STAT 8700 Final Question 5

Brian Detweiler

Thursday, December 15th

5. The obvious use for the model you developed in Question 4 would be simulate the remainder of the season (the other 240) games to try to predict what might happen. In order to do that, however, there are some useful preliminary steps. Also, for convenience, set all diagonal elements of the matrix to be equal to 1.

(a) In R (not JAGS) create a 20×20 matrix, such that the (i, j)th element of the matrix equals 1 if the game where team i is the home team and team j is the road team has been played (that is, is in the data set) and 0 otherwise.

```
# -1 Away win, 0 tie, 1 Home win
epl$Result <- sign(epl$Home.Goals - epl$Away.Goals)

epl.M <- matrix(data = 0,
                nrow = 20,
                ncol = 20)

for (i in 1:20) {
  for (j in 1:20) {
    tmp <- epl %>% filter(Home.ID == i, Away.ID == j)
    epl.M[i, j] <- nrow(tmp)
    if (i == j) {
      epl.M[i, j] <- 1
    }
  }
}
```

(b) There are no playoffs in the English Premier League. The winner is determined by which team accumulates the most number of points, with 3 points awarded to the team that wins a game, and both teams awarded 1 point each in the case of a tie. In R (not JAGS) create two matrices, each 20×20, such that the (i, j)th entry of the first matrix is the number of points won by the home team (team i) in the game between team i and team j, and the (i, j)th entry of the second matrix is the number of points won by the away team (team j) in the game between team i and team j. Obviously, if the game between team i (home) and team j (away) hasn't yet taken place, then the (i, j)th entries of both matrices will be zero.

```
epl.M.Home <- matrix(data = 0,
                     nrow = 20,
                     ncol = 20)

epl.M.Away <- matrix(data = 0,
                     nrow = 20,
                     ncol = 20)

# -1 (loss) becomes 0, 0 (tie) becomes 1, and 1 (win) becomes 2
epl$Result.Home <- epl$Result + 1
# Since a win is actually worth 3, replace 2 with 3
epl$Result.Home[epl$Result.Home == 2] <- 3

# Invert the Home results
epl$Result.Away <- epl$Result * -1
# -1 (loss) becomes 0, 0 (tie) becomes 1, and 1 (win) becomes 2
epl$Result.Away <- epl$Result.Away + 1
# Since a win is actually worth 3, replace 2 with 3
epl$Result.Away[epl$Result.Away == 2] <- 3

for (i in 1:20) {
  for (j in 1:20) {
    tmp <- epl %>% filter(Home.ID == i, Away.ID == j)
    if (nrow(tmp) > 0) {
      epl.M.Home[i, j] <- tmp$Result.Home
      epl.M.Away[j, i] <- tmp$Result.Away
    }
  }
}
```

(c) How would you use the two matrices you defined in the previous part to determine the current number of points for a particular team? Compute the current points total for Chelsea.

We can compute the current score by summing the rows and combining the Home and Away scores.

```
chelsea.home.sum <- sum(epl.M.Home[teams.with.names$Home.ID[teams.with.names$Home.Team == "Chelsea"], ])
chelsea.away.sum <- sum(epl.M.Away[teams.with.names$Home.ID[teams.with.names$Home.Team == "Chelsea"], ])

chelsea.total <- chelsea.home.sum + chelsea.away.sum
```

Chelsea has a total of 34 points.

(d) Now it is time to simulate the remaining games. Rather than trying to figure out which games are needed to be simulated, it is actually easier to simulate games for all possible combinations of teams, and then just ignore the ones you don't need. In your JAGS model, add code to simulate two new matrices of values, `SimGoalsHome[i,j]` and `SimGoalsAway[i,j]` which simulate the number of goals scored for the home team (team i) and the away team (team j) respectively for the game between team i and team j. (You can include the simulations of team i vs team i, as we can remove them later).

```
teams <- unique(c(epl$Home.Team, epl$Away.Team))

epl.M.Home <- matrix(data = -1,
                     nrow = 20,
                     ncol = 20)

epl.M.Away <- matrix(data = -1,
                     nrow = 20,
                     ncol = 20)

epl$Result.Home <- epl$Result + 1
epl$Result.Home[epl$Result.Home == 2] <- 3

# Invert the Home results
epl$Result.Away <- epl$Result * -1
epl$Result.Away <- epl$Result.Away + 1
epl$Result.Away[epl$Result.Away == 2] <- 3

for (i in 1:20) {
  for (j in 1:20) {
    tmp <- epl %>% filter(Home.ID == i, Away.ID == j)
    if (nrow(tmp) > 0) {
      epl.M.Home[i, j] <- tmp$Result.Home
      epl.M.Away[j, i] <- tmp$Result.Away
    }
  }
}
epl.M.Home
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]   -1   -1    3    3   -1   -1   -1   -1    0    -1    -1     1     3
##  [2,]    0   -1   -1   -1    3    3    1   -1    3     0    -1    -1    -1
##  [3,]   -1   -1   -1   -1   -1    3    3   -1    3    -1     0    -1    -1
##  [4,]   -1    3   -1   -1   -1    3   -1    3    0    -1     3    -1    -1
##  [5,]   -1   -1    1   -1   -1   -1   -1   -1    0     0    -1    -1     3
##  [6,]   -1   -1   -1   -1    1   -1   -1   -1   -1    -1     1     3    -1
##  [7,]    0   -1   -1    0   -1   -1   -1    3   -1    -1     0    -1     3
##  [8,]    1    3   -1   -1    3   -1   -1   -1   -1    -1    -1     1     1
##  [9,]   -1   -1   -1   -1   -1   -1    3    3   -1    -1     1    -1    -1
## [10,]   -1   -1    3    0   -1    1   -1   -1   -1    -1    -1     1     1
## [11,]    1    1   -1   -1   -1   -1   -1    3   -1     0    -1    -1     3
## [12,]   -1   -1    3    0    0   -1    3   -1   -1    -1    -1    -1    -1
```

```
## [13,]   -1    3   -1    0   -1    3   -1   -1    1    -1    -1    -1    -1
## [14,]   -1    3    0   -1   -1   -1   -1   -1   -1     0    -1    -1    -1
## [15,]    0   -1   -1   -1    0    0    3    3   -1    -1    -1     0    -1
## [16,]   -1   -1   -1    1    3   -1    0   -1    0     0     0    -1    -1
## [17,]   -1   -1   -1   -1    3   -1   -1    1    1     3    -1    -1    -1
## [18,]    0   -1    1    0   -1   -1    3    3   -1    -1     3    -1    -1
## [19,]   -1    3   -1   -1   -1    0   -1   -1   -1     0    -1     1    -1
## [20,]    0   -1    3   -1   -1   -1   -1   -1   -1    -1    -1     1     0
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20]
##  [1,]    -1    -1     3     1    -1    -1    -1
##  [2,]    -1    -1     0    -1     3    -1    -1
##  [3,]    -1     0    -1     1    -1     3    -1
##  [4,]    -1    -1    -1     3    -1    -1     3
##  [5,]     3    -1    -1    -1    -1     0     0
##  [6,]     3    -1     1     1    -1    -1     3
##  [7,]     0    -1    -1    -1    -1     1    -1
##  [8,]    -1    -1     3    -1    -1     0    -1
##  [9,]    -1     3    -1    -1     3     3    -1
## [10,]    -1     3    -1    -1    -1    -1     3
## [11,]     1    -1    -1    -1    -1    -1     1
## [12,]     1    -1    -1     0     0    -1    -1
## [13,]    -1     1     3    -1     1    -1    -1
## [14,]    -1     3     3     0    -1     1    -1
## [15,]    -1    -1    -1    -1    -1     1    -1
## [16,]    -1    -1    -1    -1     1    -1    -1
## [17,]    -1     3     3    -1    -1    -1     3
## [18,]     0    -1    -1    -1    -1    -1    -1
## [19,]    -1    -1    -1     1     3    -1     3
## [20,]     1     3    -1    -1     0    -1    -1
```

```
fileName <- "Final.5.d"

modelString ="
model {
  for(i in 1:120) {
    H[i] ~ dpois(lambda[HomeTeam[i], AwayTeam[i]])
    A[i] ~ dpois(theta[HomeTeam[i], AwayTeam[i]])
  }

  for(i in 1:n_teams) {
    for(j in 1:n_teams) {
      SimGoalsHome[i, j] ~ dpois(lambda[HomeTeam[i], AwayTeam[j]])
      SimGoalsAway[i, j] ~ dpois(theta[HomeTeam[j], AwayTeam[i]])

      # Possible scores are 0, 1, or 3
      SimPointsHome[i, j] <- combinedScoreHome[i, j] - tieHome[i, j]

      # step is 1 if x >= 0, so it'll either be 3 or 0
      combinedScoreHome[i, j] <- step(SimGoalsHome[i, j] - SimGoalsAway[i, j]) * 3
      # equals is 1 if x == y, so this will either be 2 or 0
      tieHome[i, j] <- equals(SimGoalsHome[i, j], SimGoalsAway[i, j]) * 2

      # Possible scores are 0, 1, or 3
      SimPointsAway[i, j] <- combinedScoreAway[i, j] - tieAway[i, j]
```

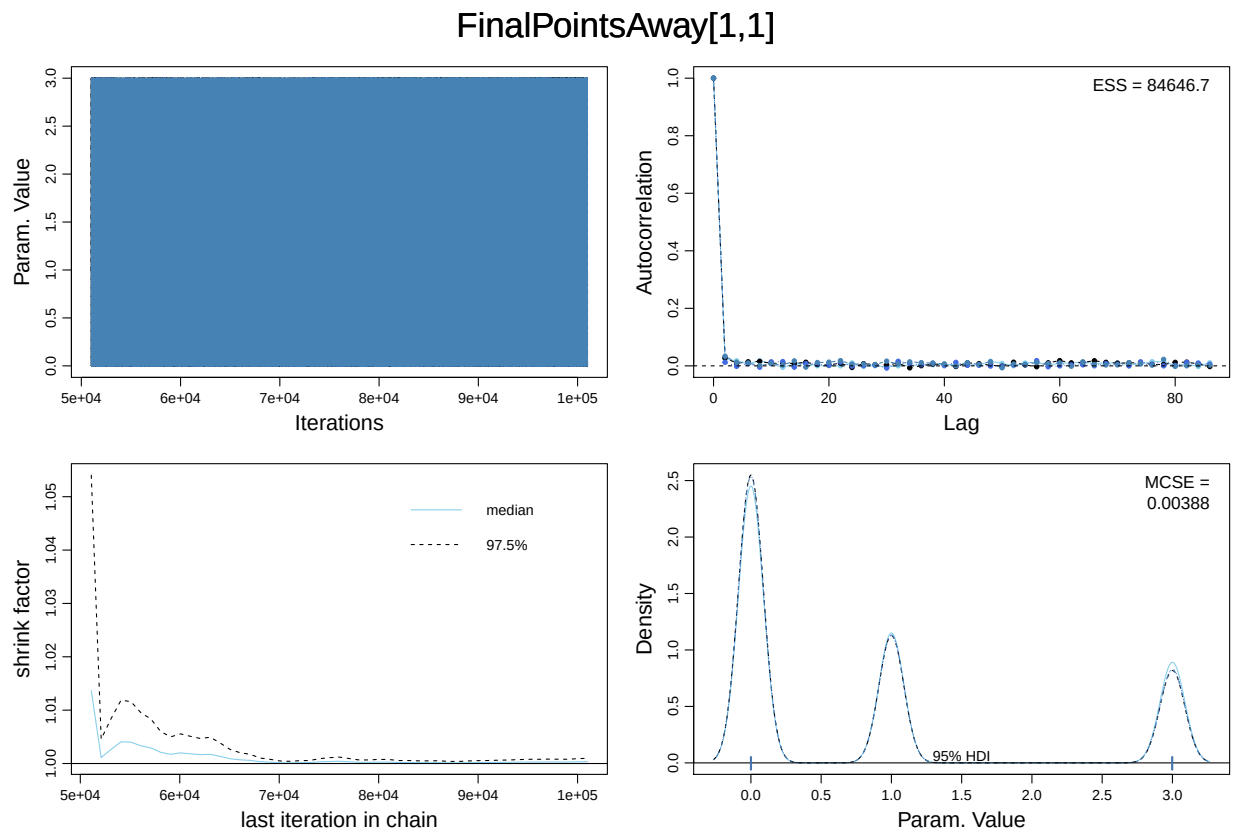
```
    # step is 1 if x >= 0, so it'll either be 3 or 0
    combinedScoreAway[i, j] <- step(SimGoalsAway[i, j] - SimGoalsHome[i, j]) * 3
    # equals is 1 if x == y, so this will either be 2 or 0
    tieAway[i, j] <- equals(SimGoalsAway[i, j], SimGoalsHome[i, j]) * 2

    # Data will be -1 if the game has not been played
    FinalPointsHome[i, j] <- step(HomePointsData[i, j]) * HomePointsData[i, j] + equals(HomePointsData

    FinalPointsAway[i, j] <- step(AwayPointsData[i, j]) * AwayPointsData[i, j] + equals(AwayPointsData

  }

  TotalPoints[i] <- sum(FinalPointsHome[i, 1:n_teams]) + sum(FinalPointsAway[i, 1:n_teams])
}

LeagueRank <- rank(TotalPoints)

for(k in 1:n_teams) {
  for(m in 1:n_teams) {
    LeagueRanks[k, m] <- equals(k, LeagueRank[m])
  }
}

for(i in 1:n_teams) {
  for(j in 1:n_teams) {
    lambda[i, j] <- exp(mu + a[i] - d[j] + gamma)
    theta[i, j] <- exp(mu + a[j] - d[i])
  }
}

for(j in 1:(n_teams - 1)) {
  a[j] ~ dnorm(group_attack, group_tau)
  d[j] ~ dnorm(group_defense, group_tau)
}
a[n_teams] <- -sum(a[1:(n_teams - 1)])
d[n_teams] <- -sum(d[1:(n_teams - 1)])

gamma ~ dgamma(0.01, 0.01)

rank.a <- rank(a)
rank.d <- rank(d)

for (k in 1:n_teams) {
  for (m in 1:n_teams) {
    #ranks.a[k, m] ~ dbern(pa[k, m])
    #ranks.d[k, m] ~ dbern(pd[k, m])
    #pa[k, m] <- equals(k, rank.a[m])
    #pd[k, m] <- equals(k, rank.d[m])
    ranks.a[k, m] <- equals(k, rank.a[m])
    ranks.d[k, m] <- equals(k, rank.d[m])
  }
}
```

```
  mu ~ dnorm(0, 0.0625)

  group_attack ~ dnorm(0, 0.0625)
  group_defense ~ dnorm(0, 0.0625)
  group_tau <- 1 / pow(group_sigma, 2)
  group_sigma ~ dunif(0, 3)
}
"


writeLines(modelString, con=fileName)

data.list <- list(H = epl$Home.Goals,
                  A = epl$Away.Goals,
                  HomeTeam = epl$Home.ID,
                  AwayTeam = epl$Away.ID,
                  HomePointsData = epl.M.Home,
                  AwayPointsData = epl.M.Away,
                  n_teams = length(teams))


epl.model = jags.model(file=fileName,
                       data=data.list,
                       n.chains=4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 240
##    Unobserved stochastic nodes: 843
##    Total graph size: 23679
##
## Initializing model
```

```
update(epl.model, n.iter=50000)

epl.samples <- coda.samples(epl.model,
                            variable.names = c("SimGoalsHome",
                                               "SimGoalsAway",
                                               "SimPointsHome",
                                               "SimPointsAway",
                                               "FinalPointsHome",
                                               "FinalPointsAway",
                                               "TotalPoints",
                                               "LeagueRank",
                                               "LeagueRanks",
                                               "rank.d",
                                               "ranks.a",
                                               "ranks.d"),
                            n.iter = 50000,
                            thin = 2)
```

```
diagMCMC(epl.samples)

epl.DIC <- dic.samples(model = epl.model, n.iter = 20000, thin = 50)
```




```
epl.DIC
```

```
## Mean deviance:  695
## penalty 27.32
## Penalized deviance: 722.4
```

```
epl.samples.M <- as.matrix(epl.samples)

smry <- summary(epl.samples)
```

(e) Making use of the step and equals functions in JAGS, add code to create two new matrices of values `SimPointsHome[i,j]` and `SimPointsAway[i,j]` which calculate the number of points won by the home team (team i) and away team (team j) in the simulated game between team i and team j conducted in the previous part.

See model.

(f) In JAGS, add code to create two new matrices of values `FinalPointsHome[i,j]` and `FinalPointsAway[i,j]` such that the (i, j)th entry of `FinalPointsHome` is equal to the actual points won by the home team in the game between team i and team j if that game has already been played (that is, it is in the data), or the number of points won by the home team in the simulated game, if that game has not yet been played (that is, it is not in the data set). Create `FinalPointsAway` in a similar way. [Hint: You can pass the matrices you created in parts (a) and (b) into JAGS as data.]

See model.

(g) In JAGS, create a vector `TotalPoints[k]` that calculates the total points obtained by team k for the season. This will be based on the method you specified in part (c), and will make use of the sum function in JAGS.

See model.

(h) Using posterior mean points as a way to rank the teams, what are the top 4 teams (in order)? What are the bottom 3 teams?

See model for first part.

```
total.points <- rep(0, 20)
for (i in 1:20) {
  total.points[i] <- smry$quantiles[paste0("TotalPoints[", i, "]"), 3]
}

teams.with.names$TotalPoints <- total.points
teams.final <- teams.with.names %>% arrange(desc(TotalPoints))
```

The top four teams ranked by final points are Chelsea, Arsenal, Liverpool, Manchester City.

The bottom three teams ranked by final points are Sunderland, Hull City, West Ham United.

(i) Just like in the previous question, rather than looking at posterior means, we could look at where their points total ranks compared to the other teams during our simulations. In your JAGS model, add a line defining a vector of parameters `LeagueRank` so that the kth element of `LeagueRank`, `LeagueRank[k]` is the rank of team k's point total compared to the points total of the other teams, with 1 indicating that team k had the highest point total, and 20 indicating that team k had the lowest point total. Using the posterior mean ranks for each team, what are the top 4 teams (in order)? What are the bottom 3 teams?

```
league.rank <- rep(0, 20)
for (i in 1:20) {
  league.rank[i] <- smry$quantiles[paste0("LeagueRank[", i, "]"), 3]
}
```

```
teams.with.names$LeagueRank <- league.rank
league.rank.final <- teams.with.names %>% arrange(desc(LeagueRank))
```

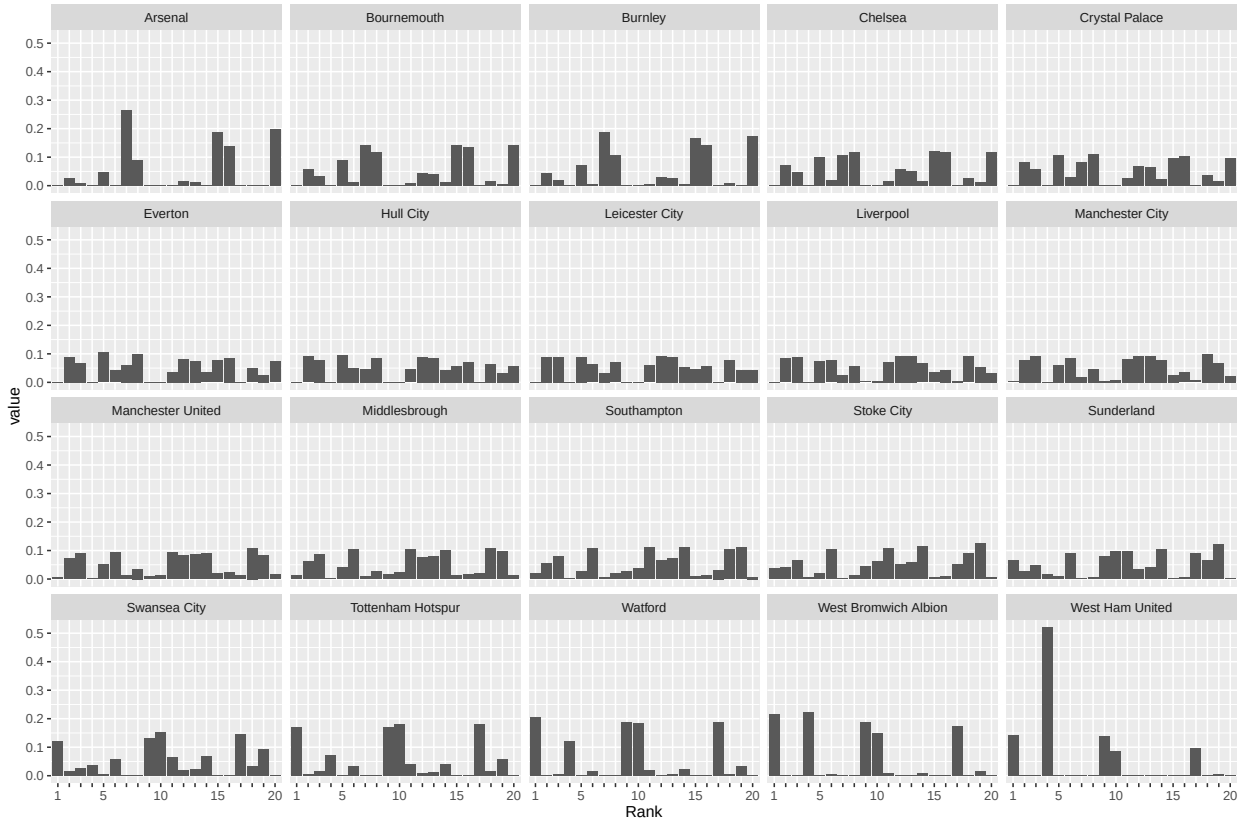
The top four teams ranked by final points are Chelsea, Arsenal, Liverpool, Manchester City.

The bottom three teams ranked by final points are Swansea City, Hull City, West Ham United.

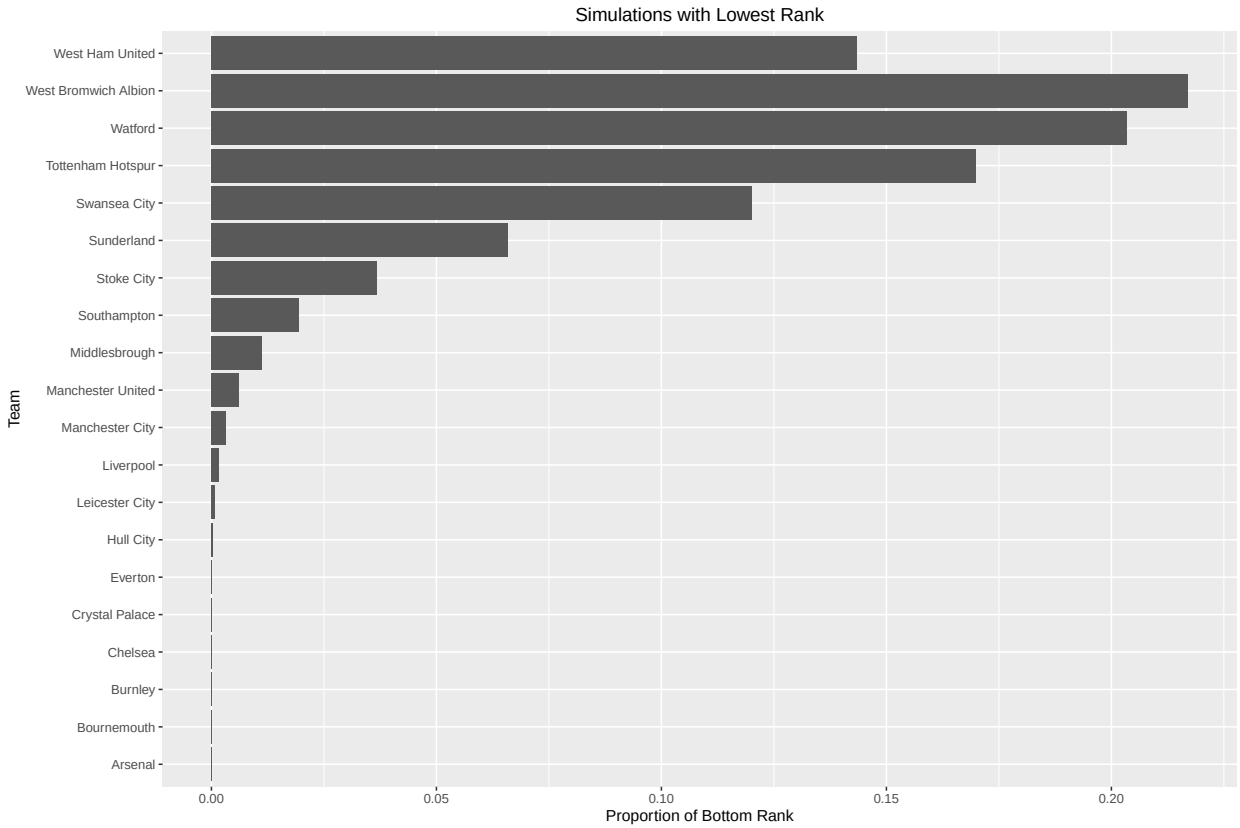
(j) Just like in the previous question, we could look to see in what proportion of simulations was a team's point total ranked in a particular spot. In your JAGS model, add code to define `LeagueRanks[k,m]` which takes value 1 if the rank of the points total for team k is equal to m and 0 otherwise. Which team finished top of the league (most points) in the highest proportion of simulations?

```
ranks.df <- data.frame(teams.with.names)
ranks.df$rank1 <- 0
ranks.df$rank2 <- 0
ranks.df$rank3 <- 0
ranks.df$rank4 <- 0
ranks.df$rank5 <- 0
ranks.df$rank6 <- 0
ranks.df$rank7 <- 0
ranks.df$rank8 <- 0
ranks.df$rank9 <- 0
ranks.df$rank10 <- 0
ranks.df$rank11 <- 0
ranks.df$rank12 <- 0
ranks.df$rank13 <- 0
ranks.df$rank14 <- 0
ranks.df$rank15 <- 0
ranks.df$rank16 <- 0
ranks.df$rank17 <- 0
ranks.df$rank18 <- 0
ranks.df$rank19 <- 0
ranks.df$rank20 <- 0

for (i in 1:20) {
  for (j in 1:20) {
    ranks.df[i, paste0("rank", j)] <- mean(epl.samples.M[,paste0("LeagueRanks[", i, ",", j, "]")])
  }
}
ranks.sub.df <- ranks.df[,-1]
ranks.sub.df <- ranks.sub.df[, -2]
ranks.sub.df <- ranks.sub.df[, -2]

ranks.melt <- melt(ranks.sub.df, id.vars = "Home.Team")
ggplot(ranks.melt, aes(x=variable, y=value)) +
  geom_bar(stat="identity") +
  facet_wrap(~Home.Team) +
  scale_x_discrete("Rank", labels = c("1", "", "", "",
                                      "5", "", "", "", "",
                                      "10", "", "", "", "",
```

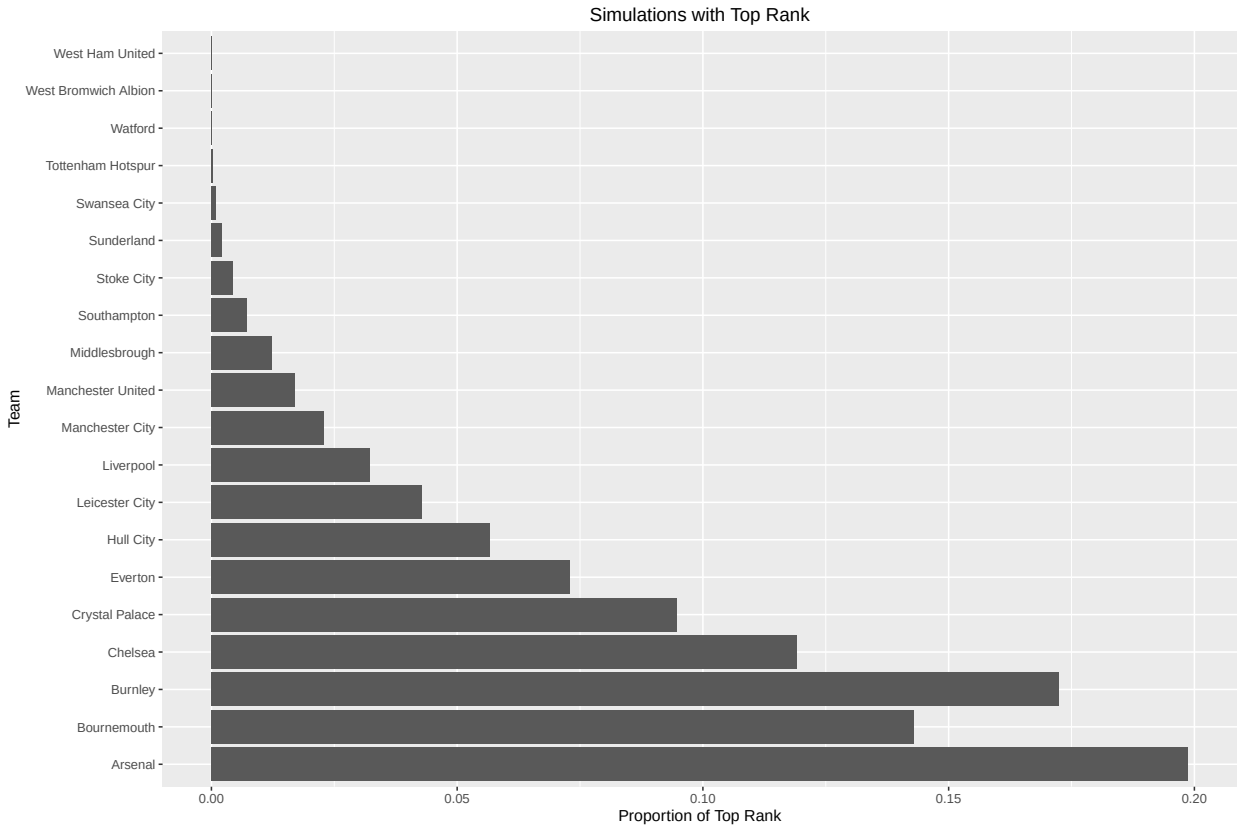
```
                                "15", "", "", "", "",
                                "20"))
```




```
ranks.sub.df.rank1 <- ranks.sub.df %>% select(Home.Team, rank1) %>% arrange(rank1)
ranks.sub.df.rank20 <- ranks.sub.df %>% select(Home.Team, rank20) %>% arrange(rank20)

ggplot(ranks.sub.df.rank1, aes(x=Home.Team, y=rank1)) +
  geom_bar(stat="identity") +
  coord_flip() +
  labs(title="Simulations with Lowest Rank", x="Team", y="Proportion of Bottom Rank")
```




```
ggplot(ranks.sub.df.rank20, aes(x=Home.Team, y=rank20)) +
  geom_bar(stat="identity") +
  coord_flip() +
  labs(title="Simulations with Top Rank", x="Team", y="Proportion of Top Rank")
```



(k) In the English Premier League, the 4 teams that finish in the top 4 positions qualify for a European tournament. Using your work from the previous part, for each team, list the proportion of simulations in which the team finished in the top 4 (from most likely to least likely). You can do this by adding things to your JAGS model, or calculating from the posterior summaries.

```
team.top.4 <- c()
for (i in 1:20) {
  team <- ranks.df$Home.Team[i]
  percent <- ranks.df$rank20[ranks.df$Home.Team == team] +
    ranks.df$rank19[ranks.df$Home.Team == team] +
    ranks.df$rank18[ranks.df$Home.Team == team] +
    ranks.df$rank17[ranks.df$Home.Team == team]

  team.top.4 <- c(team.top.4, percent)
}

ranks.df$PercentTop4 <- team.top.4

top4 <- ranks.df %>% arrange(desc(PercentTop4)) %>% select(Home.Team, PercentTop4)
top4[1:4, ]
```

```
##           Home.Team PercentTop4
## 1        Sunderland     0.27691
```

```
## 2      Swansea City     0.27407
## 3        Stoke City     0.26896
## 4 Tottenham Hotspur     0.25407
```

(l) In the English Premier League, the 3 teams that finish in the bottom 3 positions are relegated, meaning they will not be in the English Premier League the following season, but rather a lesser league (they are replaced by the 3 teams that finished at the top of this lesser league). List the proportion of simulations in which each team finished in the bottom 3 (from most likely to least likely). You can do this by adding things to your JAGS model, or calculating from the posterior summaries. Show all working.

```
team.bottom.3 <- c()
for (i in 1:20) {
  team <- ranks.df$Home.Team[i]
  percent <- ranks.df$rank1[ranks.df$Home.Team == team] +
    ranks.df$rank2[ranks.df$Home.Team == team] +
    ranks.df$rank3[ranks.df$Home.Team == team]

  team.bottom.3 <- c(team.bottom.3, percent)
}

ranks.df$PercentBottom3 <- team.bottom.3

bottom3 <- ranks.df %>% arrange(desc(PercentBottom3)) %>% select(Home.Team, PercentBottom3)
bottom3[1:3, ]
```

```
##              Home.Team PercentBottom3
## 1 West Bromwich Albion        0.22018
## 2              Watford        0.21151
## 3    Tottenham Hotspur        0.19040
```

■