```
############################
# Taylor Last
# STAT 4350 FINAL PROJECT
############################
library(ggplot2)
library(rjags)
setwd('/Users/taylorlast/Documents/UGA_FourthYear/STAT_4350/Final Project')
baseball = read.csv('baseball.csv')

#########################################################################
## Hierarchical linear model with team-specific random intercept   ##
##                                                                 ##
#########################################################################
## Load team covariates
attach(baseball)

## More exploratory data analysis
# By team denomination
ggplot(data=baseball,aes(x=Team,y=ERA.))+
  geom_boxplot()+
  theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust = 1),plot.title =
element_text(hjust = 0.5))+
  ggtitle('ERA adjusted by team')

n = dim(baseball)[1]
p = dim(baseball)[2] - 3
n.teams = length(unique(Team))

team_list = as.factor(Team)

## (1) Create data list
dataList <- list(
  "n" = n,
  "p" = p,
  "n.teams" = n.teams,
  "Y" = ERA.,
  "HR9" = HR.9,
  "BABIP" = BABIP,
  "LOB" = LOB.,
  "WAR" = WAR,
  "KBB" = K.BB.,
  "WHIP" = WHIP,
  "WIN" = WIN.,
  "Team" = as.factor(Team))
```

```
## (2) Specify list of parameter(s) to be monitored
parameters <- c("alpha","beta","sig2","sig2.alpha","icc")

## (3) Specify initial values for parameter(s) in Metropolis-Hastings algorithm
initsValues <- list(
  "alpha" = rep(0,n.teams),
  "beta" = rep(0,p),
  "tau2" = 1,
  "tau2.alpha" = 1
)

## (4) Specify parameters for running Metropolis-Hastings algorithm
adaptSteps <- 10000             # number of steps to "tune" the samplers
burnInSteps <- 20000            # number of steps to "burn-in" the samplers
nChains <- 3                  # number of chains to run
numSavedSteps <- 50000          # total number of steps in chains to save
thinSteps <- 10                 # number of steps to "thin" (1 = keep every step)
nIter <- ceiling((numSavedSteps*thinSteps)/nChains)  # steps per chain

## (5) Create, initialize, and adapt the model
# This will require you to create a separate .txt file which specifies
# the model
jagsModel <- jags.model("baseball.txt",
                        data = dataList,
                        inits = initsValues,
                        n.chains = nChains,
                        n.adapt = adaptSteps)

## (6) Burn-in the algorithm
if(burnInSteps>0){
  cat( "Burning in the MCMC chain...\n")
  update(jagsModel, n.iter = burnInSteps)
}

## (7) Run MCMC algorithm
cat("Sampling final MCMC chain...\n" )
codaSamples <- coda.samples(jagsModel,
                            variable.names = parameters,
                            n.iter = nIter,
                            thin = thinSteps)

## (8) Diagnose convergence and plot posterior densities
# par(ask=T)
# plot(codaSamples)
```

```
## (9) Calculate numerical summaries for the posterior samples
summary(codaSamples)

## (10) Retrieve posterior samples for later use
mcmcChain <- as.matrix(codaSamples)
library(MCMCvis)


MCMCtrace(codaSamples,ISB = FALSE ,
          exact = TRUE,
          pdf = FALSE)

# Examine the posterior distribution of the team random effect
alphaSamples <- matrix(NA, dim(mcmcChain)[1], n.teams)
for(i in 1:n.teams){
  alphaSamples[,i]  <- mcmcChain[, paste("alpha[",i,"]", sep="")]
}
levels(team_list)
par(mfrow=c(1,1), ask=F)
boxplot(as.data.frame(alphaSamples),
        names=as.character(1:n.teams),
        main="Posterior samples of alphas",
        xlab="Team")
abline(h=0)

teams_df = as.matrix(data.frame(num = c(1:30),team = sort(unique(Team))))

rank.alpha <- matrix(NA, dim(mcmcChain)[1], n.teams)
for(i in 1:dim(mcmcChain)[1]){
  rank.alpha[i,] <- rank(alphaSamples[i,])
}
avg.rank <- rank(apply(rank.alpha, 2, mean))


# par(mfrow=c(1,1), ask=F)
# plot(c(-15,15), c(1,30), type='n', axes=F, xlab="", ylab="", main="Team Rankings")
# axis(1,at=seq(-20, 20, length=5))
# for(i in 1:n.teams){
#   lines(quantile(alphaSamples[,i], c(0.025,0.975)), rep(avg.rank[i],2))
#   points(mean(alphaSamples[,i]), avg.rank[i], pch=19)
#   text(15, avg.rank[i], i, cex=.5)
# }

# ranked_teams = vector()
#
```

```
#
# for(i in 1:n.teams){
#   for(j in 1:n.teams){
#     if (avg.rank[j] == i){
#       ranked_teams[i]<- teams_df[j,2]
#     }
#   }
# }
# ranked_teams

ranked_teams_idx = vector()
for(i in 1:n.teams){
  for(j in 1:n.teams){
    if (avg.rank[j] == i){
      ranked_teams_idx[i]<- teams_df[j,1]
    }
  }
}
ranked_teams_idx = trimws(ranked_teams_idx)

# Sorted Alphas
temp_vec=vector()
for(i in 1:n.teams){
  temp_vec[i] = paste0('alpha[',ranked_teams_idx[i],']')
}

library(bayesplot)

# Plot the alphas to see random effects
mcmc_intervals(codaSamples,regex_pars = '^[alpha]')+
  scale_y_discrete(
    labels =
      c('alpha[1]' = levels(team_list)[1],
        'alpha[2]' = levels(team_list)[2],
        'alpha[3]' = levels(team_list)[3],
        'alpha[4]' = levels(team_list)[4],
        'alpha[5]' = levels(team_list)[5],
        'alpha[6]' = levels(team_list)[6],
        'alpha[7]' = levels(team_list)[7],
        'alpha[8]' = levels(team_list)[8],
        'alpha[9]' = levels(team_list)[9],
        'alpha[10]' = levels(team_list)[10],
        'alpha[11]' = levels(team_list)[11],
        'alpha[12]' = levels(team_list)[12],
        'alpha[13]' = levels(team_list)[13],
```

```
        'alpha[14]' = levels(team_list)[14],
        'alpha[15]' = levels(team_list)[15],
        'alpha[16]' = levels(team_list)[16],
        'alpha[17]' = levels(team_list)[17],
        'alpha[18]' = levels(team_list)[18],
        'alpha[19]' = levels(team_list)[19],
        'alpha[20]' = levels(team_list)[20],
        'alpha[21]' = levels(team_list)[21],
        'alpha[22]' = levels(team_list)[22],
        'alpha[23]' = levels(team_list)[23],
        'alpha[24]' = levels(team_list)[24],
        'alpha[25]' = levels(team_list)[25],
        'alpha[26]' = levels(team_list)[26],
        'alpha[27]' = levels(team_list)[27],
        'alpha[28]' = levels(team_list)[28],
        'alpha[29]' = levels(team_list)[29],
        'alpha[30]' = levels(team_list)[30]),
    limits = c(temp_vec)
  )+
  ggtitle('MLB teams random effects')+
  theme(plot.title = element_text(hjust = 0.5))

era.xera = read.csv('era_vs_expected.csv')
era.xera = era.xera[order(era.xera$diff),]
era.xera$Team <- factor(era.xera$Team, levels = era.xera$Team[order(era.xera$diff)])
ggplot(data = era.xera, aes(x = Team,y=diff))+
  geom_bar(stat='identity')+
  coord_flip()+
  ggtitle('ERA - xERA from FanGraphs')+
  theme(plot.title = element_text(hjust = 0.5))


fip.xfip = read.csv('FIP_vs_xFIP.csv')
fip.xfip = fip.xfip[order(fip.xfip$diff),]
fip.xfip$Team <- factor(fip.xfip$Team, levels = fip.xfip$Team[order(fip.xfip$diff)])
ggplot(data = fip.xfip, aes(x = Team,y=diff))+
  geom_bar(stat='identity')+
  coord_flip()+
  ggtitle('FIP-  -  xFIP- from FanGraphs')+
  theme(plot.title = element_text(hjust = 0.5))



# Boxplot of posteriors for beta
boxplot(mcmcChain[,31:38], main = "Posterior samples of beta",
```

```
        names=c("Intercept",
                "HR9",
                "BABIP",
                "LOB",
                "WAR",
                "KBB",
                "WHIP",
                "WIN"))
abline(h = 0, lty = 2, col="red")

# 50% (thin line) and 95% (thick line) credible interval "caterpillar" plots
# dot is the posterior median
MCMCplot(codaSamples, params = "beta",
         main = "Posterior CIs for beta")

MCMCplot(codaSamples, params = c("sig2",'sig2.alpha'),
         main = "Posterior CIs for sigmas")

# Posterior for intraclass correlation
plot(codaSamples[,39])
median(mcmcChain[,39])
mean(mcmcChain[,3])
mean(mcmcChain[,9])
mean(mcmcChain[,35])
```