STAT 651 Final Project Code Supplement

Jacob Andros

12/7/2021

Libraries

```r
library(R2jags)
library(coda)
library(truncnorm)
```

Data setup

```r
# Read in data for each inning
cols <- c(1,3:5)
inn1 <- read.csv("data/inning1.csv", sep=",")[,cols]
inn2 <- read.csv("data/inning2.csv", sep=",")[,cols]
inn3 <- read.csv("data/inning3.csv", sep=",")[,cols]
inn4 <- read.csv("data/inning4.csv", sep=",")[,cols]
inn5 <- read.csv("data/inning5.csv", sep=",")[,cols]
inn6 <- read.csv("data/inning6.csv", sep=",")[,cols]
inn7 <- read.csv("data/inning7.csv", sep=",")[,cols]
inn8 <- read.csv("data/inning8.csv", sep=",")[,cols]
inn9 <- read.csv("data/inning9.csv", sep=",")[,cols]

# Combine for one dataset
df <- rbind(inn1, inn2, inn3, inn4, inn5, inn6, inn7, inn8, inn9)
colnames(df) <- c("pitches", "name", "date", "total")
df$inning <- c(
  rep(1, nrow(inn1)),
  rep(2, nrow(inn2)),
  rep(3, nrow(inn3)),
  rep(4, nrow(inn4)),
  rep(5, nrow(inn5)),
  rep(6, nrow(inn6)),
  rep(7, nrow(inn7)),
  rep(8, nrow(inn8)),
  rep(9, nrow(inn9))
)

# cleanup
rm(inn1, inn2, inn3, inn4, inn5, inn6, inn7, inn8, inn9)
df <- df[order(df$name, df$date, df$inning),]
rownames(df) <- 1:nrow(df)

# Remove the final inning of the pitcher's start from each game (usually is not a full inning)
```

```r
ind <- which(df$inning == 1)[-1] - 1
df <- df[-ind,]

# Divide up data by pitcher
y1 <- df$pitches[df$name=="Buehler, Walker"]
y2 <- df$pitches[df$name=="Eovaldi, Nathan"]
y3 <- df$pitches[df$name=="Wheeler, Zack"]
y4 <- df$pitches[df$name=="Walker, Taijuan"]
y5 <- df$pitches[df$name=="McClanahan, Shane"]
y6 <- df$pitches[df$name=="Fried, Max"]
y <- list(y1,y2,y3,y4,y5,y6)
rm(y1,y2,y3,y4,y5,y6)
N <- unlist(lapply(y, length))
sumY <- unlist(lapply(y, sum))
```

EDA

```r
unlist(lapply(y, mean))
```

Helper functions

```r
# Helper function for Monte Carlo error
mcErr <- function(draws) {
  m <- mean(draws)
  ess <- coda::effectiveSize(draws)
  m + c(-1,1) * qnorm(.975) * sd(draws)/ess
}

# Helper function for summarizing any set of draws
summ <- function(draws) {
  draws <- unname(draws)
  med <- apply(draws, 2, median)
  credLwr <- apply(draws, 2, quantile, .025)
  credUpr <- apply(draws, 2, quantile, .975)
  mc <- t(apply(draws, 2, mcErr))
  mcLwr <- mc[,1]
  mcUpr <- mc[,2]
  df <- data.frame(med, credLwr, credUpr, mcLwr, mcUpr)
  colnames(df) <- c(
    "Median",
    "Credible Lower",
    "Credible Upper",
    "MC Error Lower",
    "MC Error Upper"
  )
  rownames(df) <- c(
    "a1", "a2", "a3", "a4", "a5", "a6",
    "b1", "b2", "b3", "b4", "b5", "b6",
    "theta1", "theta2", "theta3", "theta4", "theta5", "theta6"
  )
  df
}
```

Set up functions for MCMC sampling

```r
logPriorA <- function(a) sum(dnorm(a, 16, sqrt(3), log=TRUE))
logPriorB <- function(b) sum(dnorm(b, 1, .25, log=TRUE))
logLikTheta <- function(theta) sum(dgamma(theta, a, b, log=TRUE))
logPrior <- function(a, b, theta) logPriorA(a) + logPriorB(b) + logLikTheta(theta)
logLik <- function(theta) sum(sapply(1:6, function(i) sum(dpois(y[[i]], theta[i], log=TRUE))))
logPost <- function(a, b, theta) logPrior(a, b, theta) + logLik(theta)
logPostAB <- function(a, b) logPrior(a, b, theta) + logLik(theta)
```

My MCMC

```r
nRun <- 1000000
nBurn <- 10000
nThin <- 10
n <- nBurn+nRun
thetaDraws <- matrix(0, nrow=n/nThin, ncol=6)
aDraws <- matrix(0, nrow=n/nThin, ncol=6)
bDraws <- matrix(0, nrow=n/nThin, ncol=6)
aDraws[1,] <- a <- rnorm(6, 16, sqrt(3))
bDraws[1,] <- b <- rnorm(6, 1, 0.25)
thetaDraws[1,] <- theta <- rgamma(6, a, b)

accA <- accB <- 0
tol <- .0001
kA <- 1.5
kB <- 0.4

s1 <- Sys.time()
for (i in 2:n) {

  # Update alpha (a)
  propA <- runif(6, a - kA, a + kA)
  r <- logPostAB(propA, b) - logPostAB(a, b)
  if (log(runif(1)) < r) {
    a <- propA
    accA <- accA + 1
  }

  # Update beta (b)
  lbB <- b - kB
  lbB[lbB< tol] <- tol # Keeps the proposal values above zero
  propB <- runif(6, lbB, b + kB)
  r <- logPostAB(a, propB) - logPostAB(a, b)
  if (log(runif(1)) < r) {
    b <- propB
    accB <- accB + 1
  }

  # Update theta
  theta <- rgamma(6, a+sumY, b+N)
  # Storage and thinning
  if (i %% nThin == 0) {
    j <- i/nThin
    thetaDraws[j,] <- theta
    aDraws[j,] <- a
    bDraws[j,] <- b
  }

}
timeMCMC <- Sys.time() - s1

# Burn in
burn <- (nBurn/nThin+1):(n/nThin)
aDraws <- aDraws[burn,]
bDraws <- bDraws[burn,]
thetaDraws <- thetaDraws[burn,]
draws <- cbind(aDraws, bDraws, thetaDraws)
```

My MCMC: Acceptance

```
c(accA,accB)/n
```

My MCMC: Effective SS

```
effectiveSize(thetaDraws)
effectiveSize(aDraws)
effectiveSize(bDraws)
```

My MCMC: Geweke Diags

```
# Geweke diagnostic
# (R Hat not applicable because there is only one chain for each parameter)
# No p-values below .09 for Geweke
pnorm(abs(geweke.diag(mcmc(thetaDraws))$z),lower.tail=FALSE)*2
pnorm(abs(geweke.diag(mcmc(aDraws))$z),lower.tail=FALSE)*2
pnorm(abs(geweke.diag(mcmc(bDraws))$z),lower.tail=FALSE)*2
```

My MCMC: Trace Plots

```
plot(aDraws[,1], type='l')
```

```
plot(bDraws[,1], type='l')
```

```
plot(thetaDraws[,1], type='l')
```

JAGS

Model code

```
model <- "
  model {
  for (i in 1:6) {
    for (j in ind[i]:(ind[i+1] - 1)) {
      yL[j] ~ dpois(theta[i])
    }
  }

  for (i in 1:6) {
    a[i] ~ dnorm(16, 1/3)
    b[i] ~ dnorm(1, 16)
    theta[i] ~ dgamma(a[i], b[i])
  }

}"
writeLines(model, "jags/baseball.txt")
```

Run JAGS and collect the samples

```
# Run JAGS
nIter <- 110000
nBurn <- 10000
nCores <- parallel::detectCores()
yL <- unlist(y)
ind <- c(1, cumsum(N)+1)
data.jags <-c('yL', 'ind')
parms <- c('theta', 'a', 'b')
s2 <- Sys.time()
mod.sim <- jags(data=data.jags,inits=NULL,
                parameters.to.save=parms,
                model.file='jags/baseball.txt',n.iter=nIter,
                n.burnin=nBurn,n.chains=nCores,n.thin=5)
timeJAGS <- Sys.time() - s2

# Collect posterior samples
sims <- as.mcmc(mod.sim)
chains <- unname(as.matrix(sims))
aDrawsJags <- chains[,1:6]
bDrawsJags <- chains[,7:12]
thetaDrawsJags <- chains[,14:19]
```

JAGS: Effective SS

```
effectiveSize(aDrawsJags)
effectiveSize(bDrawsJags)
effectiveSize(thetaDrawsJags)
```

JAGS: Geweke and Gelman Diags

```
# Geweke and RHat diagnostics
gelman.diag(sims) # All RHat equal to 1
# No p-values below .09 for Geweke
```

```
pnorm(abs(geweke.diag(mcmc(thetaDrawsJags))$z),lower.tail=FALSE)*2
pnorm(abs(geweke.diag(mcmc(aDrawsJags))$z),lower.tail=FALSE)*2
pnorm(abs(geweke.diag(mcmc(bDrawsJags))$z),lower.tail=FALSE)*2
```

Compare my MCMC with JAGS

Helper function

```
comp <- function(draws, drawsJags) {
  med <- cbind(apply(draws, 2, median), apply(drawsJags, 2, median))
  credLwr <- cbind(apply(draws, 2, quantile, .025), apply(drawsJags, 2, quantile, .025))
  credUpr <- cbind(apply(draws, 2, quantile, .975), apply(drawsJags, 2, quantile, .975))
  mc <- cbind(t(apply(draws, 2, mcErr)), t(apply(drawsJags, 2, mcErr)))
  mcLwr <- mc[,c(1,3)]
  mcUpr <- mc[,c(2,4)]
  df <- data.frame(med, credLwr, credUpr, mcLwr, mcUpr)
  colnames(df) <- c(
    "Median MCMC", "Median JAGS",
    "Credible Lower MCMC", "Credible Lower JAGS",
    "Credible Upper MCMC", "Credible Upper Jags",
    "MC Error Lower MCMC", "MC Error Lower JAGS",
    "MC Error Upper MCMC", "MC Error Upper JAGS"
  )
  df
}
```

Comparisons

```
aComp <- comp(aDraws, aDrawsJags)
bComp <- comp(bDraws, bDrawsJags)
thetaComp <- comp(thetaDraws, thetaDrawsJags)

aComp
bComp
thetaComp
```

Posterior predictive sampling

For the original six pitchers

```
preds <- t(apply(thetaDrawsJags, 1, function(x) {
  rpois(6, x)
}))
```

For a new pitcher (Adam Wainwright)

```
source("data/wainwright.R")
model <- "
  model {
```

```
  for (i in 1:N) {
    yW[i] ~ dpois(theta)
  }

  a ~ dnorm(16, 1/3)
  b ~ dnorm(1, 16)
  theta ~ dgamma(a, b)
}"
writeLines(model, "jags/wainwright.txt")

nIter <- 110000
nBurn <- 10000
nCores <- parallel::detectCores()
yW <- ww$pitches; N <- length(yW)
data.jags <-c('yW', 'N')
parms <- c('theta', 'a', 'b')
mod.sim <- jags(data=data.jags,inits=NULL,
                parameters.to.save=parms,
                model.file='jags/wainwright.txt',n.iter=nIter,
                n.burnin=nBurn,n.chains=nCores,n.thin=5)
simsWW <- as.mcmc(mod.sim)
chainsWW <- unname(as.matrix(simsWW))
aDrawsWW <- chainsWW[,1]
bDrawsWW <- chainsWW[,2]
thetaDrawsWW <- chainsWW[,4]

predsWW <- sapply(thetaDrawsWW, function(x) {
  rpois(1, x)
})
```

Posterior predictive checks on replicated data (using mean-squared error as the test quantity)

```
nSim <- nrow(thetaDrawsJags)
pVals <- matrix(0, nrow=nSim, ncol=6)
for (i in 1:6) {
  theta <- median(thetaDrawsJags[,i])
  yT <- y[[i]]
  nT <- length(yT)
  yRep <- t(sapply(1:nSim, function(j) rpois(1, thetaDrawsJags[j,i] )))
  observedT <- sapply(1:nSim, function(j) mean((yT - thetaDrawsJags[j,i])^2))
  predT <- sapply(1:nSim, function(j) mean((yRep[,j] - thetaDrawsJags[j,i])^2))
  pVals[,i] <- (predT >= observedT)
}
apply(pVals, 2, mean)
# No p-values below 0.11, so no red flags here
```

Sensitivity Analysis

Write alternative models

```r
# Alt 1
model <- "
  model {
  for (i in 1:6) {
    for (j in ind[i]:(ind[i+1] - 1)) {
      yL[j] ~ dpois(theta[i])
    }
  }

  for (i in 1:6) {
    a[i] ~ dnorm(16, 1)
    b[i] ~ dnorm(1, 100)
    theta[i] ~ dgamma(a[i], b[i])
  }

}"
writeLines(model, "jags/alt1.txt")

# Alt 2
model <- "
  model {
  for (i in 1:6) {
    for (j in ind[i]:(ind[i+1] - 1)) {
      yL[j] ~ dpois(theta[i])
    }
  }

  for (i in 1:6) {
    a[i] ~ dnorm(32, 1/3)
    b[i] ~ dnorm(2, 16)
    theta[i] ~ dgamma(a[i], b[i])
  }

}"
writeLines(model, "jags/alt2.txt")

# Alt 3
model <- "
  model {
  for (i in 1:6) {
    for (j in ind[i]:(ind[i+1] - 1)) {
      yL[j] ~ dpois(theta[i])
    }
  }

  for (i in 1:6) {
    a[i] ~ dunif(14, 17)
    b[i] ~ dunif(0.1, 1)
    theta[i] ~ dnorm(a[i], 1/b[i])
  }

}"
writeLines(model, "jags/alt3.txt")
```

Run alternative models

```r
altJags <- function(model.file, n=100000, nBurn=10000, nThin=5) {
  nIter <- nBurn+n
  nCores <- parallel::detectCores()
  data.jags <-c('yL', 'ind')
  parms <- c('theta', 'a', 'b')
  mod.sim <- jags(data=data.jags,inits=NULL,
                  parameters.to.save=parms,
                  model.file=model.file,n.iter=nIter,
                  n.burnin=nBurn,n.chains=nCores,n.thin=nThin)
  sims <- as.mcmc(mod.sim)
  chains <- unname(as.matrix(sims))
  chains[,-13] # Remove deviance
}

alt1 <- altJags("jags/alt1.txt")
alt2 <- altJags("jags/alt2.txt")
alt3 <- altJags("jags/alt3.txt")
```

Summarize alternative models using helper function from earlier

```r
summ(alt1)
summ(alt2)
summ(alt3)
```

Overall summary

```r
allDraws <- list(draws, chains[,-13], alt1, alt2, alt3)
summary <- lapply(allDraws, summ)
names(summary) <- c("MCMC Original", "JAGS Original", "Alt Prior 1", "Alt Prior 2", "Alt Prior 3")
summary
```

Frequentist analysis

```r
freqCI <- function(y, a=.05) {
  y0 <- sum(y)
  n <- length(y)
  lwr <- 1/(2*n) * qchisq(a/2, 2*y0)
  upr <- 1/(2*n) * qchisq(1-a/2, 2*(y0+1))
  c(lwr,upr)
}
cis <- lapply(y, freqCI)
ci.df <- data.frame(matrix(unlist(cis), nrow=length(cis), byrow=TRUE))
ci.df$MLE <- unlist(lapply(y, mean)); ci.df <- ci.df[,c(1,3,2)]
colnames(ci.df) <- c("Lower", "MLE", "Upper")
ci.df
```

Plots and Figures

Posterior density of theta for each pitcher

```r
postPitchers <- as.data.frame(chains[,14:19])
colors <- c("blue", "red", "darkmagenta", "darkorange", "aquamarine4", "black")
plot(density(postPitchers[,1]),
     xlim=c(13.5, 17),
     col=colors[1], lwd=5,
     main="", xlab="Theta | Y")
abline(v=median(postPitchers[,1]), lty=2, col=colors[1], lwd=2)
for (i in 2:6) {
  lines(density(postPitchers[,i]), col=colors[i], lwd=5)
  abline(v=median(postPitchers[,i]), lty=2, col=colors[i], lwd=2)
}
legend("topright", fill=colors,
       legend=c("Walker Buehler", "Nathan Eovaldi", "Zack Wheeler",
                "Taijuan Walker", "Shane McClanahan", "Max Fried"))
```

Posterior predictive density for each pitcher

```r
predPitchers <- as.data.frame(preds)
colors <- c("blue", "red", "darkmagenta", "darkorange", "aquamarine4", "black")
plot(density(predPitchers[,1], adjust=2),
     xlim=c(5, 27),
     col=colors[1], lwd=4,
     main="", xlab="Y*")
abline(v=mean(predPitchers[,1]), lty=2, col=colors[1], lwd=2)
for (i in 2:6) {
  lines(density(predPitchers[,i], adjust=2), col=colors[i], lwd=4)
  abline(v=mean(predPitchers[,i]), lty=2, col=colors[i], lwd=2)
}
legend("topright", fill=colors,
       legend=c("Walker Buehler", "Nathan Eovaldi", "Zack Wheeler",
                "Taijuan Walker", "Shane McClanahan", "Max Fried"))
```

Sensitivity Analysis: Nathan Eovaldi

```r
nP <- 10000
set.seed(12)
a0 <- rnorm(nP, 16, sqrt(3)); b0 <- rnorm(nP, 1, 0.25)
a1 <- rnorm(nP, 16, 1); b1 <- rnorm(nP, 1, 0.1)
a2 <- rnorm(nP, 32, sqrt(3)); b2 <- rnorm(nP, 2, 0.25)
a3 <- runif(nP, 14, 17); b3 <- runif(nP, 0.1, 1)
d0 <- rgamma(nP, a0, b0); dens0 <- density(d0); dens0$y = dens0$y/max(dens0$y)
d1 <- rgamma(nP, a1, b1); dens1 <- density(d1); dens1$y = dens1$y/max(dens1$y)
d2 <- rgamma(nP, a2, b2); dens2 <- density(d2); dens2$y = dens2$y/max(dens2$y)
d3 <- rnorm(nP, a3, b3); dens3 <- density(d3); dens3$y = dens3$y/max(dens3$y)
sens <- data.frame(chains[,15], alt1[,14], alt2[,14], alt3[,14])
colnames(sens) <- c("orig", "alt1", "alt2", "alt3")
colors <- c("red", "blue", "green", "orange")
lw <- 3
```

```r
plot(density(sens[,1]), xlim=c(12, 24), col=colors[1], lwd=7, lty=1,
     main="", xlab="Theta2 (Nathan Eovaldi)", ylab="Scaled Density")
lines(density(sens[,2]), col=colors[2], lwd=4, lty=1)
lines(density(sens[,3]), col=colors[3], lwd=3, lty=1)
lines(density(sens[,4]), col=colors[4], lwd=2, lty=1)
lines(dens0, col=colors[1], lwd=lw, lty=2)
lines(dens1, col=colors[2], lwd=lw, lty=2)
lines(dens2, col=colors[3], lwd=lw, lty=2)
lines(dens3, col=colors[4], lwd=lw, lty=2)
legend("topright", col=rep(colors,each=2), lwd=rep(3,8),
       lty=rep(c(2,1),4),
       legend=c("Original Model - Prior", "Original Model - Posterior",
                "Alternative 1 - Prior", "Alternative 1 - Posterior",
                "Alternative 2 - Prior", "Alternative 2 - Posterior",
                "Alternative 3 - Prior", "Alternative 3 - Posterior"))
```

Comparison of MCMC by hand to JAGS: Walker Buehler

```r
par(mfrow=c(1,3), cex.lab=1.5)

plot(density(aDraws[,1]), lwd=3, ylim=c(0,0.4),
     xlab="Posterior of a1", main="")
lines(density(aDrawsJags[,1]), lwd=3, lty=2)
abline(v=aComp[1,1:6],
       col=c("blue", "blue", "red", "red", "darkorange", "darkorange"),
       lty=c(1,2,1,2,1,2),
       lwd=rep(2,6))
legend("topright", col=c("blue", "blue", "red", "red", "darkorange", "darkorange"),
       lty=c(1,2,1,2,1,2),
       lwd=rep(3,6),
       legend=c("Median - MCMC", "Median - JAGS",
                "Credible LB - MCMC", "Credible LB - JAGS",
                "Credible UB - MCMC", "Credible UB - JAGS"))

plot(density(bDraws[,1]), lwd=2, ylim=c(0,2.2),
     xlab="Posterior of b1", main="")
lines(density(bDrawsJags[,1]), lwd=2, lty=2)
abline(v=bComp[1,1:6],
       col=c("blue", "blue", "red", "red", "darkorange", "darkorange"),
       lty=c(1,2,1,2,1,2),
       lwd=rep(2,6))
legend("topright", col=c("blue", "blue", "red", "red", "darkorange", "darkorange"),
       lty=c(1,2,1,2,1,2),
       lwd=rep(3,6),
       legend=c("Median - MCMC", "Median - JAGS",
                "Credible LB - MCMC", "Credible LB - JAGS",
                "Credible UB - MCMC", "Credible UB - JAGS"))

plot(density(thetaDraws[,1]), lwd=2, ylim=c(0,1.7),
     xlab="Posterior of Theta1", main="")
lines(density(thetaDrawsJags[,1]), lwd=2, lty=2)
abline(v=thetaComp[1,1:6],
       col=c("blue", "blue", "red", "red", "darkorange", "darkorange"),
       lty=c(1,2,1,2,1,2),
       lwd=rep(2,6))
legend("topright", col=c("blue", "blue", "red", "red", "darkorange", "darkorange"),
       lty=c(1,2,1,2,1,2),
       lwd=rep(3,6),
       legend=c("Median - MCMC", "Median - JAGS",
                "Credible LB - MCMC", "Credible LB - JAGS",
                "Credible UB - MCMC", "Credible UB - JAGS"))
```

```
par(mfrow=c(1,1), cex.lab=1)
```

Trace Plots: Zack Wheeler

```
par(mfrow=c(2,3), cex.main=3, cex.lab=2)
plot(aDraws[,3], type='l', main="", xlab="a3", ylab="", xaxt='n')
plot(bDraws[,3], type='l', main="MCMC", xlab="b3", ylab="", xaxt='n')
plot(thetaDraws[,3], type='l', main="", xlab="theta3", ylab="", xaxt='n')
plot(aDrawsJags[,3], type='l', main="", xlab="a3", ylab="", xaxt='n')
plot(bDrawsJags[,3], type='l', main="JAGS", xlab="b3", ylab="", xaxt='n')
plot(thetaDrawsJags[,3], type='l', main="", xlab="theta3", ylab="", xaxt='n')
```

```
par(mfrow=c(1,1), cex.main=1, cex.lab=1)
```