

# Applied Bayesian Analysis : NCSU ST 540

Midterm2

*Bruce Campbell*

---

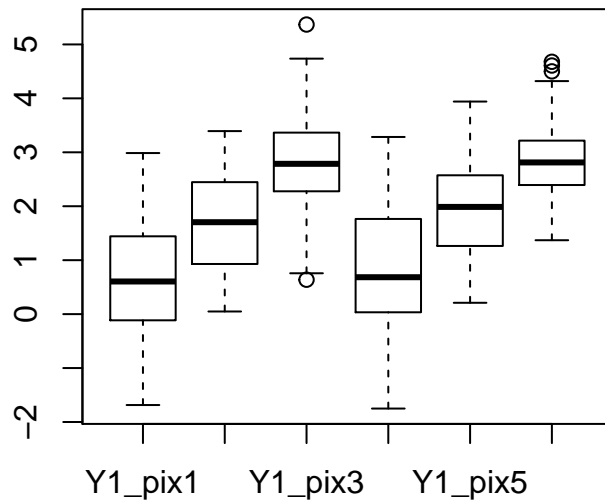
This section is a test section where we generate and fit a vector autoregressive model -  $VAR(1) \in \mathbf{R}^6$  given by

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

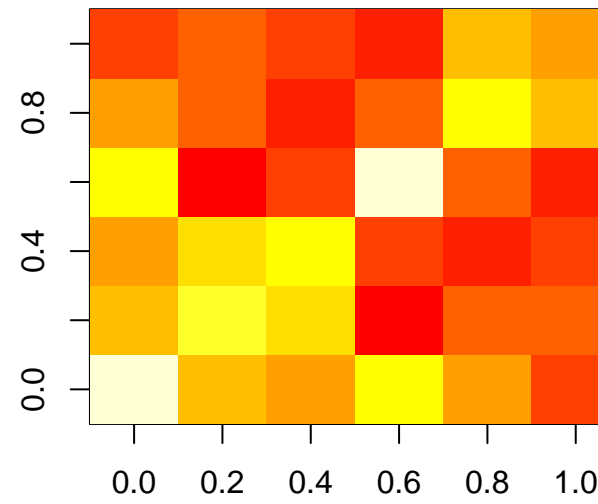
$$\epsilon \sim N(0, \Sigma)$$

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.

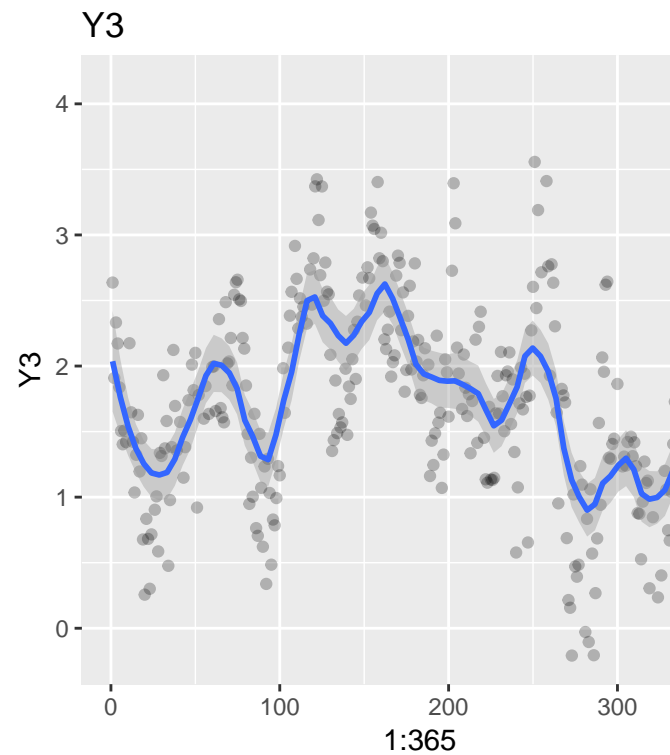
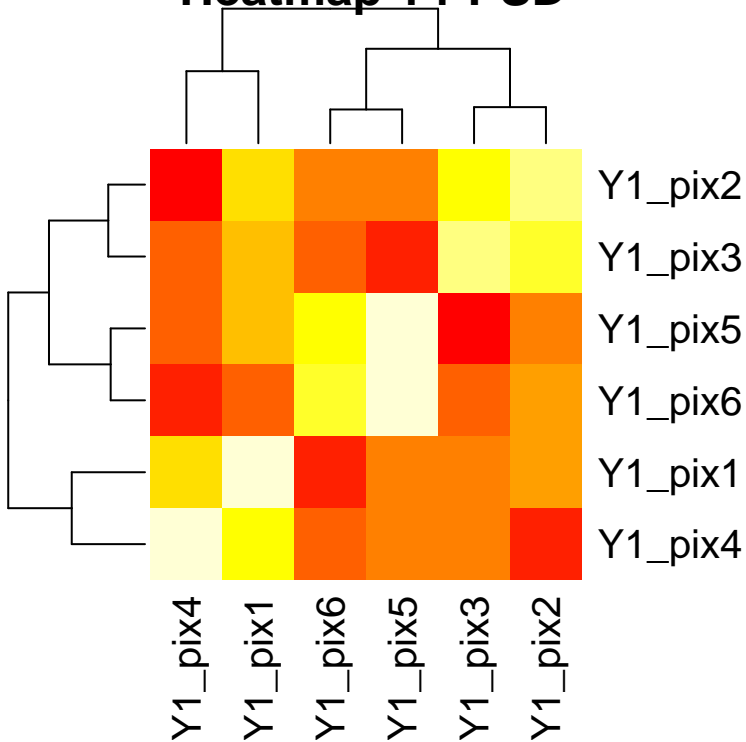
## Data Visualization

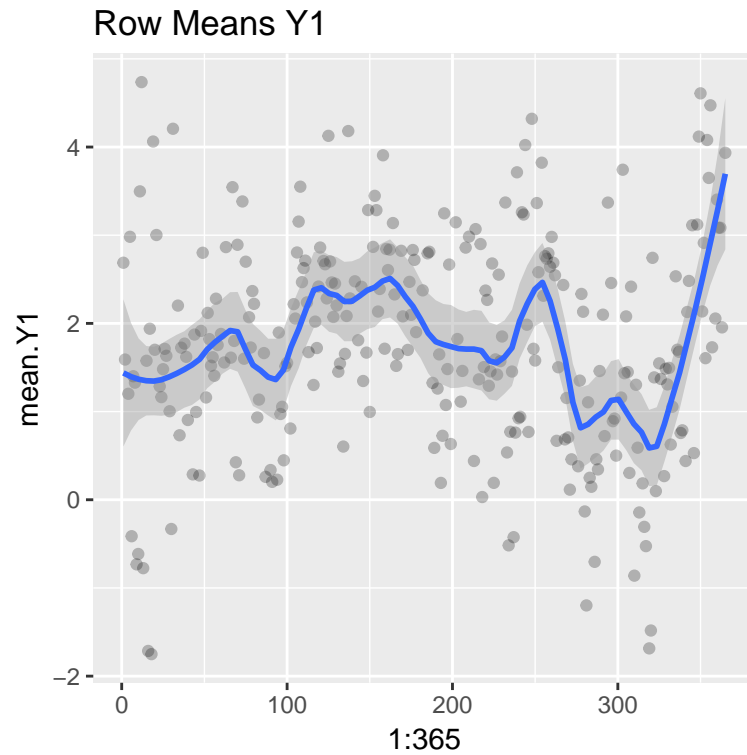


**Y1 : Nearst PSD to pairwise complete obs cov ma**



**Heatmap Y1 PSD**





### Test Data

```
rho = 0.9
nu = matrix((1:6) * 0.1, p, 1)
y = matrix(NA, N, p)
y[1, ] = (1:6)
for (t in 2:N) {
  y[t, ] = mvrnorm(1, nu + rho * y[t -
    1, ], empirical_sigma)
}
for (t in 2:N) {
  y[t, ] = mvrnorm(1, nu + rho * y[t -
    1, ], empirical_sigma)
}

Y1.test <- y
Y2.test <- y[, 1]
Y3.test <- rowMeans(y)
# redaction proportions
Y1.redaction.prop <- 0.8
Y2.redaction.prop <- 0.1
Y3.redaction.prop <- 0.1

Y1.test.redacted <- Y1.test
```

```

Y2.test.redacted <- Y2.test
Y3.test.redacted <- Y3.test

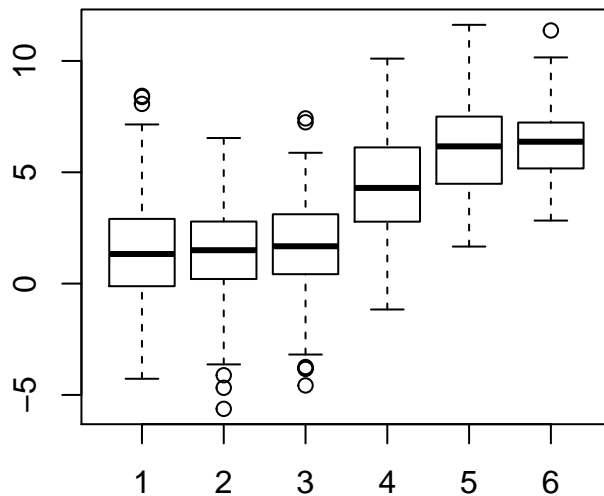
Y1.test.redacted.unlisted <- unlist(Y1.test.redacted)
Y1.NA.index <- sample(length(Y1.test.redacted),
  floor(length(Y1.test.redacted) * Y1.redaction.prop))
Y1.test.redacted.unlisted[Y1.NA.index] <- NA
Y1.test.redacted <- matrix(Y1.test.redacted.unlisted,
  ncol = 6, byrow = FALSE)

Y2.NA.index <- sample(length(Y2.test.redacted),
  floor(length(Y2.test.redacted) * Y2.redaction.prop))
Y2.test.redacted[Y2.NA.index] <- NA

Y3.NA.index <- sample(length(Y3.test.redacted),
  floor(length(Y3.test.redacted) * Y3.redaction.prop))
Y3.test.redacted[Y3.NA.index] <- NA

boxplot(Y1.test)

```

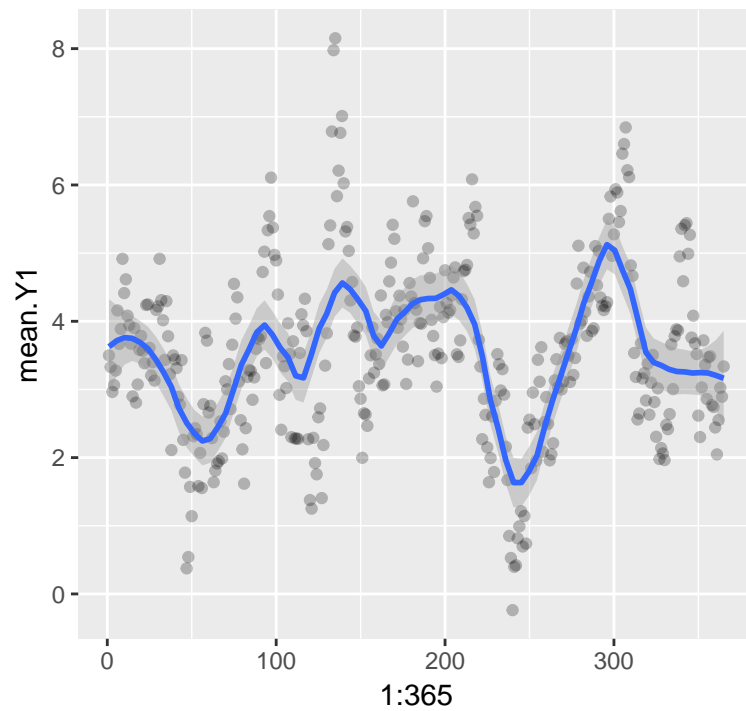


```

ggplot(data.frame(mean.Y1 = rowMeans(Y1.test,
  na.rm = TRUE)), aes(x = 1:365, y = mean.Y1)) +
  geom_point(alpha = 0.25) + geom_smooth(method = "loess",
  span = 0.22) + ggtitle("TEST DATA : Row Means Y1.test")

```

TEST DATA : Row Means Y1.test



```
library(R2OpenBUGS)
n.chains = 2
n.thin = 2
n.burnin = 5000
n.samples = 20000

x <- as.matrix(Y1)

stacks_dat <- list(Y1 = Y1, Y2 = Y2, Y3 = Y3,
  p = 6, N = 365)

mlr_model <- function() {

  for (i in 1:N) {
    # x[i,1] ~ dnorm(theta[i,1], inv.var)
    Y2[i] ~ dnorm(theta[i, 1], inv.var)

    Y3[i] ~ dnorm(thetaBar[i], inv.var)
    thetaBar[i] <- 1/6 * (theta[i,
      1] + theta[i, 2] + theta[i,
      3] + theta[i, 4] + theta[i,
      5] + theta[i, 6])
  }

  inv.var ~ dgamma(0.01, 0.01)
  for (i in 1:N) {
```

```

    for (j in 1:p) {
      theta[i, j] ~ dnorm(Y1[i, 1],
        inv.var)
    }
  }

  # Missing data model for x
  for (i in 1:N) {
    Y1[i, 1:p] ~ dmnorm(Y1_mn[], Y1_prec[,
      ])
  }

  # Priors for missing-data model
  # parameters
  for (j in 1:p) {
    Y1_mn[j] ~ dnorm(0, 0.01)
  }
  Y1_prec[1:p, 1:p] ~ dwish(R[, ], k)
  Y1_cov[1:p, 1:p] <- inverse(Y1_prec[,
    ])

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

mlr_inits <- function() {
  list(I = diag(p), tau = 0.01)
}

samps <- bugs(data = stacks_dat, inits = mlr_inits,
  parameters.to.save = c("theta"), model.file = mlr_model,
  codaPkg = TRUE, debug = FALSE, n.chains = n.chains,
  n.burnin = n.burnin, n.iter = n.samples,
  n.thin = n.thin, DIC = F)

out.coda <- read.bugs(samps)

## Abstracting theta[1,1] ... 15000 valid values
## Abstracting theta[1,2] ... 15000 valid values
## Abstracting theta[1,3] ... 15000 valid values
## Abstracting theta[1,4] ... 15000 valid values
## Abstracting theta[1,5] ... 15000 valid values
## Abstracting theta[1,6] ... 15000 valid values

```

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

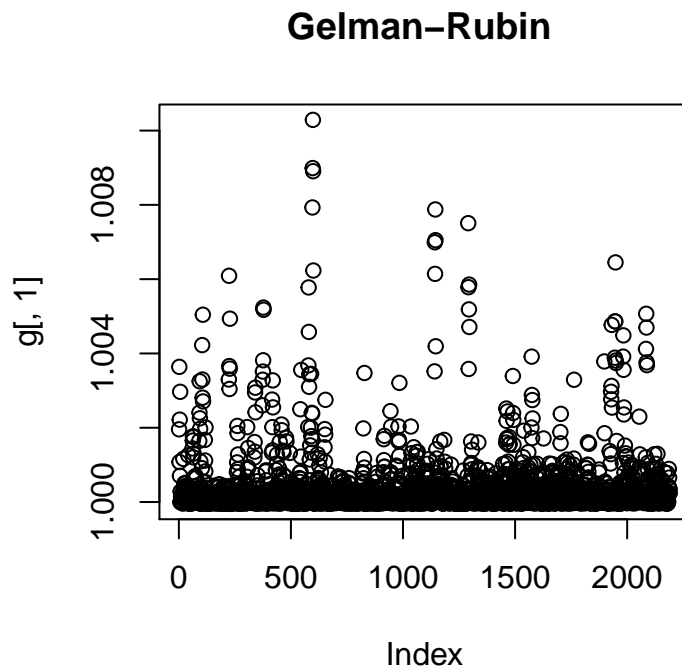


[illegible]

```
## Abstracting theta[365,1] ... 15000 valid values
## Abstracting theta[365,2] ... 15000 valid values
## Abstracting theta[365,3] ... 15000 valid values
## Abstracting theta[365,4] ... 15000 valid values
## Abstracting theta[365,5] ... 15000 valid values
## Abstracting theta[365,6] ... 15000 valid values

# save(out.coda,file =
# 'out.coda_OPENBUGS_Imputation.RData')

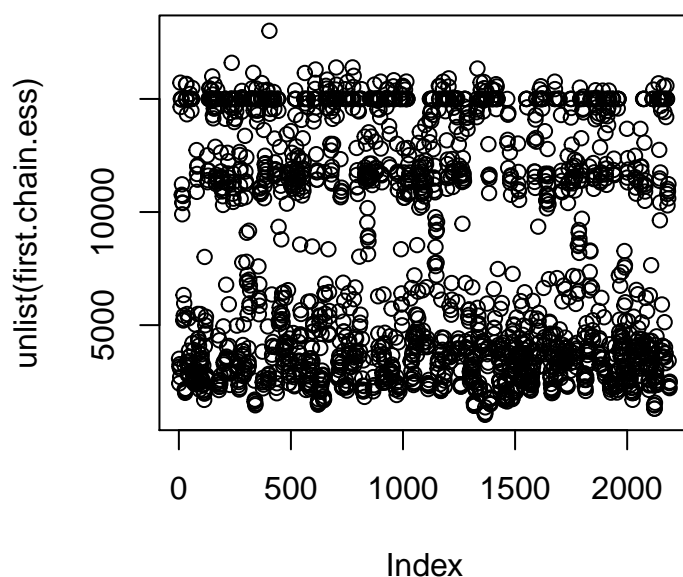
if (n.chains > 1) {
  g <- matrix(NA, nrow = nvar(out.coda),
    ncol = 2)
  for (v in 1:nvar(out.coda)) {
    g[v, ] <- gelman.diag(out.coda[,
      v])$psrf
  }
  count.coeff.gt <- sum(g[, 1] > 1.1)
  count.coeff.gt
  plot(g[, 1], main = "Gelman-Rubin ")
}
```



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

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

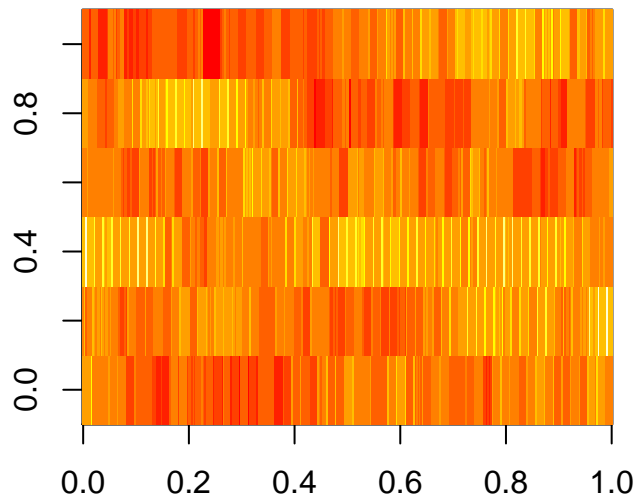
## Effective Sample Size



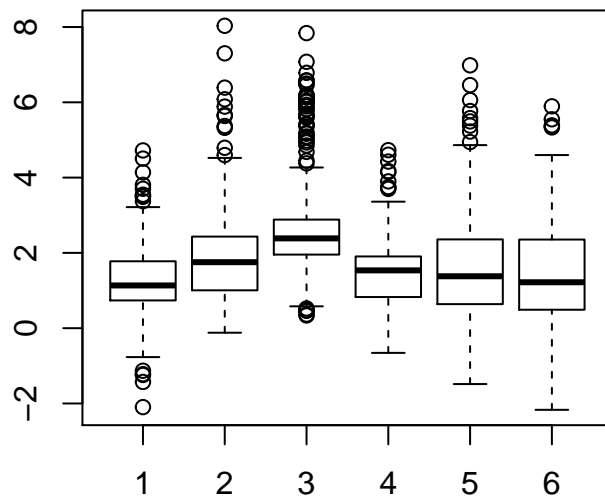
```
chain <- out.coda[[1]]
# imputedY1 <- Y1 for( i in
# 1:ncol(chain) ) { colname <-
# colnames(chain)[i] idx
# <-gsub('x','',colname) idx
# <-gsub('\\[','',idx)
# idx<-gsub('\\]','',idx)
# strsplit(idx,',') idi <-
# as.numeric(strsplit(idx,',')[[1]][1])
# idj <-
# as.numeric(strsplit(idx,',')[[1]][2])
# if(grepl('Sigma',colname) ) {
# print(paste('skipping
# ',colname,sep='')) next } samples <-
# chain[,i] imputedY1[idi,idj]
# <-mlv(samples)$M }

theta.posterior.modes <- list()
for (i in 1:ncol(chain)) {
  colname <- colnames(chain)[i]
  if (grepl("theta", colname)) {
    samples <- chain[, i]
    theta.posterior.modes[colname] <- mlv(samples)$M
  } else {
    next
  }
}
```

```
    }  
  }  
  theta.posterior <- matrix(unlist(theta.posterior.modes),  
    ncol = 6, byrow = FALSE)  
  image(theta.posterior)
```



```
boxplot(theta.posterior)
```



```
write.csv(theta.posterior, file = "theta.posterior.csv")
```

```
# image(Y1, main='Y1')
```

```
# image(imputedY1, main='Imputed')
```

```
# write.csv(unscaled.theta.map, file =  
# 'unscaled-theta-map.csv')
```