    0    0    0    0    0     1
+  [5,]    0    1    0    0    0    0    1    0    0     0
+  [6,]    0    1    0    0    0    0    0    1    0     0
+  [7,]    0    1    0    0    0    0    0    0    1     0
+  [8,]    0    1    0    0    0    0    0    0    0     1
+  [9,]    0    0    1    0    0    0    1    0    0     0
+ [10,]    0    0    1    0    0    0    0    1    0     0
+ [11,]    0    0    1    0    0    0    0    0    1     0
+ [12,]    0    0    1    0    0    0    0    0    0     1
+ [13,]    0    0    0    1    0    0    1    0    0     0
+ [14,]    0    0    0    1    0    0    0    1    0     0
+ [15,]    0    0    0    1    0    0    0    0    1     0
+ [16,]    0    0    0    1    0    0    0    0    0     1
+ [17,]    0    0    0    0    1    0    1    0    0     0
+ [18,]    0    0    0    0    1    0    0    1    0     0
+ [19,]    0    0    0    0    1    0    0    0    1     0
+ [20,]    0    0    0    0    1    0    0    0    0     1
+ [21,]    0    0    0    0    0    1    1    0    0     0
+ [22,]    0    0    0    0    0    1    0    1    0     0
+ [23,]    0    0    0    0    0    1    0    0    1     0
+ [24,]    0    0    0    0    0    1    0    0    0     1
```

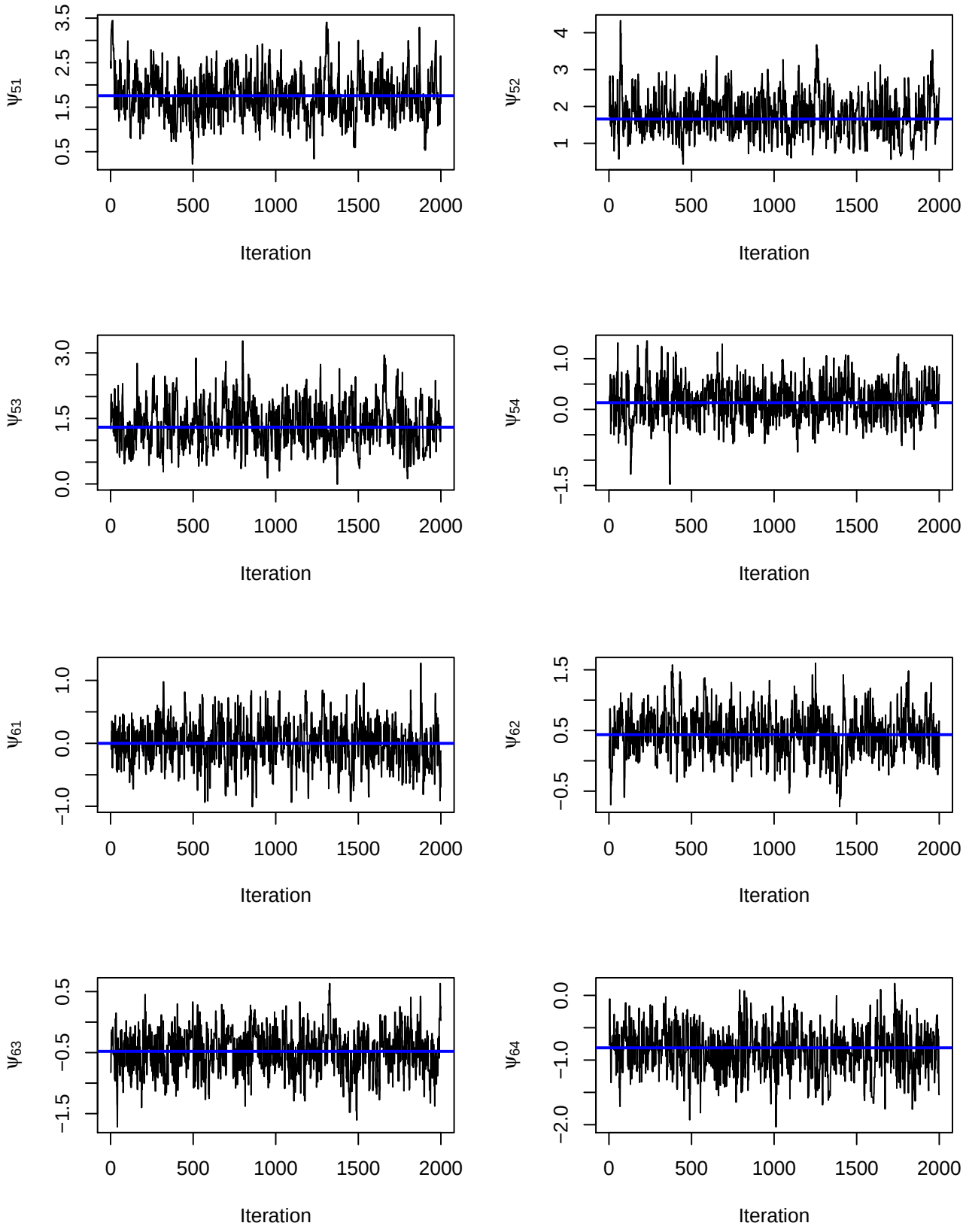
```
#########################################################################
#Starting values
library(mvnfast)
old.psi<-new.psi<-log(y.mat/z.mat)

psi.hat<-log(y.mat/z.mat)
old.al<-apply(old.psi,1,mean)
old.mu<-apply(old.psi,2,mean)
old.sigsq<-1/rgamma(Jcol,10,5)

nsamp<-2000
nburn<-500
nthin<-1

nits<-nburn+nsamp*nthin
sq<-0.5
ico<-0

psi.samp.new<-th.samp.new<-array(0,c(nsamp,Irow,Jcol))
al.samp.new<-matrix(0,nrow=nsamp,ncol=Irow)
mu.samp.new<-sig.samp.new<-matrix(0,nrow=nsamp,ncol=Jcol)

prior.var<-5^2
prior.a<-5;prior.b<-2.5

for(iter in 1:nits){

        for(i in 1:Irow){
                for(j in 1:Jcol){
                old.pij<-1/(1+exp(-old.psi[i,j]))
                        old.like<-y.mat[i,j]*log(old.pij)+z.mat[i,j]*log(1-old.pij)
                        old.prior<-dnorm(old.psi[i,j],old.al[i]+old.mu[j],sqrt(old.sigsq[j]))
                        new.psi[i,j]<-old.psi[i,j]+rnorm(1)*sq
                        new.pij<-1/(1+exp(-new.psi[i,j]))
                        new.like<-y.mat[i,j]*log(new.pij)+z.mat[i,j]*log(1-new.pij)
                        new.prior<-dnorm(new.psi[i,j],old.al[i]+old.mu[j],sqrt(old.sigsq[j]))
```

```
                        if(log(runif(1)) < new.like+new.prior-old.like-old.prior){
                                old.psi[i,j]<-new.psi[i,j]
                        }
                }
        }

        #Do the joint update
        psivec<-as.vector(t(old.psi))
        Dinv<-diag(1/old.sigsq[rep(1:4,6)])
        Vinv<-diag(1/prior.var,Irow+Jcol)
        Mval<-solve(t(A) %*% (Dinv %*% A) + Vinv)
        mval<-Mval %*% (t(A) %*% (Dinv %*% psivec))

        vtheta<-rmvn(1,mu=mval,sigma=Mval)

        old.al<-vtheta[1:6]
        old.mu<-vtheta[7:10]

        mean.mat<-outer(old.al,old.mu,"+")
        for(j in 1:Jcol){
                ssq.j<-sum((old.psi[,j]-mean.mat[,j])^2)
                old.sigsq[j]<-1/rgamma(1,Irow/2+prior.a,ssq.j/2+prior.b)
        }
        if(iter > nburn & iter %% nthin ==0){
                ico<-ico+1
                psi.samp.new[ico,,]<-old.psi
                th.samp.new[ico,,]<-1/(1+exp(-old.psi))
                al.samp.new[ico,]<-old.al
                mu.samp.new[ico,]<-old.mu
                sig.samp.new[ico,]<-old.sigsq
        }
}
```

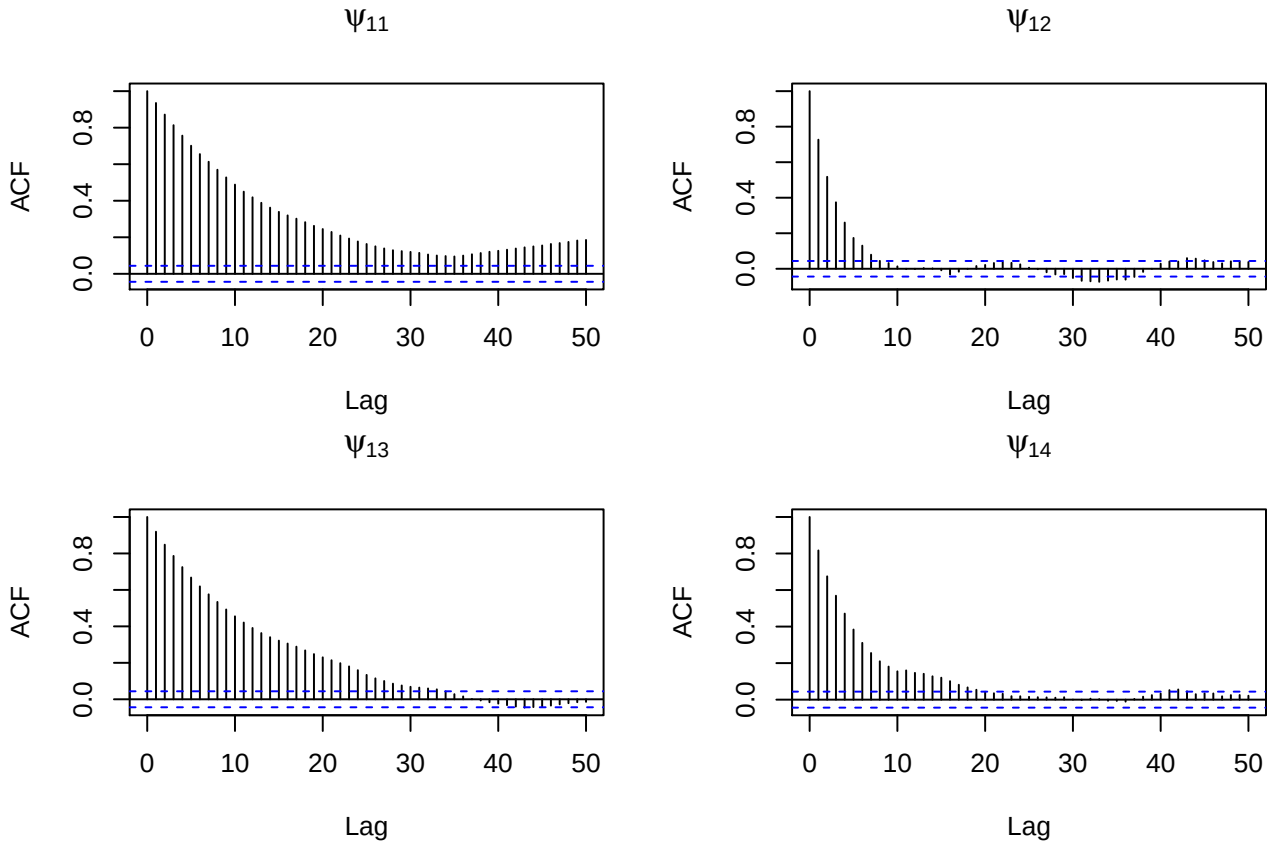
```
psi.means.new<-apply(psi.samp.new,2:3,mean)
psi.medians.new<-apply(psi.samp.new,2:3,median)
psi.lower.new<-apply(psi.samp.new,2:3,quantile,prob=0.025)
psi.upper.new<-apply(psi.samp.new,2:3,quantile,prob=0.975)

expit<-function(x){return(1/(1+exp(-x)))}
th.medians.new<-expit(psi.medians.new)
th.lower.new<-expit(psi.lower.new)
th.upper.new<-expit(psi.upper.new)

par(mfrow=c(2,2),mar=c(4,4,3,2))
for(i in 1:Irow){
        for(j in 1:Jcol){
                plot(psi.samp.new[,i,j],type="l",ylab=substitute(psi[ival][jval],
                list(ival=i,jval=j)),xlab="Iteration")
                abline(h=psi.hat[i,j],col="blue",lwd=2)
        }
}
```

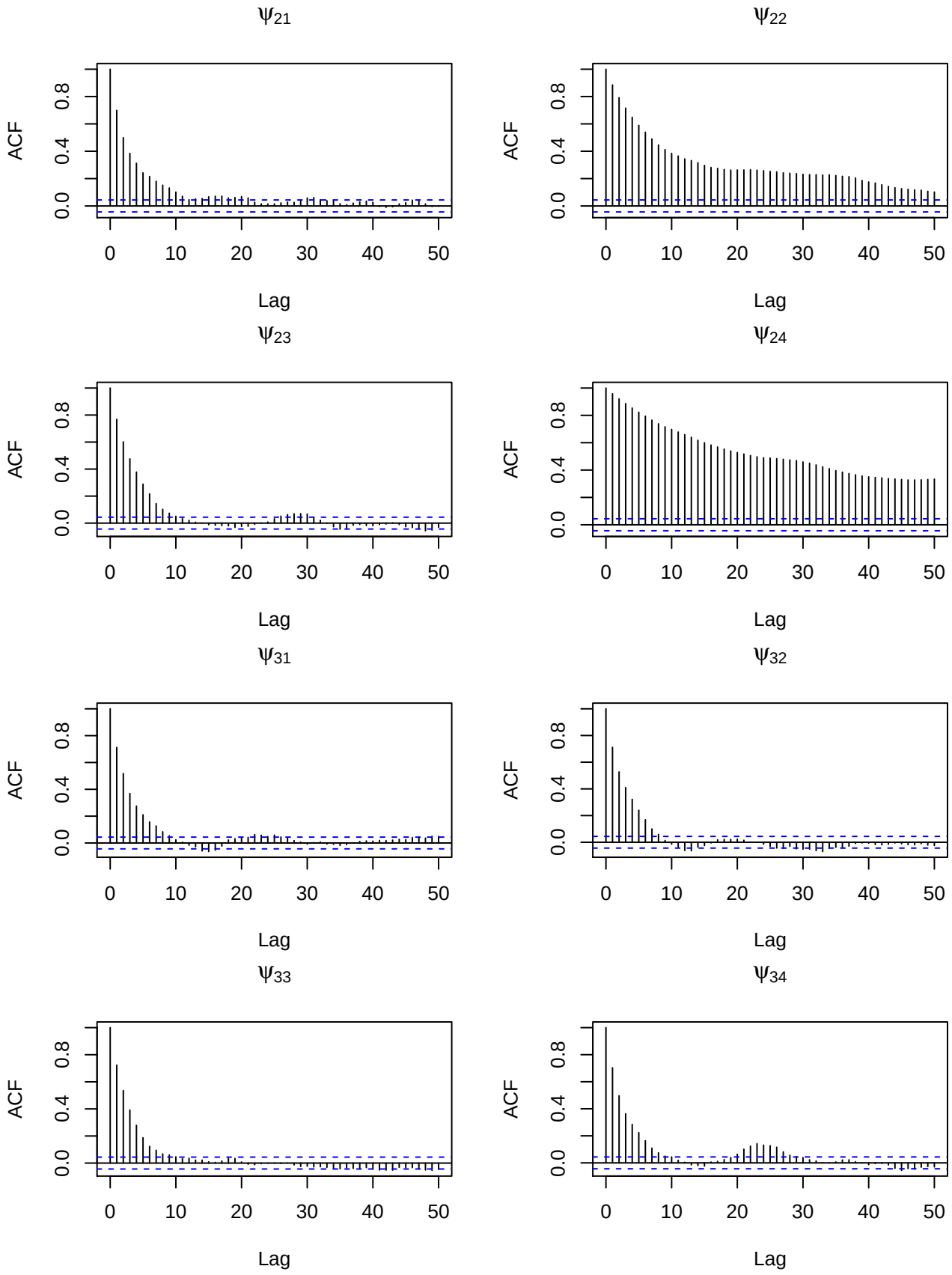

ψ11
ψ12
ψ13
ψ14
ψ21
ψ22
ψ23
ψ24
Iteration
0
500
1000
1500
2000

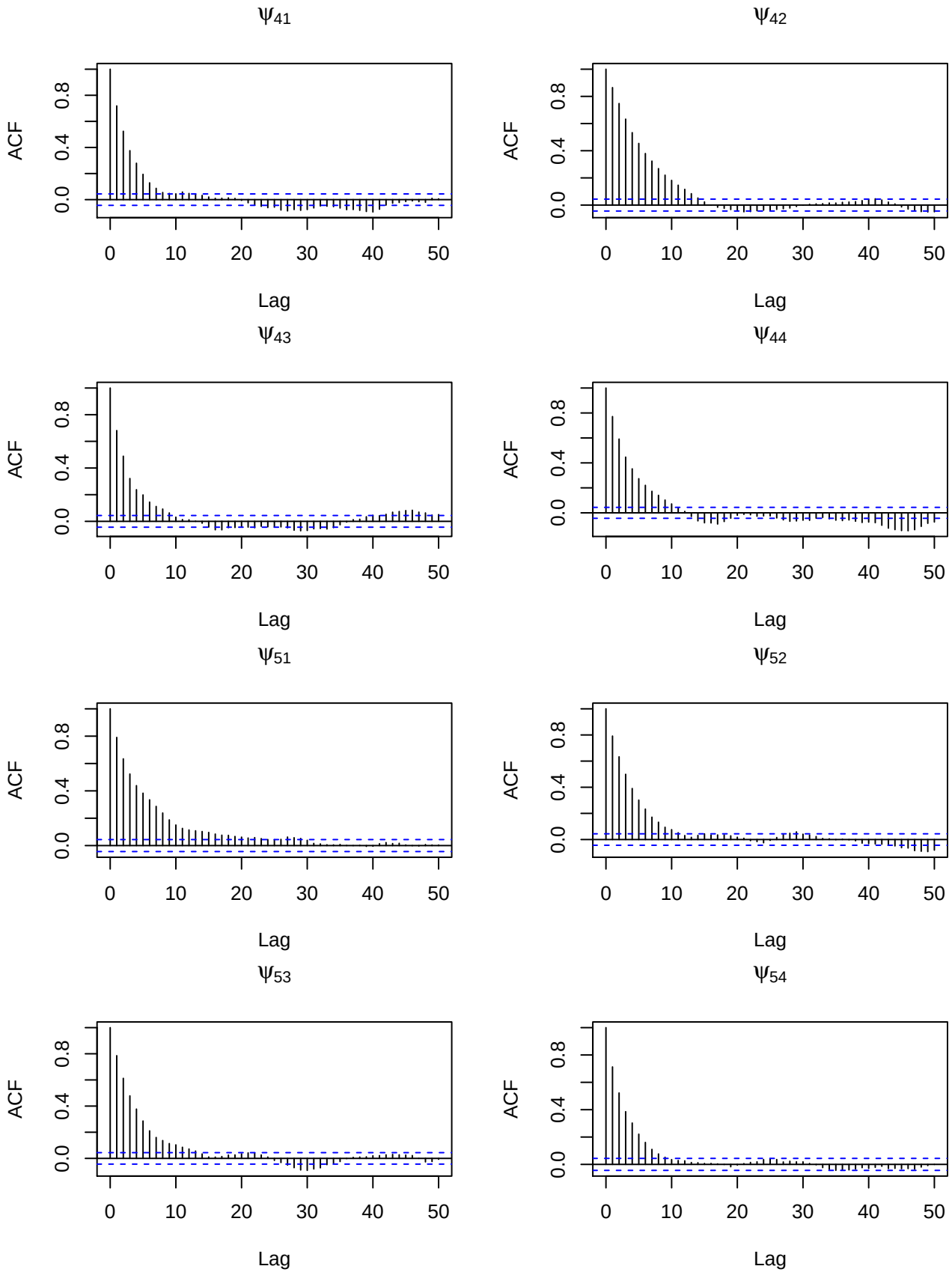

ψ31
ψ32
ψ33
ψ34
ψ41
ψ42
ψ43
ψ44
Iteration
0
500
1000
1500
2000

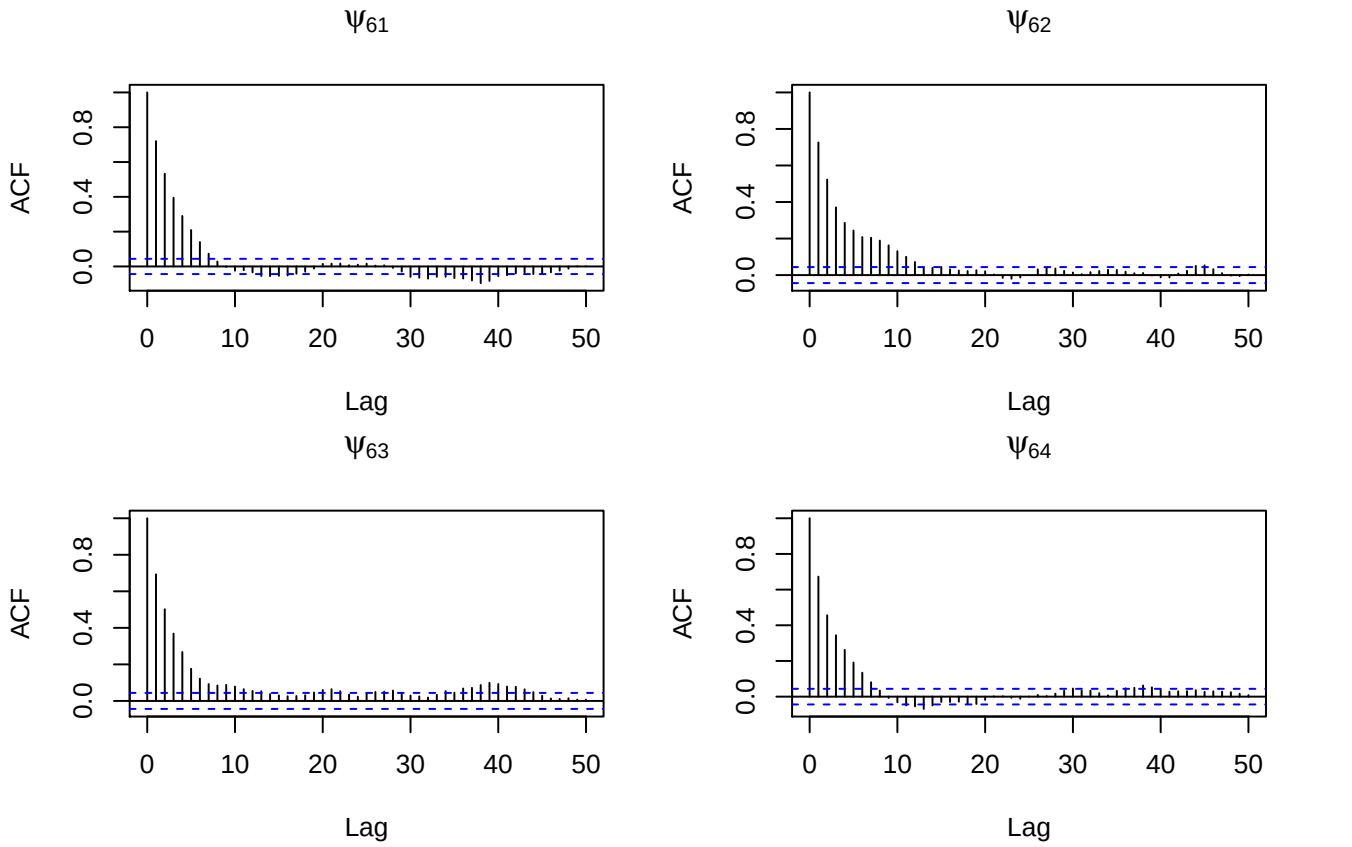

ψ51
ψ52
ψ53
ψ54
ψ61
ψ62
ψ63
ψ64
Iteration
0
500
1000
1500
2000

```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(i in 1:Irow){
        for(j in 1:Jcol){
                acf(psi.samp.new[,i,j],main=substitute(psi[ival][jval],
                list(ival=i,jval=j)),lag.max=50)
        }
}
```

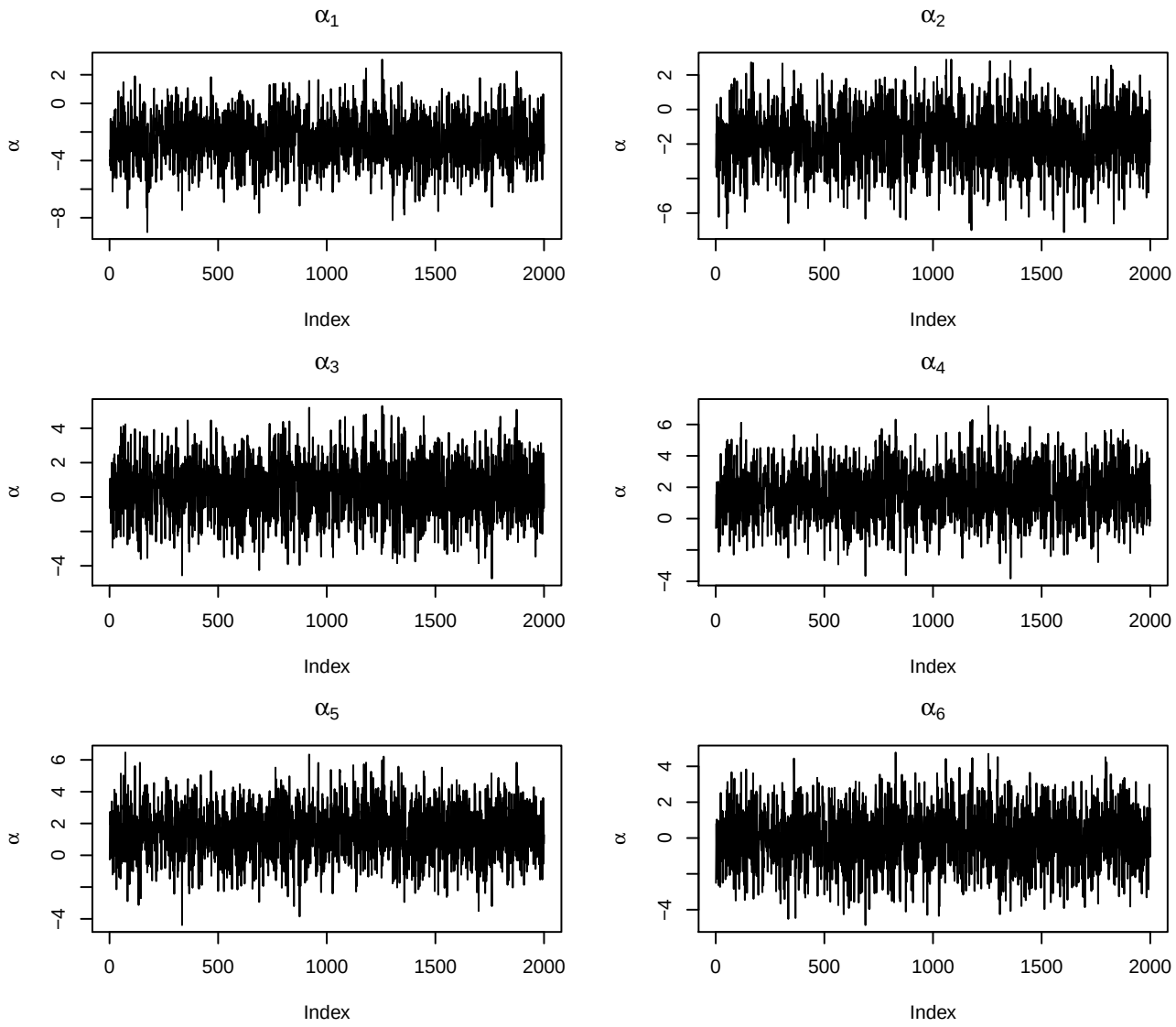



ψ21
ψ22
ψ23
ψ24
ψ31
ψ32
ψ33
ψ34
ACF
Lag
0.0
0.4
0.8
0
10
20
30
40
50

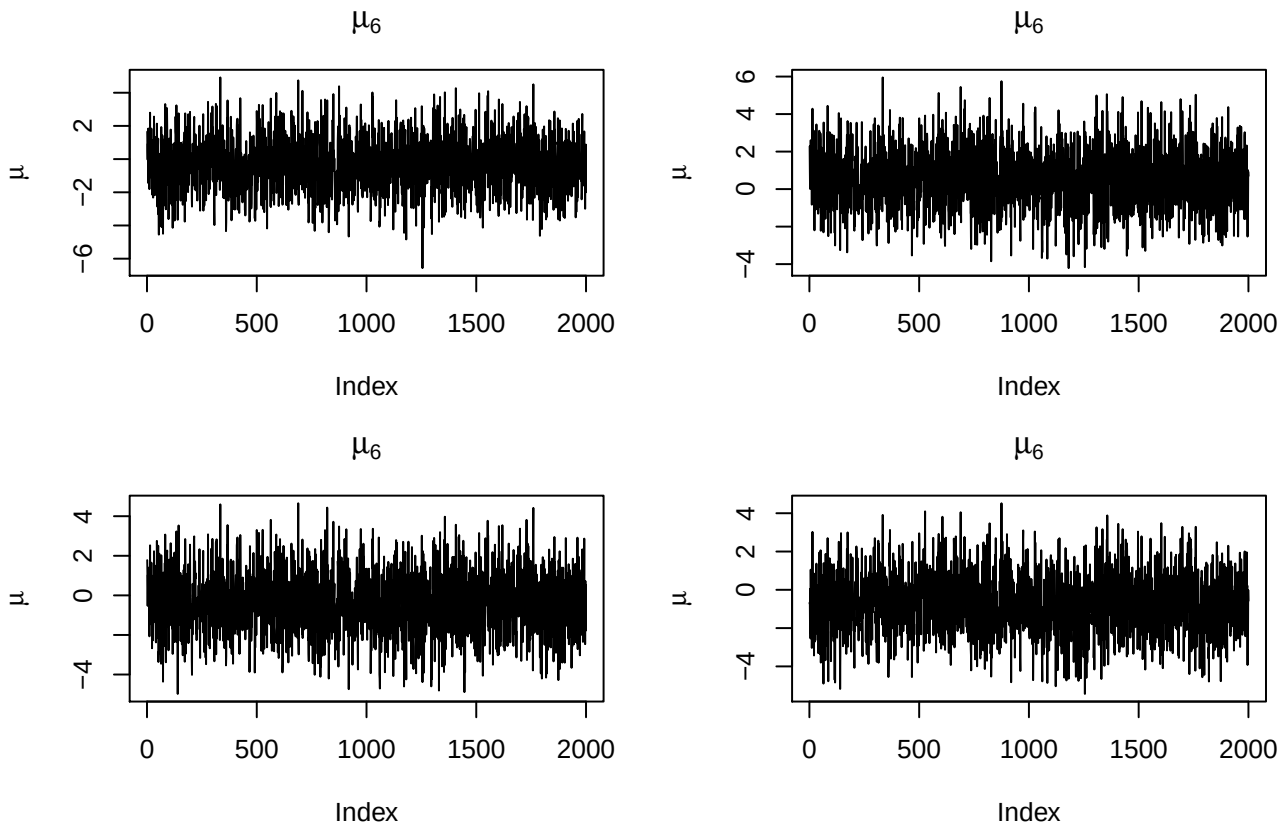

ψ41
ψ42
ψ43
ψ44
ψ51
ψ52
ψ53
ψ54
ACF
Lag
0.0
0.4
0.8
0
10
20
30
40
50



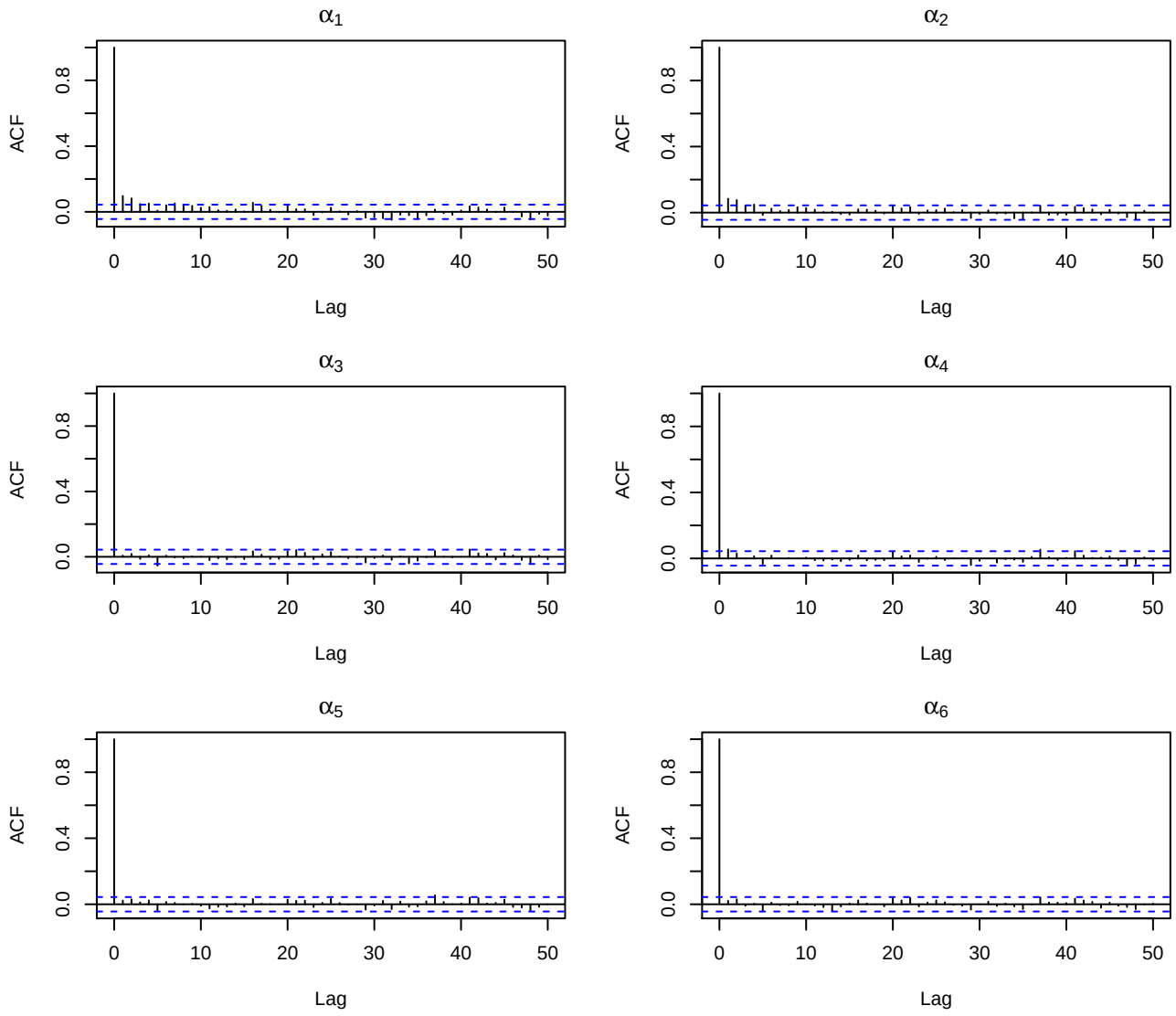

```
par(mfrow=c(3,2),mar=c(4,4,3,2))
for(i in 1:Irow){
        plot(al.samp.new[,i],type="l",ylab=expression(alpha),
            main=substitute(alpha[ival], list(ival=i)))
}
```



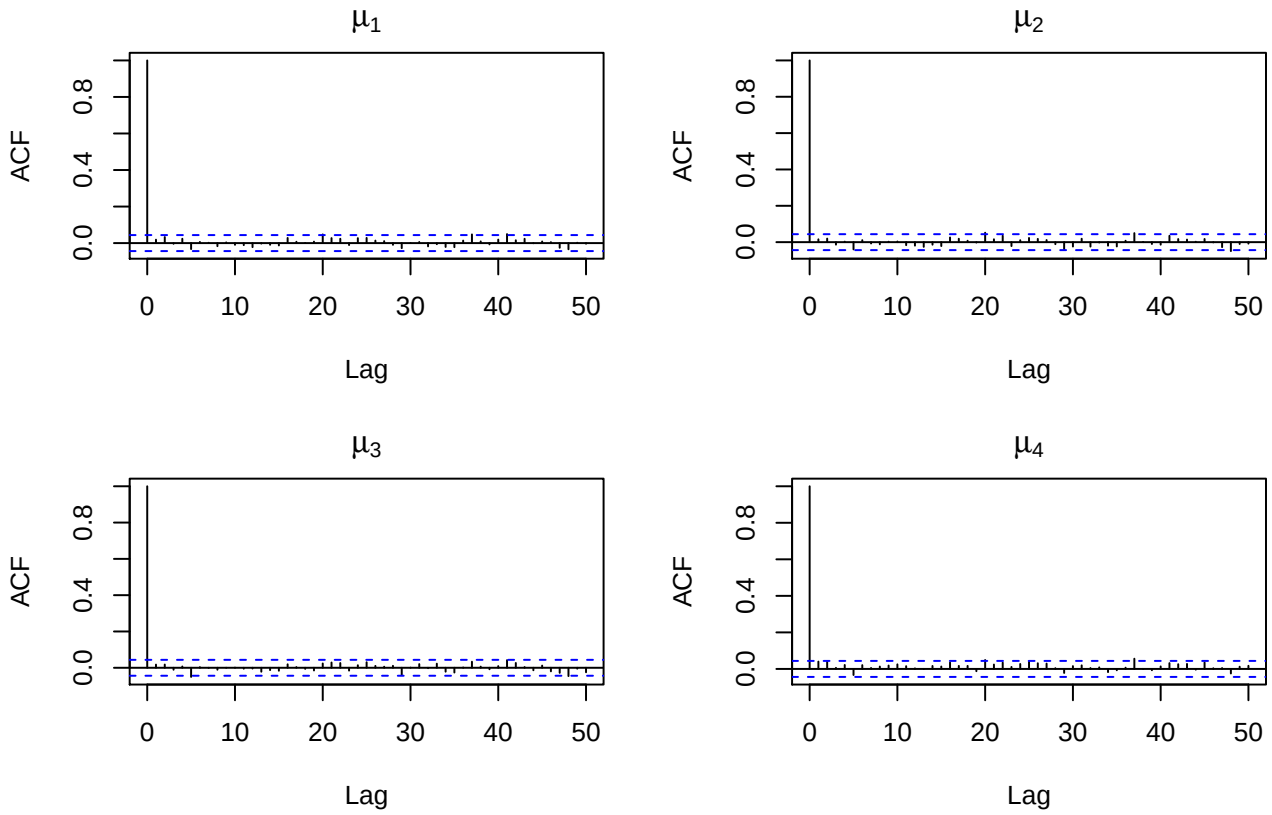

```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        plot(mu.samp.new[,j],type="l",ylab=expression(mu),
            main=substitute(mu[jval], list(jval=i)))
}
```



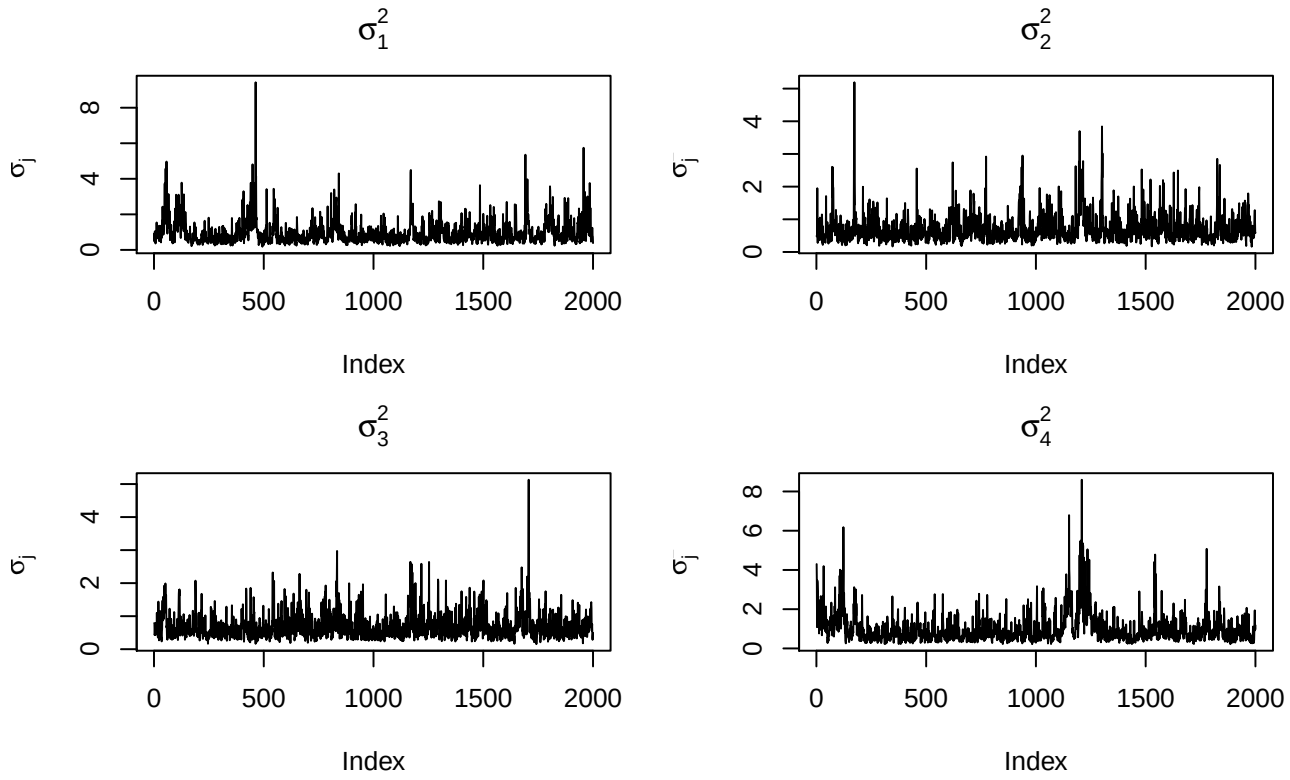

```
par(mfrow=c(3,2),mar=c(4,4,3,2))
for(i in 1:Irow){
        acf(al.samp.new[,i],main=' ',lag.max=50)
    title(substitute(alpha[ival], list(ival=i)),line=1)
}
```



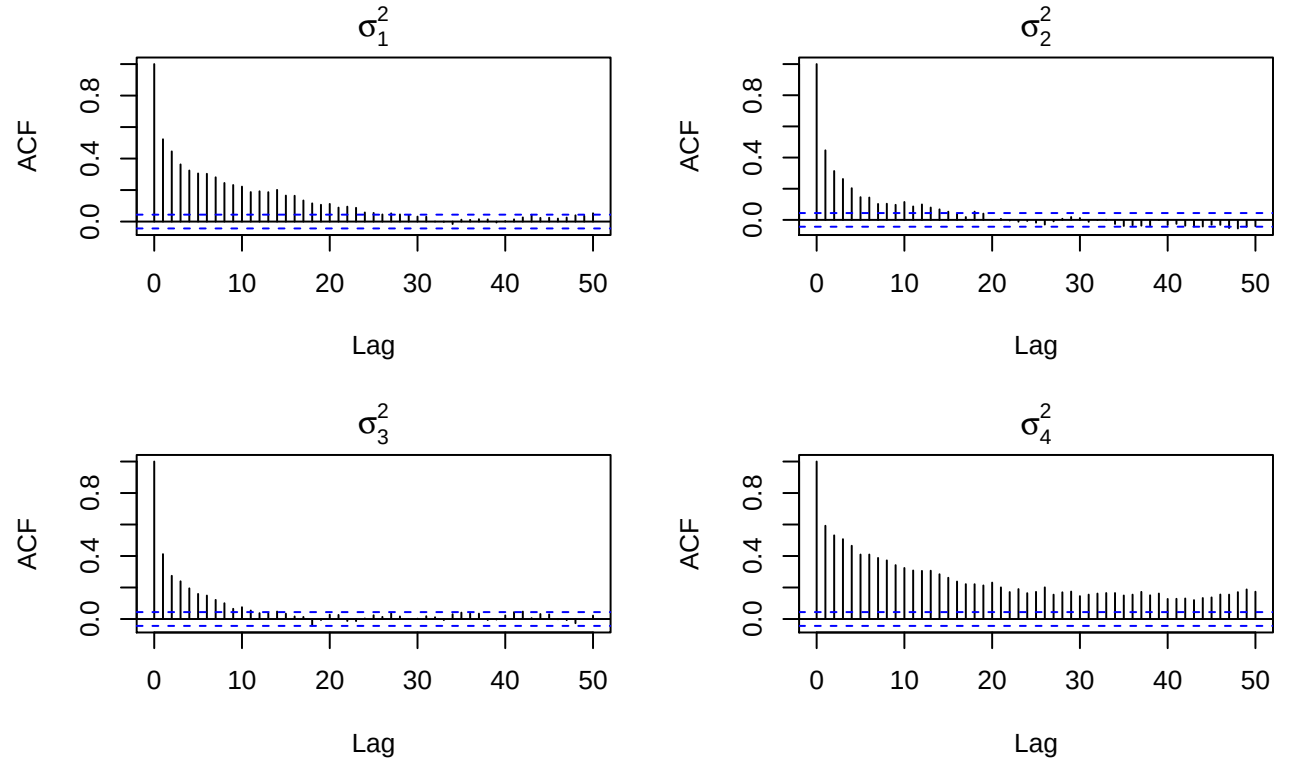

```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        acf(mu.samp.new[,j],main=' ',lag.max=50)
    title(substitute(mu[jval], list(jval=j)),line=1)
}
```




```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        plot(sig.samp.new[,j],type="l",ylab=expression(sigma[j]^2),
            main=substitute(sigma[jval]^2, list(jval=j)))
}
```



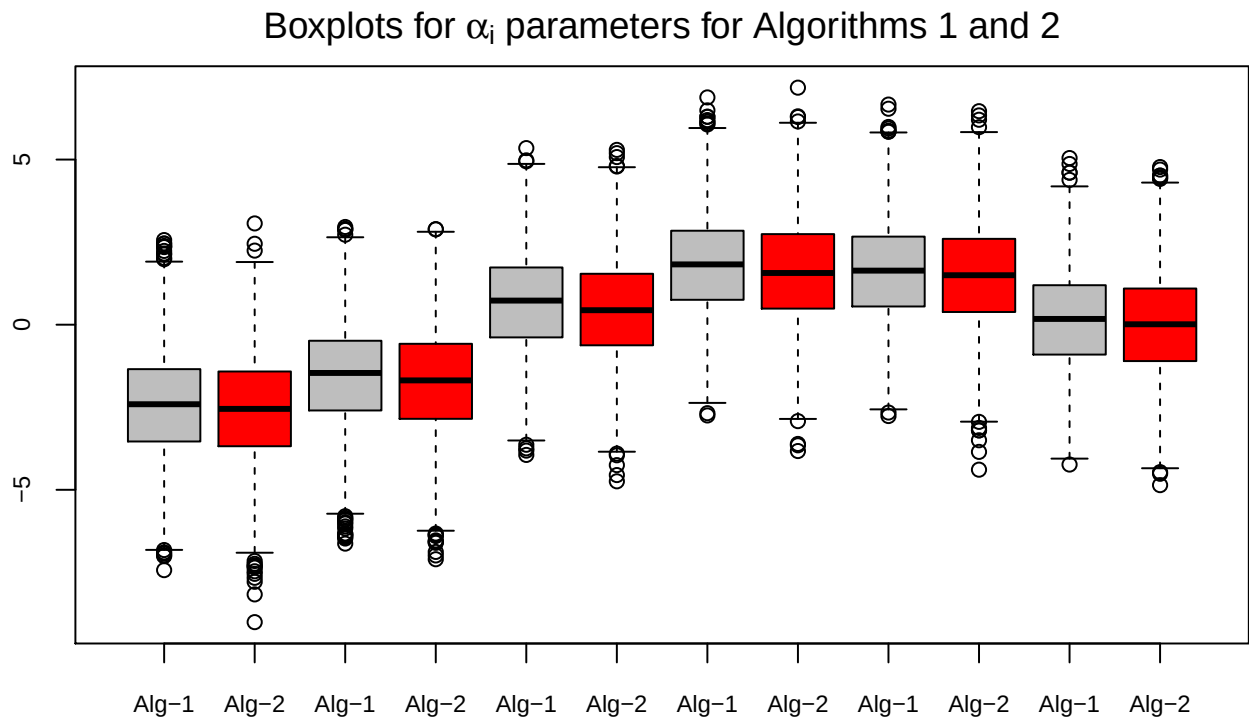

```
par(mfrow=c(2,2),mar=c(4,4,3,2))
for(j in 1:Jcol){
        acf(sig.samp.new[,j],main=' ',lag.max=50)
    title(substitute(sigma[jval]^2, list(jval=j)),line=1)
}
```



```
colnames(al.samp.new)<-c("al1","al2","al3","al4","al5","al6")
colnames(mu.samp.new)<-c("mu1","mu2","mu3","mu4")
colnames(sig.samp.new)<-c("sig1","sig2","sig3","sig4")
effectiveSize(al.samp.new)

+      al1      al2      al3      al4      al5      al6
+ 1228.588 1281.572 2000.000 1790.213 2000.000 2000.000

effectiveSize(mu.samp.new)

+      mu1      mu2      mu3      mu4
+ 2000.000 2000.000 2000.000 1695.609

effectiveSize(sig.samp.new)

+      sig1      sig2      sig3      sig4
+ 185.31886 325.65089 401.31422  79.55514

round(cor(cbind(al.samp.new,mu.samp.new)),3)

+        al1    al2    al3    al4    al5    al6    mu1    mu2    mu3    mu4
+ al1  1.000  0.794  0.883  0.886  0.856  0.876 -0.882 -0.905 -0.886 -0.885
+ al2  0.794  1.000  0.873  0.830  0.871  0.875 -0.877 -0.881 -0.899 -0.860
+ al3  0.883  0.873  1.000  0.904  0.912  0.927 -0.926 -0.936 -0.939 -0.924
+ al4  0.886  0.830  0.904  1.000  0.879  0.904 -0.918 -0.914 -0.921 -0.905
+ al5  0.856  0.871  0.912  0.879  1.000  0.909 -0.910 -0.923 -0.929 -0.920
+ al6  0.876  0.875  0.927  0.904  0.909  1.000 -0.928 -0.936 -0.937 -0.927
+ mu1 -0.882 -0.877 -0.926 -0.918 -0.910 -0.928  1.000  0.923  0.925  0.911
+ mu2 -0.905 -0.881 -0.936 -0.914 -0.923 -0.936  0.923  1.000  0.936  0.921
+ mu3 -0.886 -0.899 -0.939 -0.921 -0.929 -0.937  0.925  0.936  1.000  0.923
+ mu4 -0.885 -0.860 -0.924 -0.905 -0.920 -0.927  0.911  0.921  0.923  1.000
```

In the new blocked algorithm, we observe that the μ and α parameters are being sampled much more successfully.

```
al.both<-cbind(al.samp,al.samp.new)
al.both<-al.both[,c(1,7,2,8,3,9,4,10,5,11,6,12)]
mu.both<-cbind(mu.samp,mu.samp.new)
mu.both<-mu.both[,c(1,5,2,6,3,7,4,8)]
par(mar=c(4,4,2,0))
boxplot(al.both,col=c('gray','red'),names=rep(c('Alg-1','Alg-2'),6),cex.axis=0.75)
title(expression(paste('Boxplots for ',alpha[i],' parameters for Algorithms 1 and 2')))
```




```
boxplot(mu.both,col=c('gray','red'),names=rep(c('Alg-1','Alg-2'),4),cex.axis=0.75)
title(expression(paste('Boxplots for ',mu[j],' parameters for Algorithms 1 and 2')))
```