Appendix C: N-mixture models

Version 1.0

Contents

List of Tables

List of Figures

1 Simulation Study

1.1 Summary

This section describes the simulation study used for assessing the closure assumption of N-mixture models in the manuscript *A guide to Bayesian model checking for ecologists*.

1.2 Setting up the R workspace

Run the following script to install and load required packages and functions needed for the analysis.

```
rm(list=ls())

required.packages=c("coda",
                    "devtools",
                    "mcmcse",
                    "parallel",
                    "purrr",
                    "roxygen2",
                    "xtable"
                    )
## install.packages(required.packages)
lapply(required.packages,library,character.only=TRUE)

## Source the MCMC algorithm
source(paste("~/MCMCAlgorithmParallel.R",
             sep=""))

## Wrapper for parallel processing
## Also found in HierarchicalGOF package
run.chain.2pl.list=function(models,
                            Y.list,
                            X,
                            W,
                            n.iter,
                            checkpoint,
                            thin,
                            name.l,
                            starting.values
                            ){
    chain.list=mclapply(models,
                        function(m){
                            ## Set the seed for this core
                            this.Y=Y.list[[m]]
```

```
                            this.start=starting.values.l[[m]]
                            this.name=name.l[[m]]
                            ## Run the chain on this core
                            this.chain=run.Nmixmcmc.parallel(this.Y,
                                                             X,
                                                             W,
                                                             n.iter,
                                                             checkpoint,
                                                             thin,
                                                             this.name,
                                                             this.start
                                                             )

                            ## Return the chain from this core
                            return(this.chain)
                        },
                        mc.cores=min(length(models),detectCores())
                        )

    ## Save the initial random seed as the name of the chain
    names(chain.list)=models
    return(chain.list)
}
```

1.3 Simulate data for each value of c

```
###
### Simulate data
###

Y.list=list()
c.val=rep(seq(0,0.35,length.out=8),each=2)
seed=rep(1:(length(c.val)/2),each=2)
n=300
J=5
alpha=c(1,-1)
w1=rep(1,n)
w2=rbinom(n,1,0.5)
W=cbind(w1,w2)
p=exp(W%*%alpha)/(1+exp(W%*%alpha))
beta=c(4.5,1)
x1=rep(1,n)
```

```
x2=rbinom(n,1,0.5)
X=cbind(x1,x2)
lambda=exp(X%*%beta)
N.true=matrix(,n,J)
N.true[,1]=rpois(n,lambda)
for(i in 1:length(c.val)){
    set.seed(seed[i])
    for(j in 2:J){
        N.true[,j]=mapply(
            rdunif,
            1,
            round(N.true[,j-1]*(1+c.val[i])),
            round(N.true[,j-1]*(1-c.val[i])))
    }
    Y.list[[i]]=matrix(rbinom(J*n,N.true,p),n,J)
}
starting.values.l=list()
for(i in 1:length(c.val)){
    if((i/1)%%1==0){
        starting.values.l[[i]]=list(alpha+0.1,
                                    beta+0.1,
                                    apply(Y.list[[i]],1,max))
    }
    if((i/2)%%1==0){
        starting.values.l[[i]]=list(alpha-0.1,
                                    beta-0.1,
                                    apply(Y.list[[i]],1,max))
    }
}

name.l=list()
for(i in 1:length(c.val)){
    name.l[[i]]=paste("~/SimExample",i,".RData",sep="")
}
save.image(paste("~/SimulatedData.RData",sep=""))
```

1.4 Run MCMC algorithms

In this example, 16 MCMC algorithms are run in parallel to fit 16 different models (two chains for each of 8 values of c) which requires 16 cores. If 16 cores are not available, models must be fit sequentially using a smaller number of cores. This may take a while.

```
###
### Run algorithm
###

n.iter=10000000
checkpoint=100000
thin=1
models=1:16

MCMCOutput=run.chain.2pl.list(models,
                              Y.list,
                              X,
                              W,
                              n.iter,
                              checkpoint,
                              thin,
                              name.l,
                              starting.values.l
                              )

## Save model fitting results
save(MCMCOutput,file=paste("~/MCMCOutput.RData",sep=""))
```

1.5 Summarize results

```
##
## Load output and calculate results
##

load(paste("~/SimulatedData.RData",
           sep=""))

##
## Create empty containers for output
##

GR.Diag=numeric(8)
ESS=numeric(8)
Status=numeric(8)
Bayes.p=numeric(8)
Mean.N=numeric(8)  # Truth = 50989
LB=numeric(8)
```

```
UB=numeric(8)

##
## Calculate summaries for each of 8 model fits with different c values
##

for(i in 1:8){

    w1=new.env()
    load(paste("~/SimExample",
               i*2-1,
               ".RData",sep=""),
         envir=w1)
    w2=new.env()
    load(paste("~/SimExample",
               i*2,
               ".RData",sep=""),
         envir=w2)
    status=sum(!is.na(w2$out[,1]))
    thin=1000    # large thinning value required due to autocorrelation
    burn=100000
    ind=seq(burn+1,status,thin)
    length(ind)
    N.tot1=w1$out[ind,5]
    N.tot2=w2$out[ind,5]
    Mean.N[i]=mean(w1$out[ind,5])
    LB[i]=quantile(w1$out[ind,5],0.025)
    UB[i]=quantile(w1$out[ind,5],0.975)

    ##
    ## Gelman Rubin Diagnostic
    ##

    chain1=mcmc(w1$out[1:status,1:4])
    chain2=mcmc(w2$out[1:status,1:4])
    out.list=mcmc.list(chain1,chain2)
    GR.Diag[i]=gelman.diag(out.list,confidence = 0.95,
                           transform=FALSE,autoburnin=TRUE)[2]

    ##
    ## Effective sample size
    ##

    ESS[i]=min(ess(chain1))
```

```
    Status[i]=status

    ##
    ## Bayesian p-value
    ##

    T.mcmc.chi2=w1$out[ind,6]
    T.data.chi2=w1$out[ind,7]
    Bayes.p[i]=sum(T.mcmc.chi2>=T.data.chi2,na.rm=TRUE)/length(ind)

}

##
## Sampled posterior predictive value
##

sppv=rep(NA,8)
for(i in 1:8){
    w1=new.env()
    load(paste("~/SimExample",
               i*2-1,
               ".RData",sep=""),
         envir=w1)
    w2=new.env()
    load(paste("~/SimExample",
               i*2,
               ".RData",sep=""),
         envir=w2)
    (status=sum(!is.na(w2$out[,1])))
    thin=100
    burn=100000
    ind=seq(burn+1,status,thin)
    length(ind)

    param.vec.id=sample(ind,1)
    param.vec=w1$out[param.vec.id,1:4]
    alpha.sppv=param.vec[1:2]
    p.sppv=exp(W%*%alpha.sppv)/(1+exp(W%*%alpha.sppv))
    beta.sppv=param.vec[3:4]
    lambda.sppv=exp(X%*%beta.sppv)
    N.sppv=matrix(,n,1)
    reps=1000
    T.mcmc.sppv.chi2=numeric(reps)
    T.data.sppv.chi2=numeric(reps)
```

```
    N.sppv[,1]=rpois(n,lambda.sppv)
    for(k in 1:reps){
        N.sppv[,1]=rpois(n,lambda.sppv)
        y.sppv=matrix(rbinom(J*n,N.sppv,p.sppv),n,J)
        T.mcmc.sppv.chi2[k]=sum(apply(y.sppv,1,var))
        T.data.sppv.chi2[k]=sum(apply(Y.list[[i]],1,var))
    }
    sppv[i]=sum(T.mcmc.sppv.chi2>=T.data.sppv.chi2)/reps
}
```

1.6 Model checking results table for simulated data

```
##
## Create a table of results
##

GR=unlist(GR.Diag)
p.value=Bayes.p
c=unique(c.val)
xtable.data=cbind(c,p.value,sppv,Mean.N,LB,UB,GR,ESS)
print(xtable(xtable.data), include.rownames=FALSE)
```

c	p-value	sppv	Abundance (truth=50,989)	95% CRI	GR	ESS
0.00	0.48	0.27	51,200	(49,295, 53,481)	1.00	3,420
0.05	0.40	1.00	60,047	(56,605, 63,868)	1.00	3,260
0.10	0.00	1.00	81,299	(75,223, 89,601)	1.01	3,194
0.15	0.00	1.00	97,066	(89,149, 104,360)	1.13	3,199
0.20	0.00	0.02	117,624	(108,825, 127,007)	1.03	3,184
0.25	0.00	0.01	119,397	(110,477, 125,992)	1.06	3,206
0.30	0.00	0.00	133,797	(124,194, 141,117)	1.10	3,195
0.35	0.00	0.00	139,951	(133,351, 147,086)	1.00	3,213

Table 1: Results of one simulation for examining the effect of the closure assumption on model fit. The notation c represents the maximum proportion of the population that could move in or out of a site between $j-1$ and j, p-value is the posterior predicitive p-value using a χ-squared goodness-of-fit statistic, sppv is the sampled predictive p-value using the sum of variance test statistic, Abundance is the mean of the marginal posterior distribution for total abundance at the 300 sites, the 95% CRI are the 95% credible intervals, GR is the multi-variate Gelman-Rubin convergence diagnostic, and ESS is the effective sample size of 10,000,000 MCMC iterations.

1.7 Plot MCMC output

The script used for plotting MCMC output is provided for one simulation ($c = 0$), below.

```
##
## Figures c=0
##

w1=new.env()
load(paste("~/SimExample",
           1,
           ".RData",sep=""),
     envir=w1)
w2=new.env()
load(paste("~/SimExample",
           2,
           ".RData",sep=""),
     envir=w2)

par(mfrow=c(5,2),mar=c(3,4,0,1))
plot(w1$out[ind,1],type='l',
     ylim=c(min(w1$out[ind,1]),max(1,max(w1$out[ind,1]))),
     ylab=expression(alpha[0]))
lines(w2$out[ind,1],col=3)
abline(h=1,col=2)
plot(density(w1$out[ind,1]),
     xlim=c(min(w1$out[ind,1]),max(1,max(w1$out[ind,1]))),
     main='')
abline(v=1,col=2)
plot(w1$out[ind,2],type='l',
         ylim=c(min(-1,min(w1$out[ind,2])),max(w1$out[ind,2])),
         ylab=expression(alpha[1]))
lines(w2$out[ind,2],col=3)
abline(h=-1,col=2)
plot(density(w1$out[ind,2]),
         xlim=c(min(-1,min(w1$out[ind,2])),max(w1$out[ind,2])),
         main='')
abline(v=-1,col=2)
plot(N.tot1,type='l',
         ylim=c(min(min(N.tot1),sum(N.true[,1])),max(N.tot1)),
         ylab='N')
lines(N.tot2,col=3)
abline(h=sum(N.true[,1]),col=2)
plot(density(N.tot1),main='',
         xlim=c(min(min(N.tot1),sum(N.true[,1])),max(N.tot1)))
```

```
abline(v=sum(N.true[,1]),col=2)
plot(w1$out[ind,3],type='l',
         ylim=c(min(4.5,min(w1$out[ind,3])),max(w1$out[ind,3])),
         ylab=expression(beta[0]))
lines(w2$out[ind,3],col=3)
abline(h=4.5,col=2)
plot(density(w1$out[ind,3]),main='',
         xlim=c(min(4.5,min(w1$out[ind,3])),max(w1$out[ind,3])))
abline(v=4.5,col=2)
plot(w1$out[ind,4],type='l',
         ylab=expression(beta[1]))
lines(w2$out[ind,4],col=3)
abline(h=1,col=2)
plot(density(w1$out[ind,4]),
         main='')
abline(v=1,col=2)
```

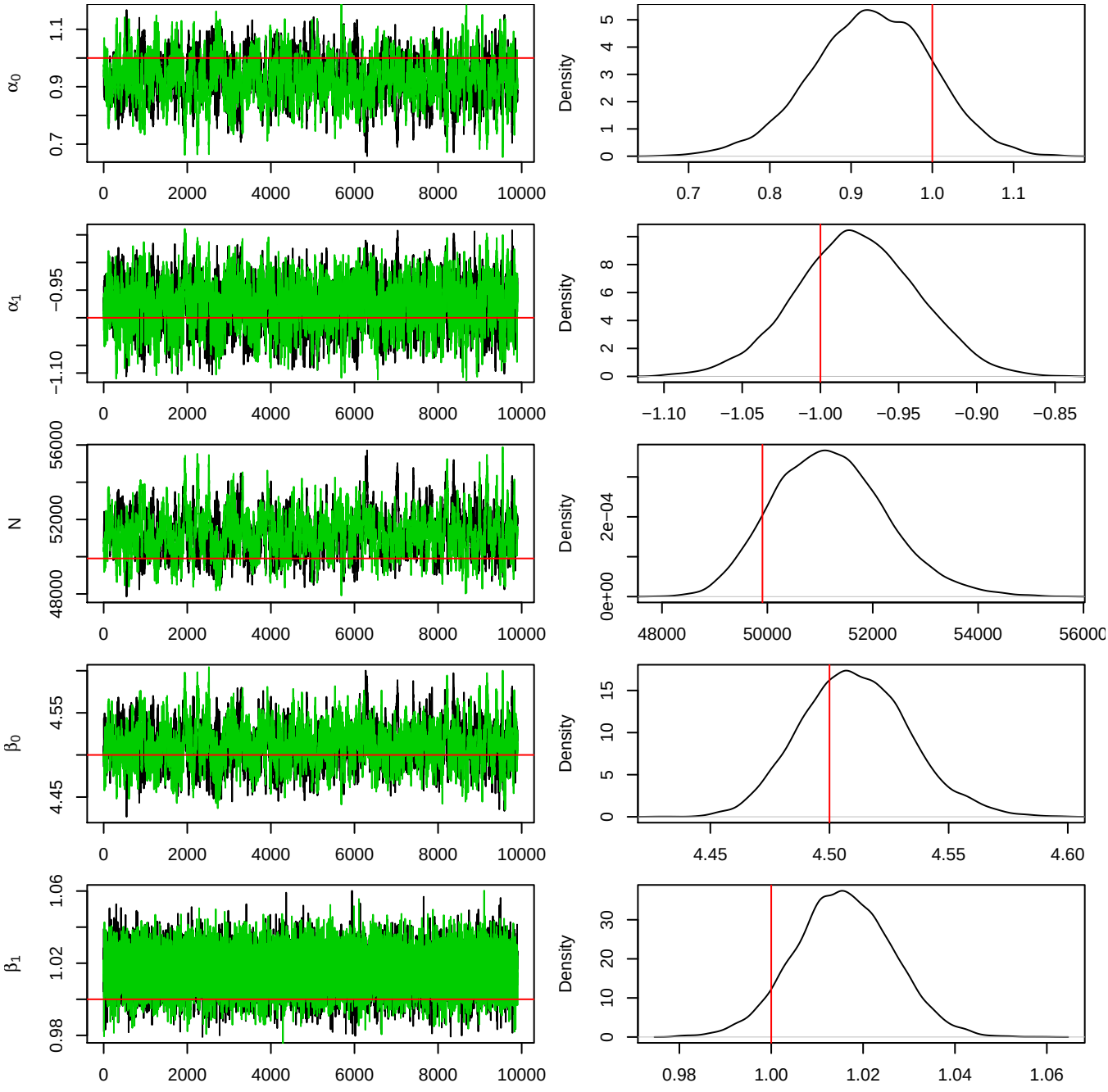


Figure 1: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0$ (i.e., the population was closed). Using a Gelman-Rubin diagnostic, there is no evidence that the model failed to converge (GR=1.00), and posterior distributions recovered true parameter values well. Both the posterior predictive p-value (0.48) and the sampled predictive p-value (0.27) suggested no lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration.

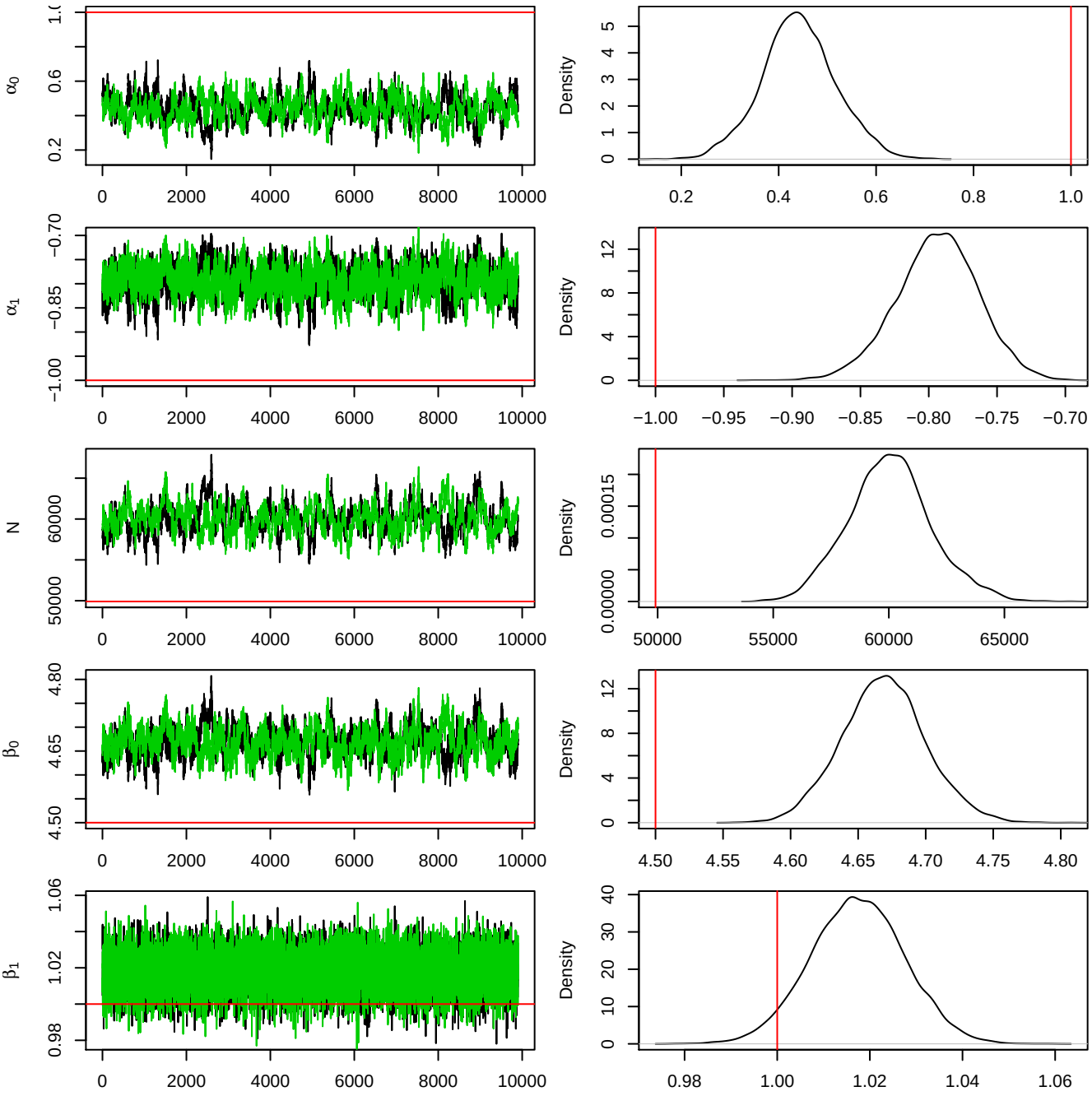


Figure 2: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.05$ (i.e., up to 5% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). Using a Gelman-Rubin diagnostic, there is no evidence that the model failed to converge (GR=1.00), but posterior distributions did not recover true parameter values well. The posterior predictive p-value (0.40) suggested no lack of model fit, but the sampled predictive p-value (1.00) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration.

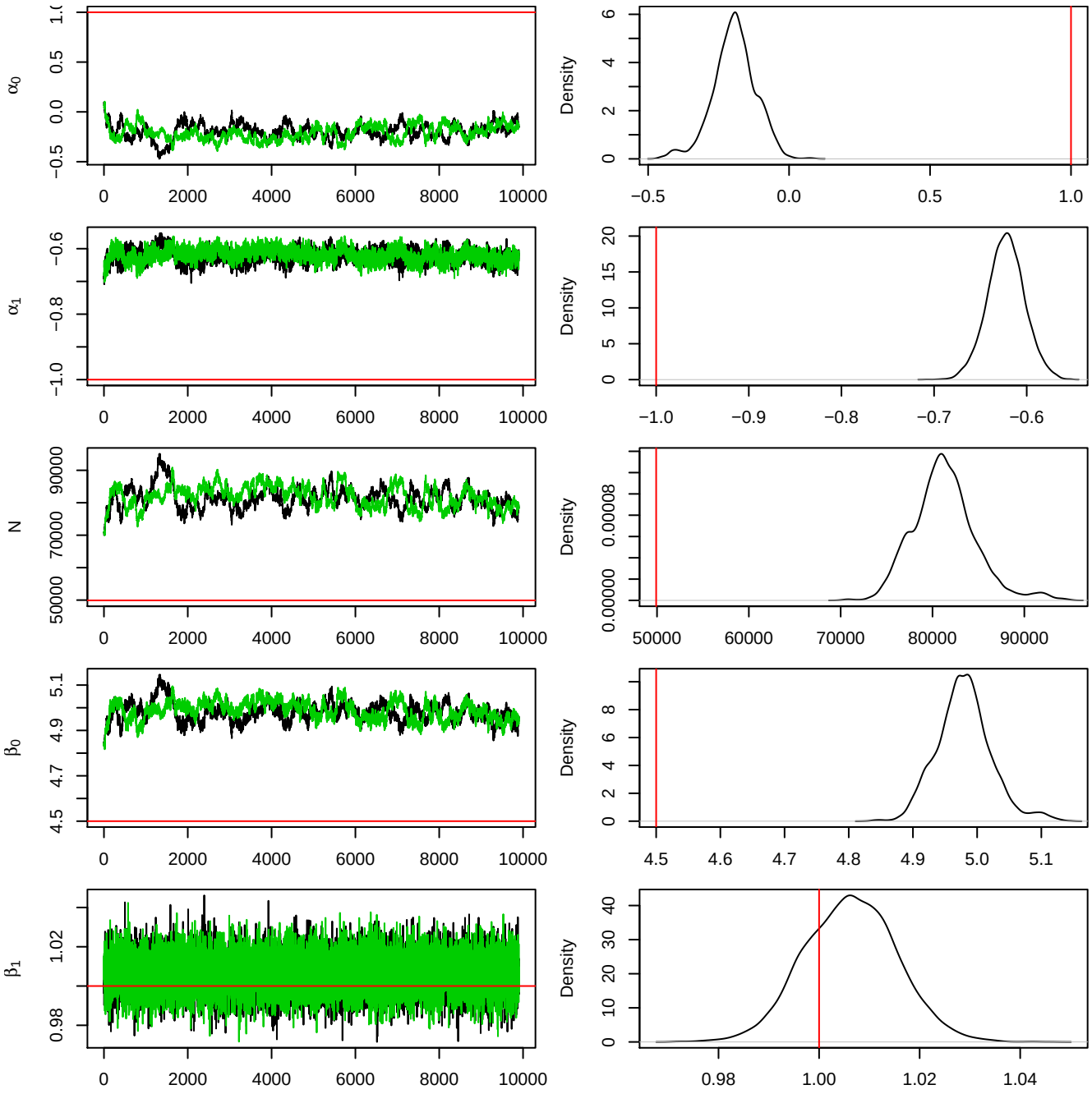


Figure 3: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.10$ (i.e., up to 10% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). Using a Gelman-Rubin diagnostic, there is no evidence that the model failed to converge (GR=1.01), but posterior distributions did not recover true parameter values well. Both the posterior predictive p-value (0.00) and the sampled predictive p-value (1.00) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration, with a 100,000 burn-in period.

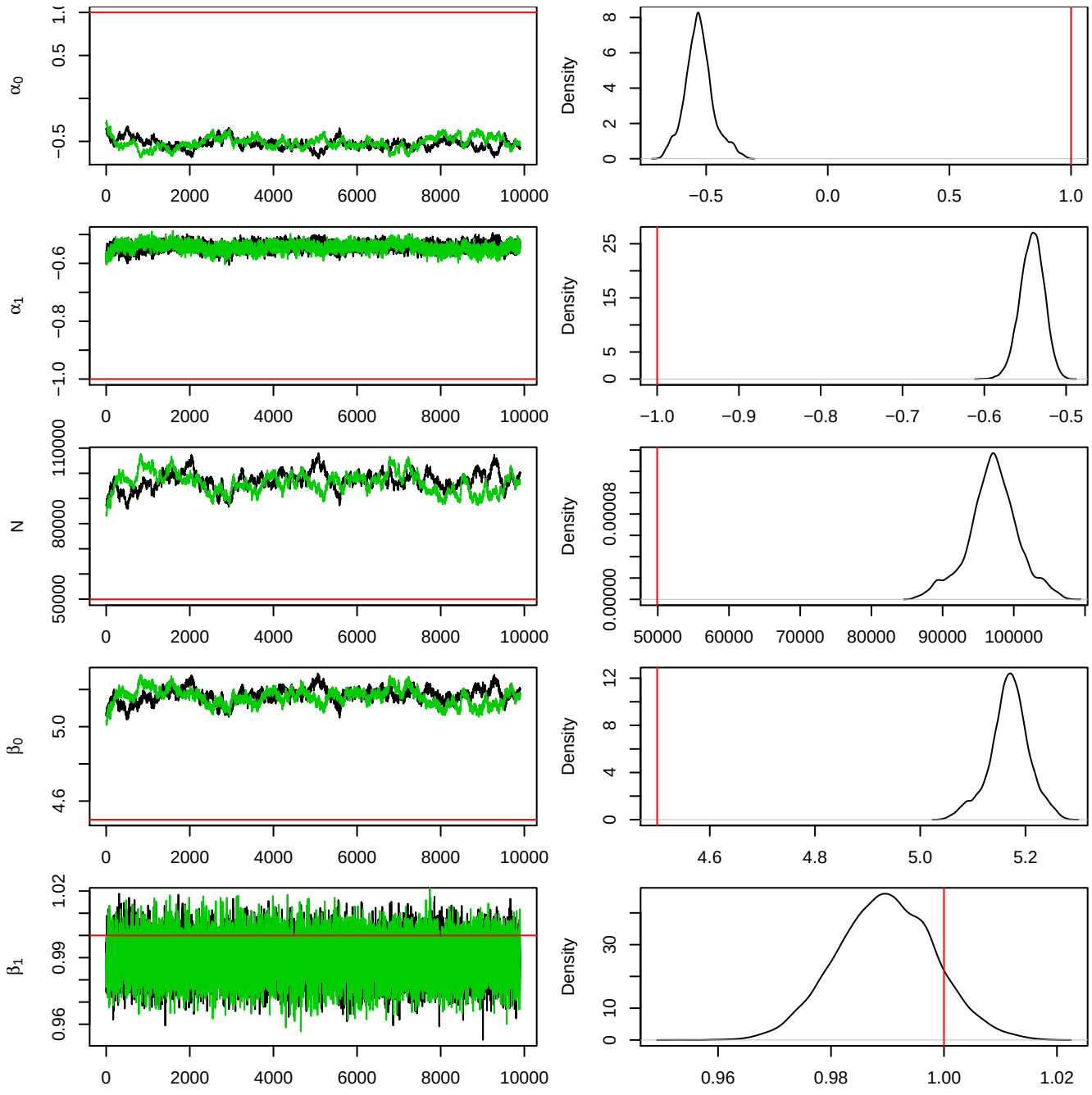


Figure 4: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.15$ (i.e., up to 15% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). The model did not appear to converge (Gelman-Rubin diagnostic=1.13). Both the posterior predictive p-value (0.00) and the sampled predictive p-value (1.00) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration, with a 100,000 burn-in period.

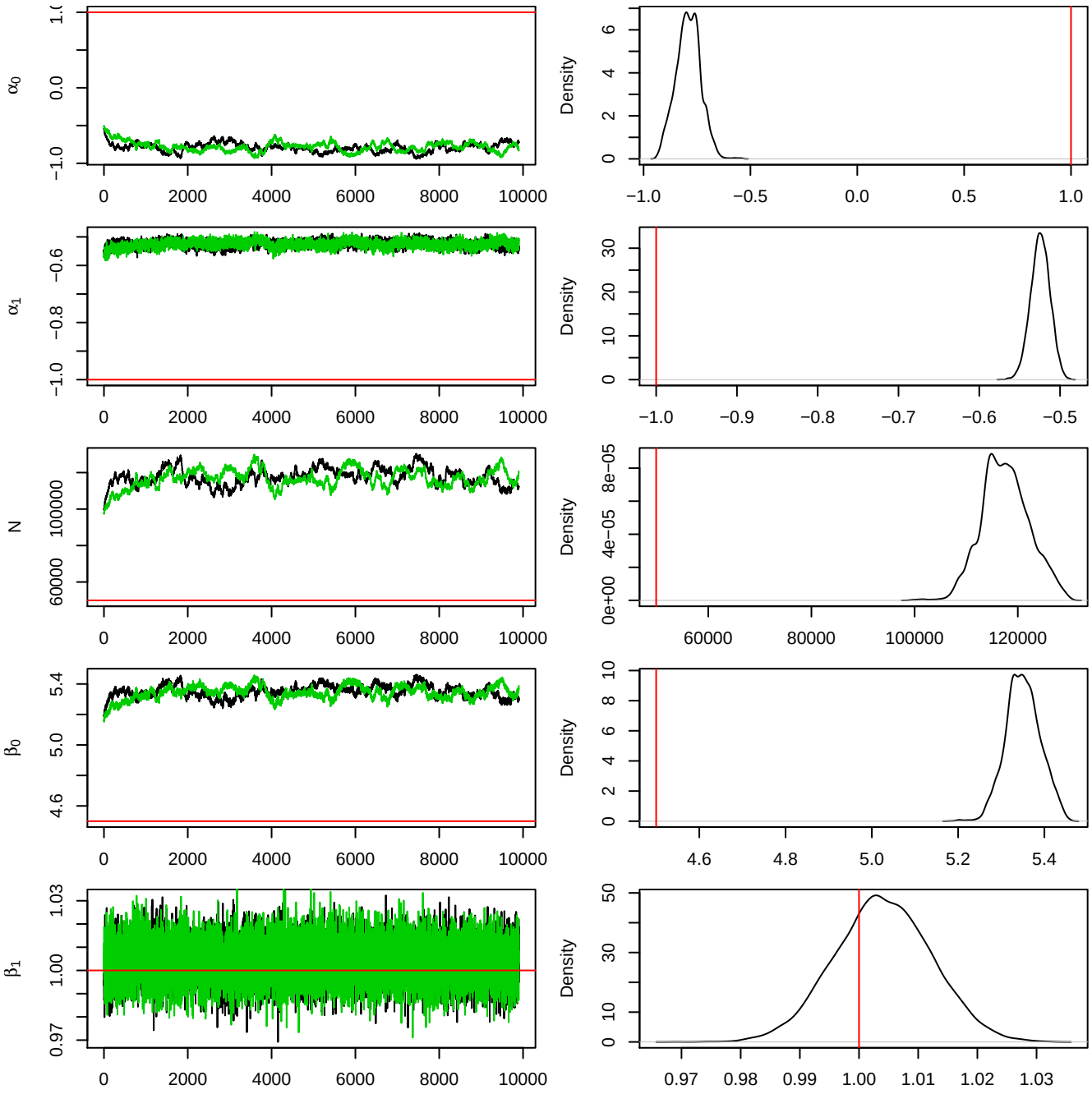


Figure 5: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.20$ (i.e., up to 20% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). Using a Gelman-Rubin diagnostic, there is no evidence that the model failed to converge (GR=1.03), but posterior distributions did not recover true parameter values well. Both the posterior predictive p-value (0.00) and the sampled predictive p-value (0.02) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration, with a 100,000 burn-in period.

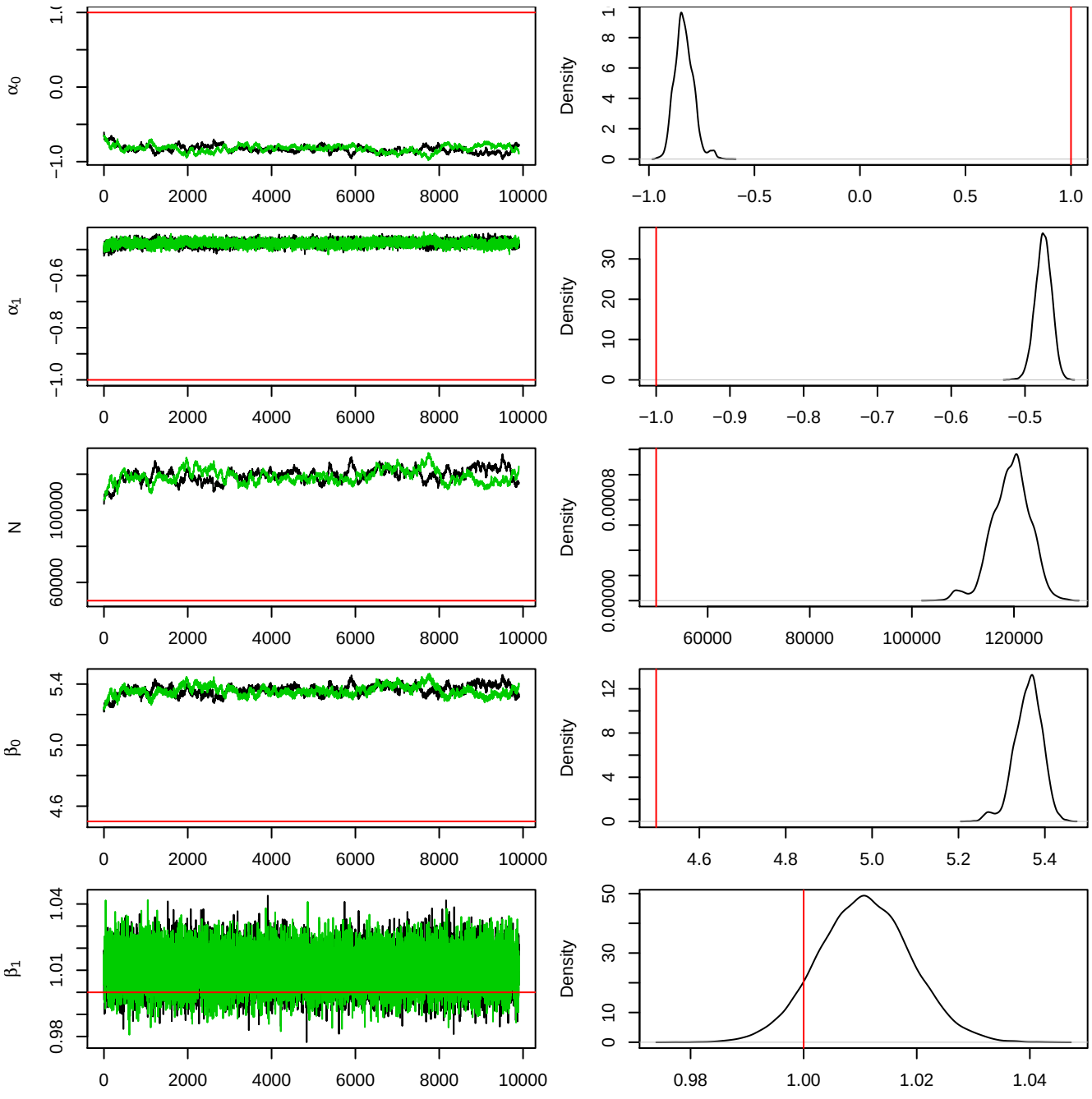


Figure 6: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.25$ (i.e., up to 25% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). Using a Gelman-Rubin diagnostic, there is some evidence that the model failed to converge (GR=1.06). Both the posterior predictive p-value (0.00) and the sampled predictive p-value (0.01) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration, with a 100,000 burn-in period.

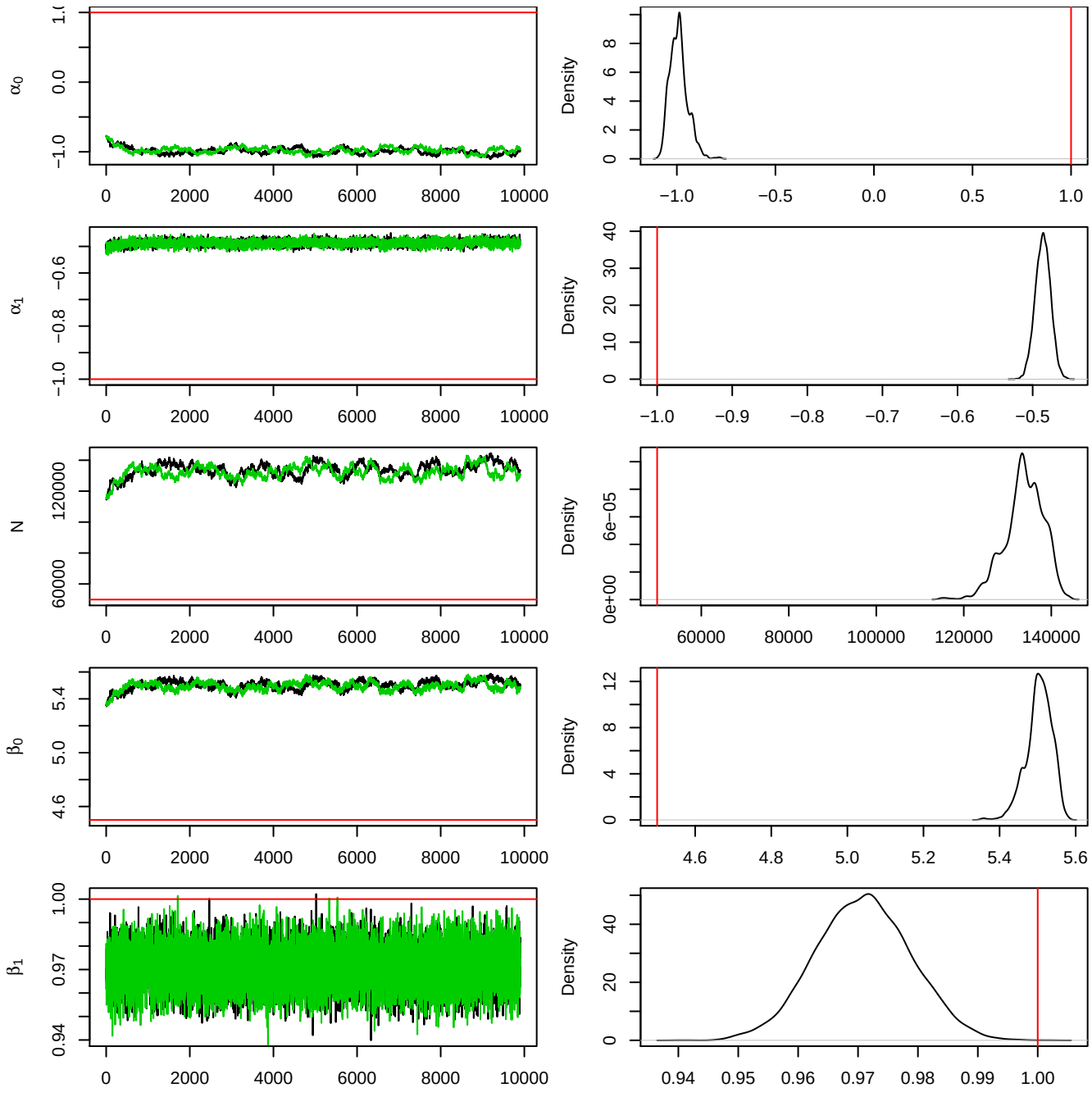


Figure 7: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.30$ (i.e., up to 30% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). The model did not appear to converge (Gelman-Rubin diagnostic=1.10). Both the posterior predictive p-value (0.00) and the sampled predictive p-value (0.00) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration, with a 100,000 burn-in period.

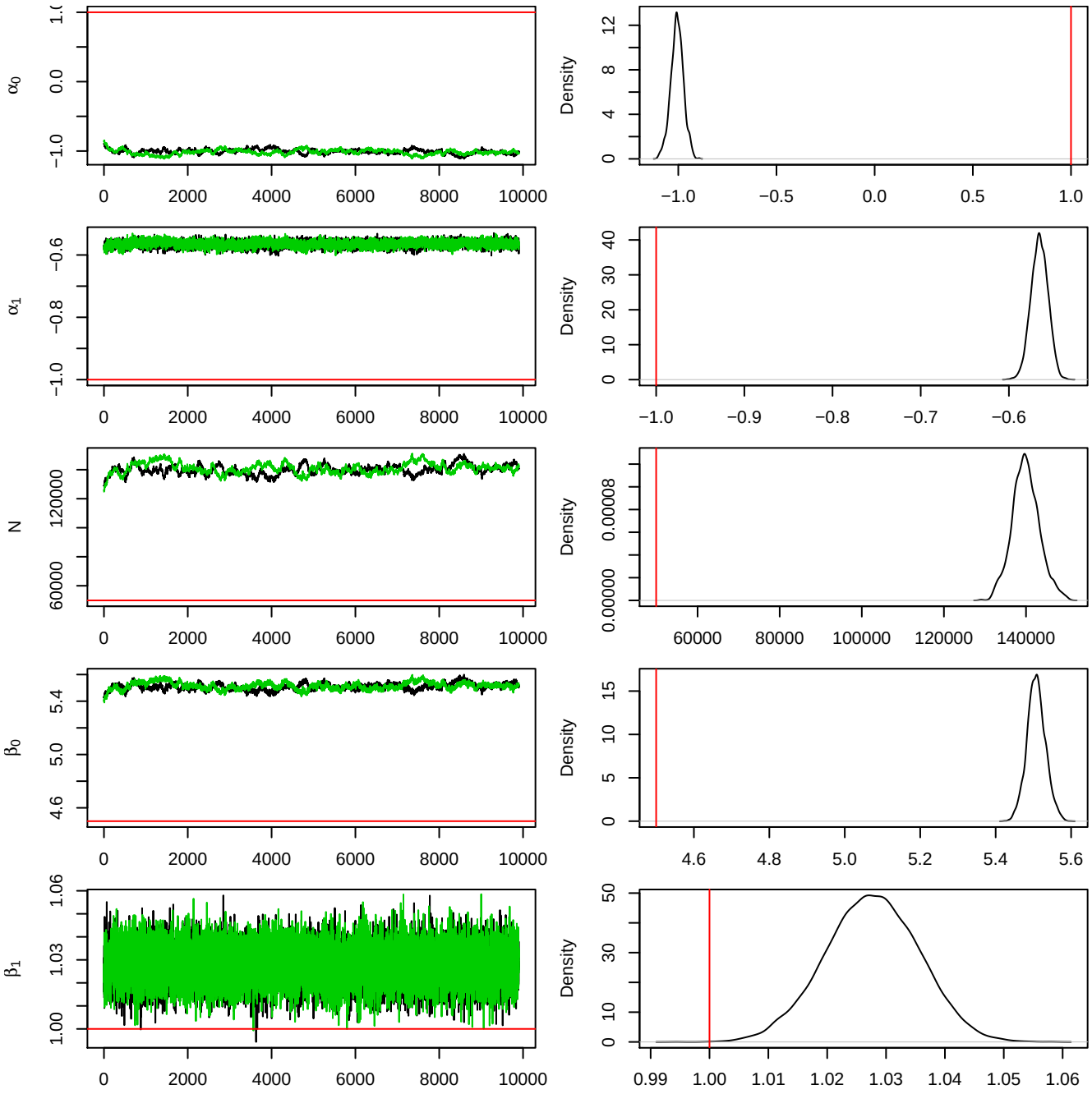


Figure 8: Trace plots and marginal posterior distributions of parameters in N-mixture model when $c = 0.35$ (i.e., up to 35% of $N_{i,j-1}$ was allowed to leave or enter the population at time j). Using a Gelman-Rubin diagnostic, there is no evidence that the model failed to converge (GR=1.00), but posterior distributions did not recover true parameter values well. Both the posterior predictive p-value (0.00) and the sampled predictive p-value (0.00) suggested lack of model fit. 10,000,000 MCMC iterations were conducted and thinned to every 1,000 iteration, with a 100,000 burn-in period.

2 Sea Otters

2.1 Summary

This section describes the sea otter study used for assessing the closure assumption of N-mixture models in the manuscript *A guide to Bayesian model checking for ecologists*.

2.2 All data

In what follows we conduct model checking using 21 sites sea otters were observed at in Glacier Bay National Park, including one site where observers noted a violation of the closure assumption. In the next section, we remove the site and conduct model checking.

```
rm(list=ls())

name="~/MCMCOutputAllDataChain1.RData"

source(paste("~/Dropbox/MCMCAlgorithms/NMixtureModel/",
             "SpatialVaryingNMixtureModel/MCMCAlgorithm.R",
             sep=""))

## Load Data
load(paste("~/Dropbox/GitHub/HierarchicalGOF/",
             "HierarchicalGOF/data/SeaOtterData.RData",sep=""))

## Use all the data including the 19th row of data where sea otters
## were observed violating the closure assumption
Y=SeaOtterData

## Priors
q.p=1
r.p=1
alpha=0.001
beta=0.001
n.iter=10000000
thin=1
checkpoint=1000000
```

2.3 Run MCMC algorithm

Fit a simple N-mixture model with no covariates.

```
## Run algorithm
Nmixmcmc(Y=Y,q.p,r.p,alpha,beta,n.iter,checkpoint,name,thin)
```

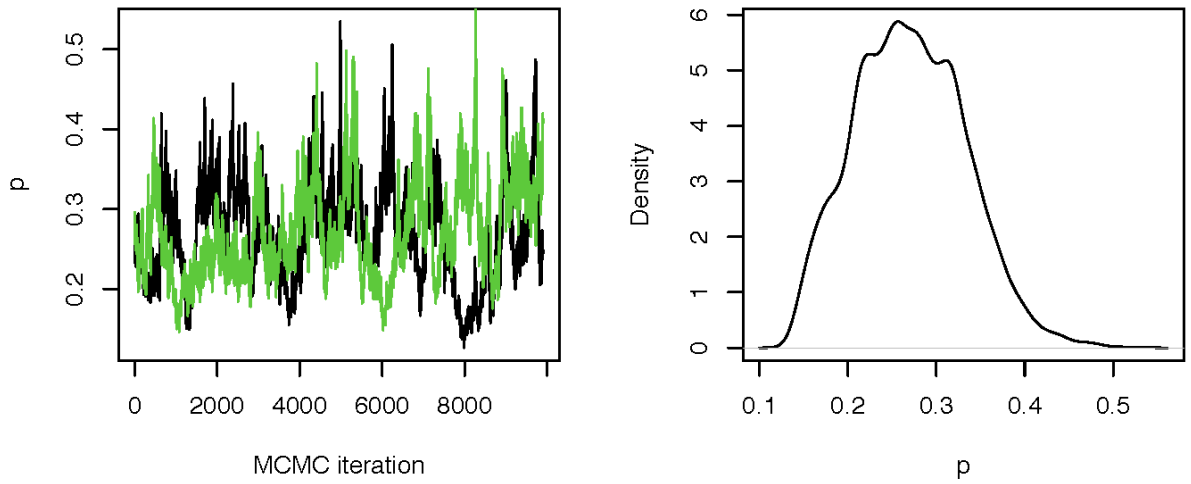
2.4 Summarize results using all data

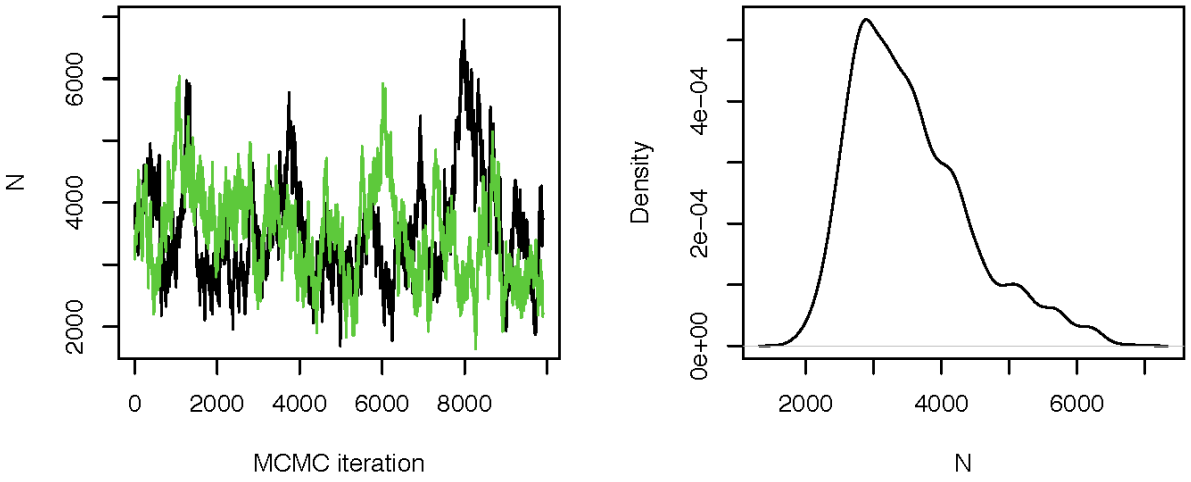
```
##
## Load output and calculate results
##

w1=new.env()
load(paste("~/MCMCOutputAllDataChain1.RData",
           sep=""),
     envir=w1)
w2=new.env()
load(paste("~/MCMCOutputAllDataChain2.RData",
           sep=""),
     envir=w2)
(status=sum(!is.na(w2$out[[1]][,1])))

## [1] 10000000

thin=1000    # large thinning value required due to autocorrelation
burn=100000
ind=seq(burn+1,status,thin)
length(ind)

## [1] 9900

N.tot1=w1$out[[4]][ind]
N.tot2=w2$out[[4]][ind]
Mean.N=mean(N.tot1)
LB=quantile(N.tot1,0.025)
UB=quantile(N.tot1,0.975)

##
## Gelman Rubin Diagnostic
##

mcmc1.tmp=cbind(w1$out[[1]][1:status,],w1$out[[3]][1:status,])
mcmc2.tmp=cbind(w2$out[[1]][1:status,],w2$out[[3]][1:status,])
chain1=mcmc(mcmc1.tmp)
chain2=mcmc(mcmc2.tmp)
out.list=mcmc.list(chain1,chain2)
(GR.Diag=gelman.diag(out.list,confidence = 0.95,
                     transform=FALSE,autoburnin=TRUE)[2])

## $mpsrf
## [1] 1.030395
```

```
##
## Effective sample size
##

(ESS=min(ess(chain1)))

## [1] 3210.614

##
## Bayesian p-value
##

T.mcmc.chi2=w2$out[[5]][ind]
T.data.chi2=w2$out[[6]][ind]
(Bayes.p=sum(T.mcmc.chi2>=T.data.chi2,na.rm=TRUE)/length(ind))

## [1] 0.04787879

##
## Sampled posterior predictive value
##

n=dim(Y)[1]
J=dim(Y)[2]
param.vec.id=sample(ind,1)
p.sppv=w1$out[[1]][param.vec.id]
lambda.sppv=w1$out[[3]][param.vec.id,]
N.sppv=w1$out[[2]][param.vec.id,]
Expected.Y=matrix(N.sppv*p.sppv,dim(Y)[1],dim(Y)[2])
reps=100000
T.mcmc.sppv.chi2=numeric(reps)
T.data.sppv.chi2=numeric(reps)
for(k in 1:reps){
    y.sppv=matrix(rbinom(n*J,N.sppv,p.sppv),n,J)
    y.sppv[is.na(Y)]=NA
    ## T.mcmc.sppv.chi2[k]=sum(apply(y.sppv,1,var,na.rm=TRUE))
    ## T.data.sppv.chi2[k]=sum(apply(Y,1,var,na.rm=TRUE))
    T.mcmc.sppv.chi2[k]=sum(((y.sppv-Expected.Y)^2)/Expected.Y,na.rm=TRUE)
    T.data.sppv.chi2[k]=sum(((Y-Expected.Y)^2)/Expected.Y,na.rm=TRUE)

}
(sppv=sum(T.mcmc.sppv.chi2>=T.data.sppv.chi2)/reps)

## [1] 0.05887
```

2.5 Plot MCMC output

```
par(mfrow=c(2,2))
plot(w1$out[[1]][ind,1],type='l',col=1,main="",
     xlab="MCMC iteration",ylab="p")
lines(w2$out[[1]][ind,1], col=3)
plot(density(w1$out[[1]][ind,1]),main="",xlab="p",
             ylab="Density")
plot(w1$out[[4]][ind],type='l',col=1,main="",
     xlab="MCMC iteration", ylab="N")
lines(w2$out[[4]][ind], col=3)
plot(density(w1$out[[4]][ind]),main="",
     xlab="N", ylab="Density")
```





2.6 Removing the site where the closure assumption was violated

```
rm(list=ls())

name="~/MCMCOutputModifiedlDataChain1.RData"

source(paste("~/Dropbox/MCMCAlgorithms/NMixtureModel/",
             "SpatialVaryingNMixtureModel/MCMCAlgorithm.R",
             sep=""))
```

```
## Load Data
load(paste("~/Dropbox/GitHub/HierarchicalGOF/",
             "HierarchicalGOF/data/SeaOtterData.RData",sep=""))
Y=SeaOtterData

## Remove the 19th row of data where sea otters
## were observed violating the closure assumption
Y=Y[-19,]

## Priors
q.p=1
r.p=1
alpha=0.001
beta=0.001
n.iter=10000000
thin=1
checkpoint=1000000
```

2.7 Run MCMC algorithm

Fit a simple N-mixture model with no covariates.

```
## Run algorithm
Nmixmcmc(Y=Y,q.p,r.p,alpha,beta,n.iter,checkpoint,name,thin)
```

2.8 Summarize results using partial data

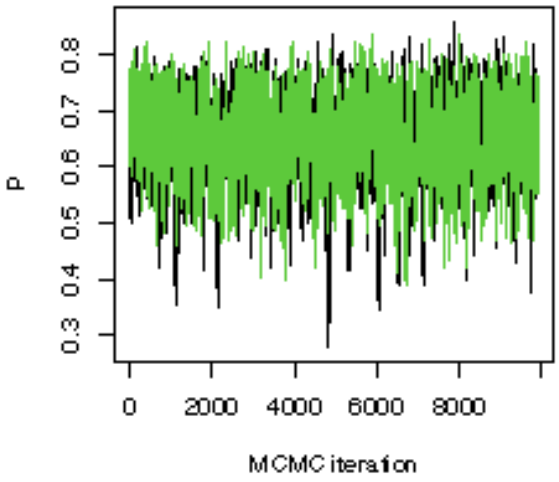
```
##
## Load output and calculate results
##

w1=new.env()
load(paste("~/MCMCOutputModifiedDataChain1.RData",
           sep=""),
     envir=w1)
w2=new.env()
load(paste("~/MCMCOutputModifiedDataChain2.RData",
           sep=""),
     envir=w2)
(status=sum(!is.na(w2$out[[1]][,1])))

## [1] 10000000
```

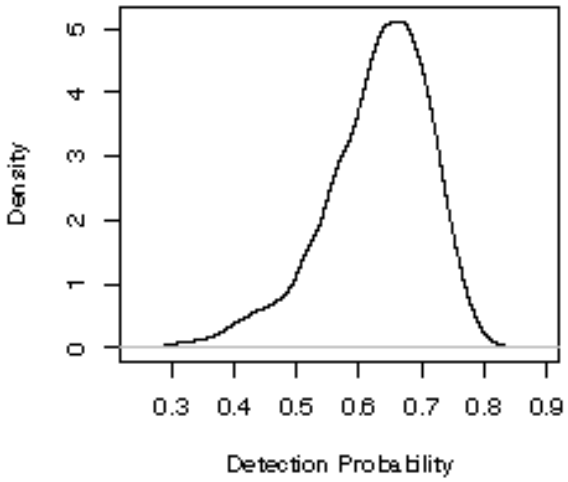
```
thin=1000    # large thinning value required due to autocorrelation
burn=100000
ind=seq(burn+1,status,thin)
length(ind)

## [1] 9900

N.tot1=w1$out[[4]][ind]
N.tot2=w2$out[[4]][ind]
Mean.N=mean(N.tot1)
LB=quantile(N.tot1,0.025)
UB=quantile(N.tot1,0.975)

##
## Gelman Rubin Diagnostic
##

mcmc1.tmp=cbind(w1$out[[1]][1:status,],w1$out[[3]][1:status,])
mcmc2.tmp=cbind(w2$out[[1]][1:status,],w2$out[[3]][1:status,])
chain1=mcmc(mcmc1.tmp)
chain2=mcmc(mcmc2.tmp)
out.list=mcmc.list(chain1,chain2)
(GR.Diag=gelman.diag(out.list,confidence = 0.95,
                     transform=FALSE,autoburnin=TRUE)[2])

## $mpsrf
## [1] 1.003851

##
## Effective sample size
##

(ESS=min(ess(chain1)))

## [1] 3465.526

##
## Bayesian p-value
##

T.mcmc.chi2=w2$out[[5]][ind]
T.data.chi2=w2$out[[6]][ind]
(Bayes.p=sum(T.mcmc.chi2>=T.data.chi2,na.rm=TRUE)/length(ind))

## [1] 0.5633333
```

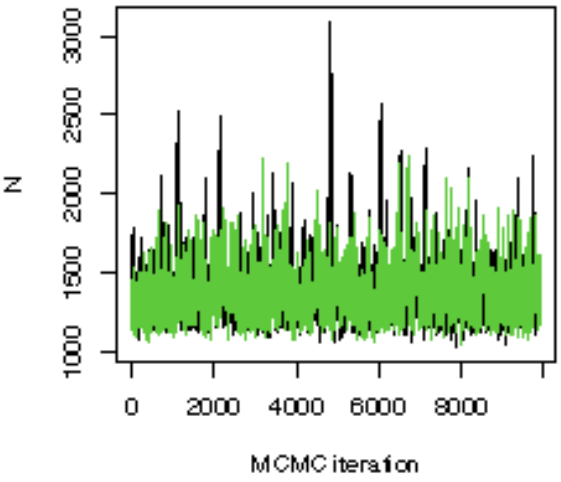
```
##
## Sampled posterior predictive value
##

set.seed(2017)
n=dim(Y)[1]
J=dim(Y)[2]
param.vec.id=sample(ind,1)
p.sppv=w1$out[[1]][param.vec.id]
lambda.sppv=w1$out[[3]][param.vec.id,]
N.sppv=w1$out[[2]][param.vec.id,]
Expected.Y=matrix(N.sppv*p.sppv,dim(Y)[1],dim(Y)[2])
reps=100000
T.mcmc.sppv.chi2=numeric(reps)
T.data.sppv.chi2=numeric(reps)
for(k in 1:reps){
    y.sppv=matrix(rbinom(n*J,N.sppv,p.sppv),n,J)
    y.sppv[is.na(Y)]=NA
    ## T.mcmc.sppv.chi2[k]=sum(apply(y.sppv,1,var,na.rm=TRUE))
    ## T.data.sppv.chi2[k]=sum(apply(Y,1,var,na.rm=TRUE))
    T.mcmc.sppv.chi2[k]=sum(((y.sppv-Expected.Y)^2)/Expected.Y,na.rm=TRUE)
    T.data.sppv.chi2[k]=sum(((Y-Expected.Y)^2)/Expected.Y,na.rm=TRUE)

}
(sppv=sum(T.mcmc.sppv.chi2>=T.data.sppv.chi2)/reps)

## [1] 0.82266
```

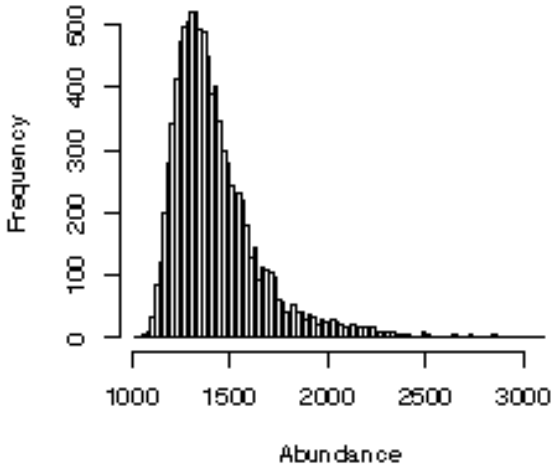
2.9 Plot MCMC output

```
par(mfrow=c(2,2))
plot(w1$out[[1]][ind,1],type='l',col=1,main="",
     xlab="MCMC iteration",ylab="p")
lines(w2$out[[1]][ind,1], col=3)
plot(density(w1$out[[1]][ind,1]),main="",
     xlab="Detection Probability")

plot(w1$out[[4]][ind],type='l',col=1,main="",
     xlab="MCMC iteration", ylab="N")
lines(w2$out[[4]][ind],col=3)
hist(w1$out[[4]][ind],main="",
             xlab="Abundance",breaks=100)
```


p
0.3
0.4
0.5
0.6
0.7
0.8
0
2000
4000
6000
8000
MCMC iteration


Density
0
1
2
3
4
5
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Detection Probability


N
1000
1500
2000
2500
3000
0
2000
4000
6000
8000
MCMC iteration


Frequency
0
100
200
300
400
500
1000
1500
2000
2500
3000
Abundance