Applied Bayesian Analysis : NCSU ST 540

Midterm2

Bruce Campbell

VAR(1) in JAGS

This section is a test section where we generate and fit a vector autoregressive model - $VAR(1) \in \mathbf{R}^6$ given by

$$y_t = \nu + \rho * y_{t-1} + \epsilon$$

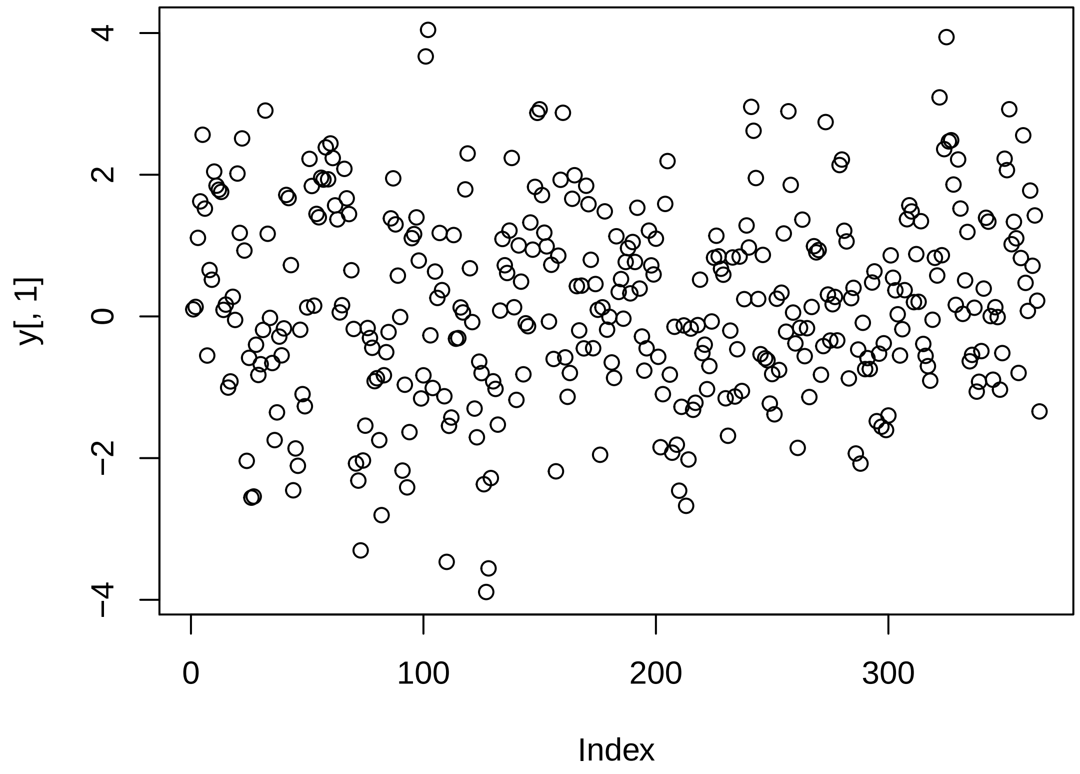
$$\epsilon \sim N(0, \Sigma)$$

We use the $y1$ data to calculate a NaN firendly sample covariance and then we find the nearest positive semidefinite matrix to use to generate data for the model.

Notes - imputation is not working for this model - this section seeks to find the parameters that best explain the data

```
library(rjags)
library(coda)
library(modeest)
library(MASS)
load("E2.RData")

DEBUG <- TRUE
if(DEBUG)
{
nSamples <- 5000
n.chains <- 1
} else
{
nSamples <- 40000
n.chains <- 8
}
cor.y1 <- cor(Y1, use = "pairwise.complete.obs")
cov.y1  <- cov(Y1, use = "pairwise.complete.obs")

N <- nrow(Y1)
p = 6
Y1.scaled <- scale(Y1)
# Try to simulate using the corr
```

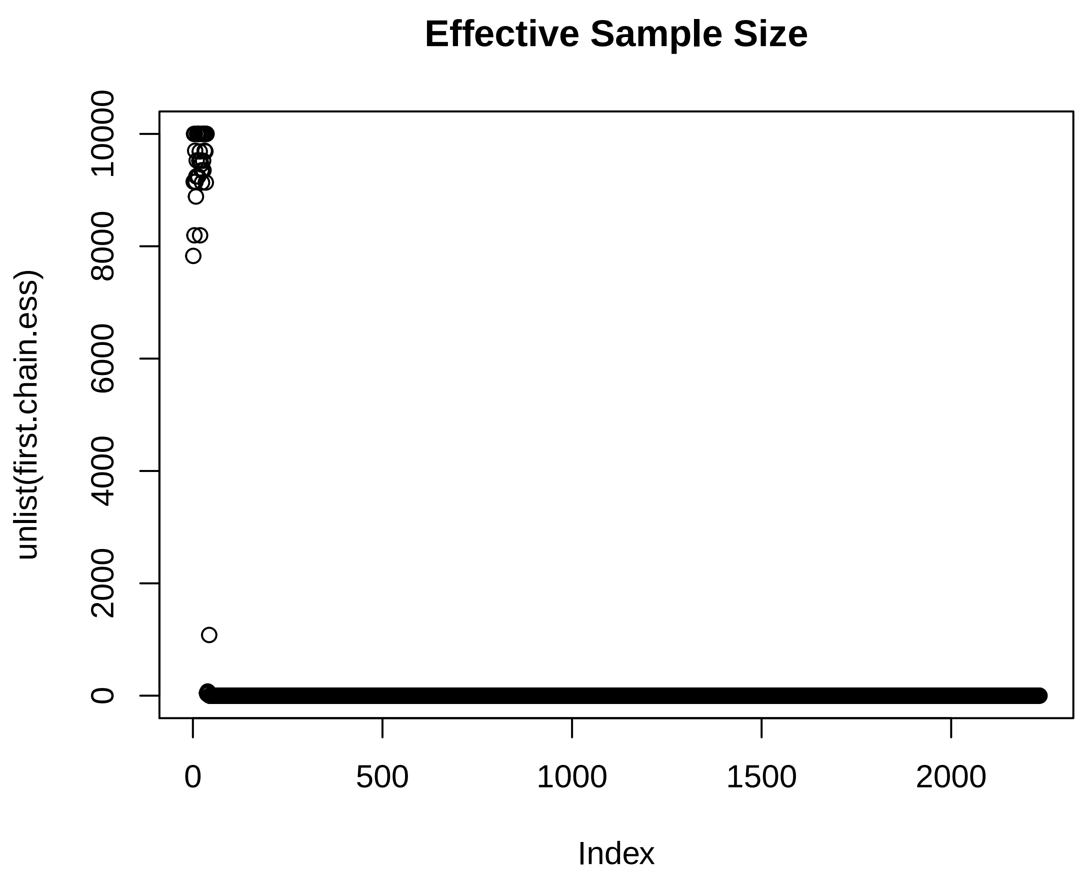
```
#install.packages("Matrix")
library("Matrix")

sig <- nearPD(cov.y1)
N = 365
Sigma = sig$mat
rho = .6
nu = matrix(rep(.1,6), p, 1)
y = matrix(NA, N,p)
y[1,] = nu
for(t in 2:N)
{
  y[t,] = mvrnorm(1, nu + rho * y[t-1,], Sigma)
}
plot(y[,1])
```



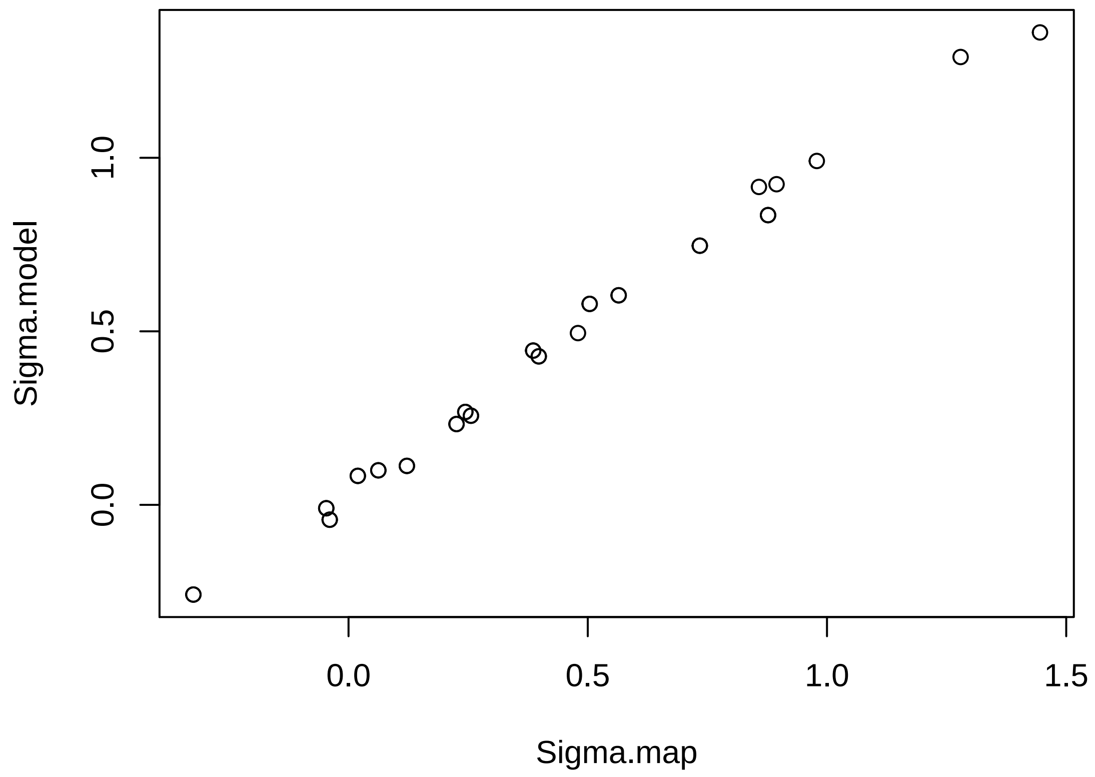
```
# Jags code to fit the model to the simulated data
model_code = '
model
{
  # Likelihood
  for (t in 2:N)
  {
    y[t, ] ~ dmnorm(mu[t, ], precisionAR)
    mu[t, 1:p] <- nu + rho * y[t-1,]
  }
  precisionAR ~ dwish(I, p+1)
  Sigma <- inverse(precisionAR)

  # Priors
  rho ~ dunif(-1, 1)
  for(i in 1:p)
  {
    nu[i] ~ dnorm(0, 0.01)
  }

  # Missing data model for y - not working in JAGS
  #for(i in 1:N)
  #{
    #y[i,1:p]~dmnorm(x_mn[],x_prec[,])
  #}

  # Priors for missing-data model parameters
  for(j in 1:p)
  {
    x_mn[j]~dnorm(0,0.01)
  }
  x_prec[1:p,1:p]~dwish(R[,],k)
  x_cov[1:p,1:p]<-inverse(x_prec[,])
  k <- p+0.1
  for(j1 in 1:p)
  {
    for(j2 in 1:p)
    {
      R[j1,j2] <- 0.1*equals(j1,j2)
    }
  }
}
'
# Set up the data
model_data = list(N = N, p = p, y = y, I = diag(p))
# Choose the parameters to watch
model_parameters =  c("nu", "rho", "Sigma","y")

model <- jags.model(textConnection(model_code),data = model_data,n.chains = n.chains)#Compile
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 364
##    Unobserved stochastic nodes: 15
##    Total graph size: 1207
##
## Initializing model
```

```r
update(model, nSamples, progress.bar="none"); # Burnin
out.coda  <- coda.samples(model, variable.names=model_parameters,n.iter=2*nSamples)

#plot(out.coda )

if(n.chains > 1)
{
  gelman.srf <-gelman.diag(out.coda)
  plot(gelman.srf$psrf,main = "Gelman Diagnostic")
}

chains.ess <- lapply(out.coda,effectiveSize)
first.chain.ess <- chains.ess[1]
plot(unlist(first.chain.ess), main="Effective Sample Size")
```



```
chain <- out.coda[[1]]
posterior.means <- list()
posterior.modes <- list()
for( i in 1:length(colnames(chain)) )
{
  colname <- colnames(chain)[i]
  samples <- chain[,i]
  posterior.means[colname] <-mean(samples)
  posterior.modes[colname] <-mlv(samples)$M
}

# plot(posterior.means,posterior.modes) # Nice and unimodal

Sigma.map <- unlist(posterior.means)[1:36]
Sigma.model<- unlist(sig$mat)
plot(Sigma.map,Sigma.model)
```



```
rho.map <- unlist(posterior.means)[43]

nu.map <-  unlist(posterior.means)[37:42]

# y.map <-  matrix(unlist(posterior.means)[44:2233],ncol=6, byrow=FALSE)
# plot(y[,1]-y.map[,1])
```

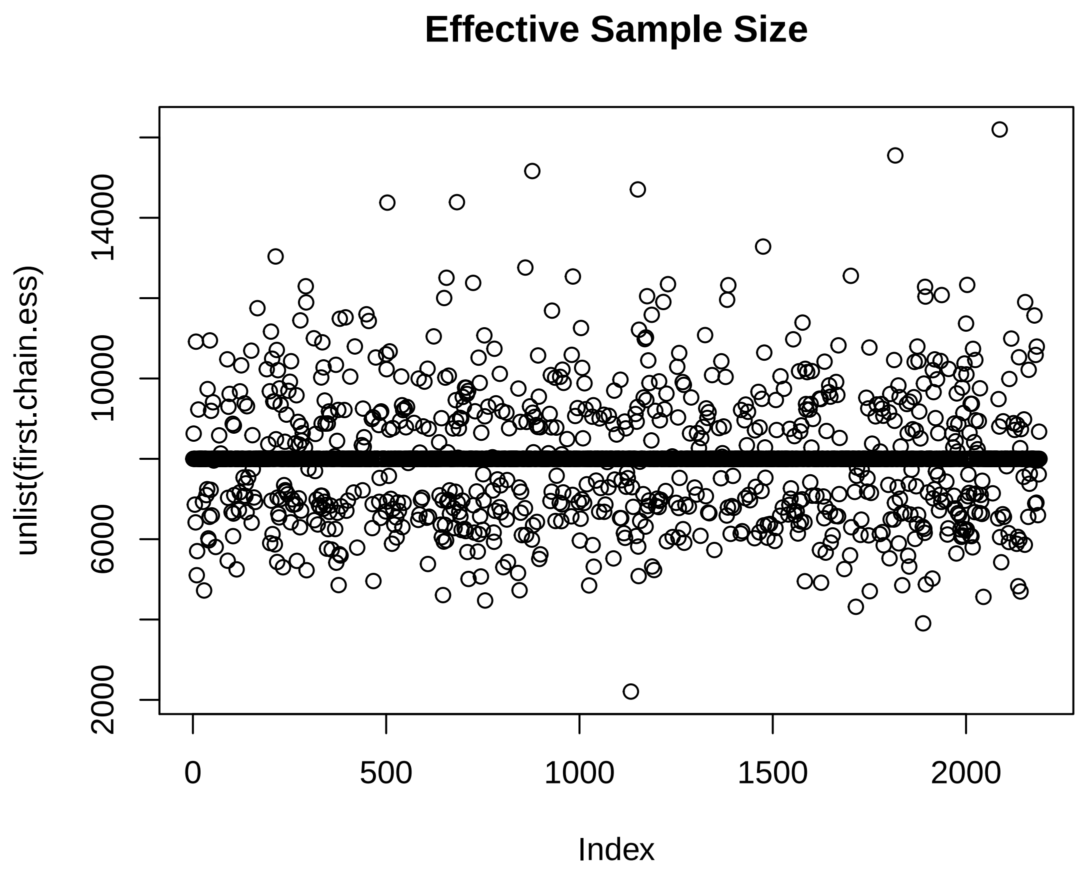
Missing data imputation in Openbugs

Here we try to implement missing data imputation in OpenBugs. We use the data set y1

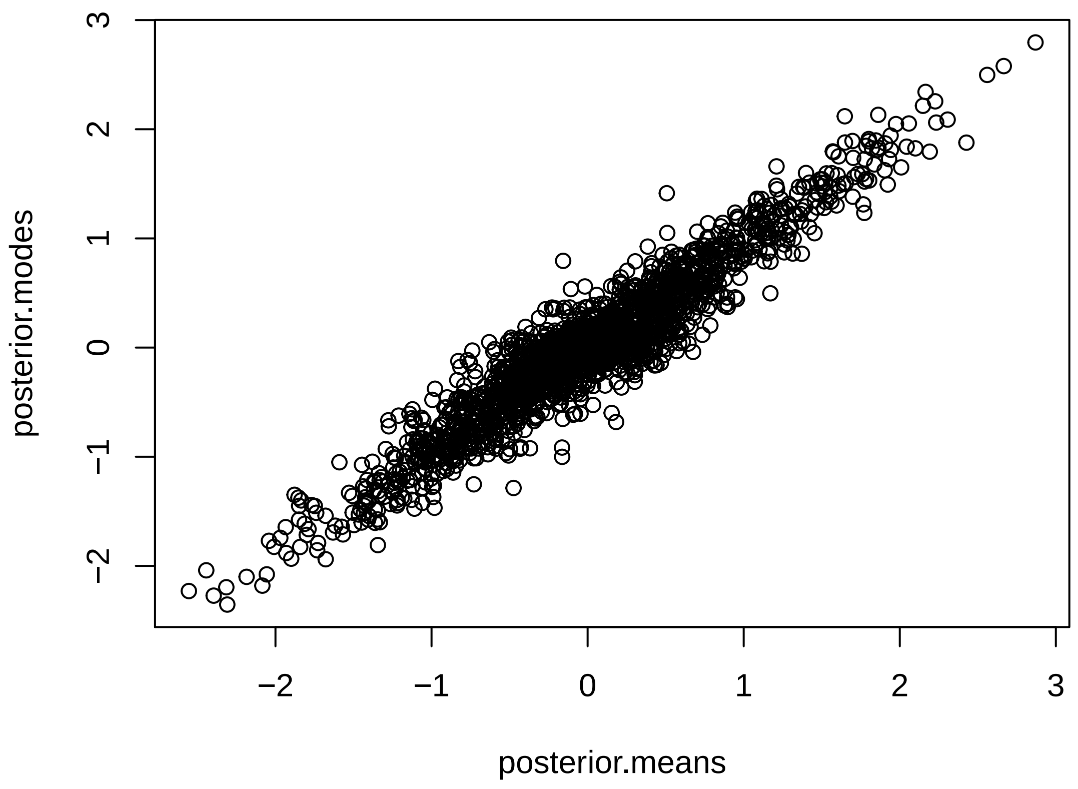
```
rm(list = ls())
setwd("d:/brucebcampbell-git/bayesian-learning-with-R")
load("E2.RData")
library(R2OpenBUGS)
library(rjags)
library(coda)
library(modeest)
N <- nrow(Y1)
```

```
p = 6
x <- scale(Y1)

mlr_model2 <- function() {
    for (i in 1:N) {
        theta[i, 1:p] ~ dmnorm(x[i, 1:p],
            precision2[, ])
    }

    # Prior for likelihood parameters: mu2,
    # precision2, rho
    rho ~ dunif(-1, 1)

    for (j in 1:p) {
        mu2[j] ~ dnorm(0, 0.01)
    }

    precision2[1:p, 1:p] ~ dwish(R[, ],
        k)

    # Missing data model for x
    for (i in 1:N) {
        x[i, 1:p] ~ dmnorm(x_mn[], x_prec[,
            ])
    }

    # Priors for missing-data model
    # parameters
    for (j in 1:p) {
        x_mn[j] ~ dnorm(0, 0.01)
    }
    x_prec[1:p, 1:p] ~ dwish(R[, ], k)
    x_cov[1:p, 1:p] <- inverse(x_prec[,
        ])

    k <- p + 0.1
    for (j1 in 1:p) {
        for (j2 in 1:p) {
            R[j1, j2] <- 0.1 * equals(j1,
                j2)
        }
    }
}

n.chains = 1
nSamples = 10000
stacks_dat <- list(x = x, p = 6, N = 365)
```

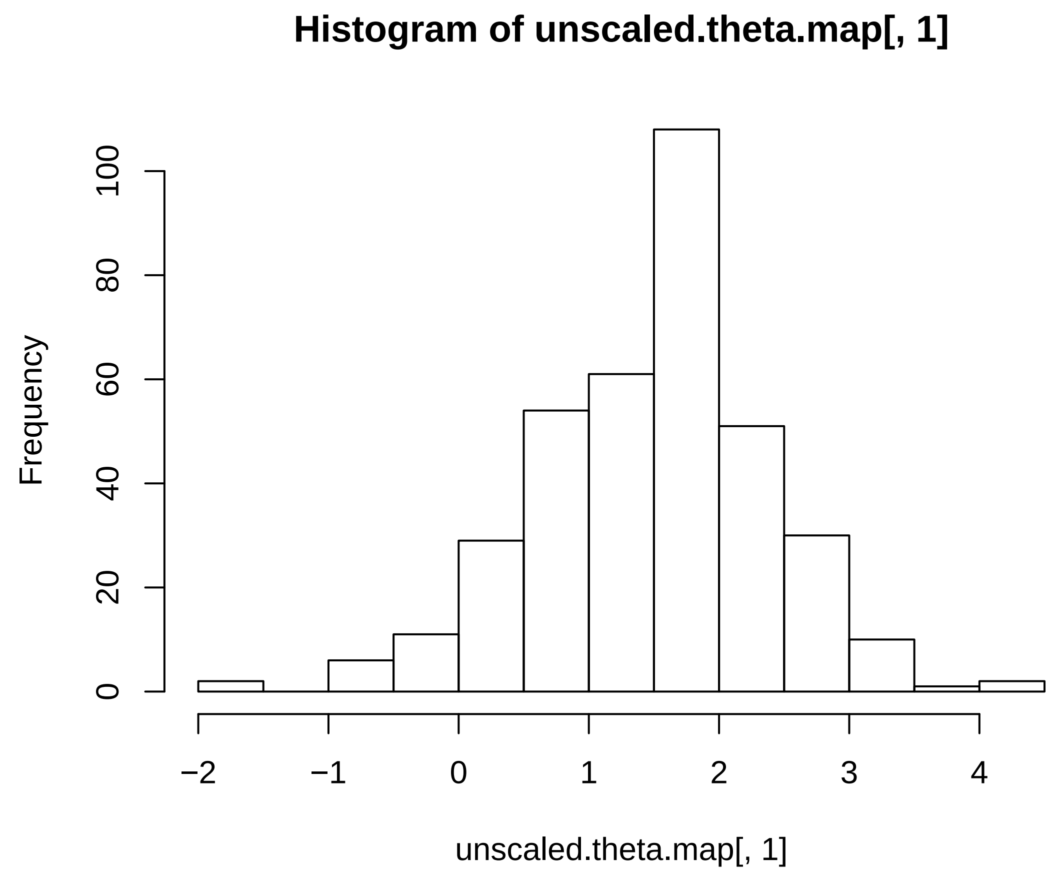
```r
mlr_inits <- function() {
    list(rho = 0)
}

samps <- bugs(data = stacks_dat, inits = mlr_inits,
    parameters.to.save = c("theta"), model.file = mlr_model2,
    codaPkg = TRUE, n.chains = n.chains,
    n.burnin = 2000, n.iter = nSamples,
    n.thin = 10, DIC = F)

out.coda <- read.bugs(samps)

if (n.chains > 1) {
    gelman.srf <- gelman.diag(out.coda)
    count.coeff.gt <- sum(gelman.srf$psrf >
        1.1)
    count.coeff.gt
}

chains.ess <- lapply(out.coda, effectiveSize)

first.chain.ess <- chains.ess[1]
plot(unlist(first.chain.ess), main = "Effective Sample Size")
```



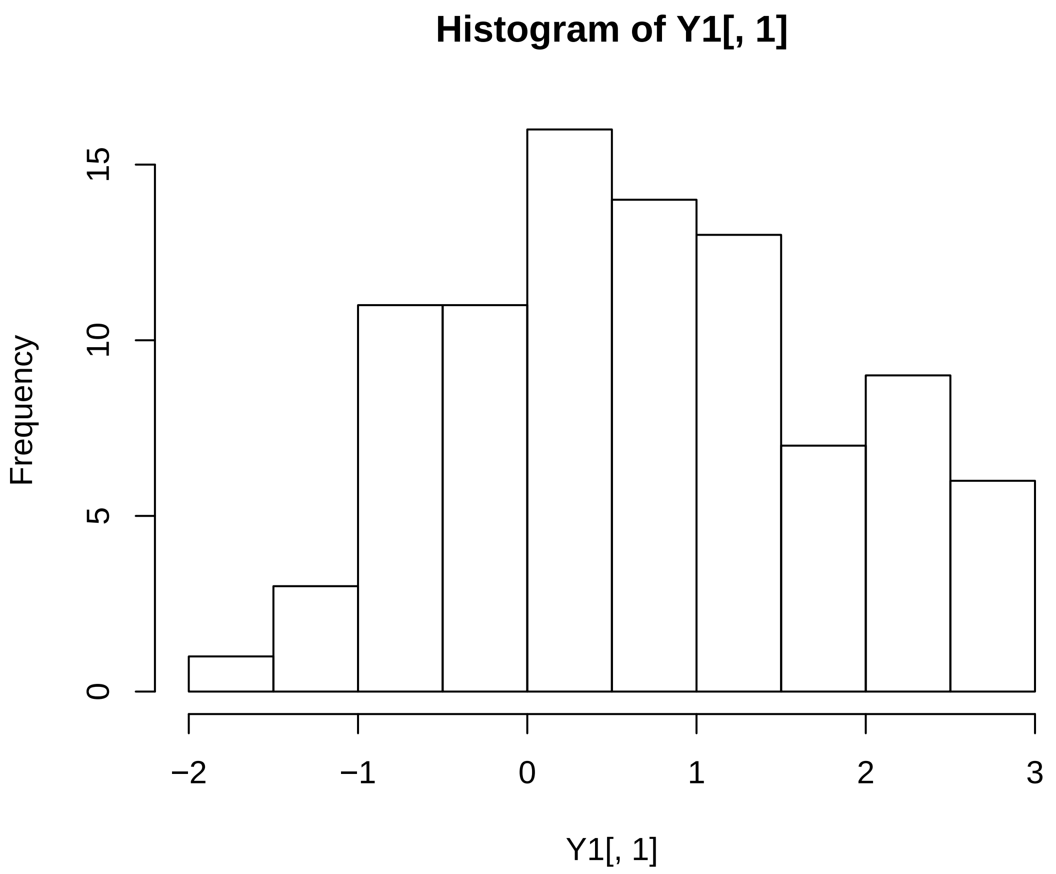
```r
chain <- out.coda[[1]]
posterior.means <- list()
posterior.modes <- list()

for (i in 1:(365 * 6)) {
    colname <- colnames(chain)[i]

    samples <- chain[, i]

    posterior.means[i] <- mean(samples)

    posterior.modes[i] <- mlv(samples)$M
}

plot(posterior.means, posterior.modes)
```



```r
theta.map <- matrix(unlist(posterior.means)[1:2190],
    ncol = 6, byrow = FALSE)

unscaled.theta.map <- (theta.map + colMeans(Y1,
    na.rm = TRUE)) * apply(Y1, 2, sd, na.rm = TRUE)

write.csv(unscaled.theta.map, file = "unscaled-theta-map.csv")
hist(unscaled.theta.map[, 1])
```


Histogram of unscaled.theta.map[, 1]
Frequency
0
20
40
60
80
100
−2
−1
0
1
2
3
4
unscaled.theta.map[, 1]

```r
hist(Y1[, 1])
```


Histogram of Y1[, 1]
Frequency
0
5
10
15
−2
−1
0
1
2
3
Y1[, 1]