STAT 8700 Final Question 6

Brian Detweiler

Thursday, December 15th

```r
# -1 Away win, 0 tie, 1 Home win
epl$Result <- sign(epl$Home.Goals - epl$Away.Goals)
epl.M.Home <- matrix(data = 0,
                     nrow = 20,
                     ncol = 20)

epl.M.Away <- matrix(data = 0,
                     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
    }
  }
}
```

6. Consider the Poisson Regression Model:

$$Y_i \sim Poisson(\lambda_i)$$
$$log(\lambda_i) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_{ik}$$

and recall that $E[Y_i] = \lambda_i$ and $Var(Y_i) = \lambda_i$.

As discussed in class, one of the concerns with the Poisson regression model is the requirement that the mean and variance be equal, and this can cause problems if the data is overdispersed.

One solution is to replace the Poisson distribution with a version of the Negative Binomial distribution

$$P(Y_i = y) = \frac{\Gamma(y+r)}{y!\Gamma(r)} p_i^r (1-p_i)^y$$

which is parameterized by p_i and r with $0 \leq p_i \leq 1$ and $r > 0$ (traditionally when you first learn the Negative Binomial distribution you learn that r has to be an integer, but in reality it can be any positive real number).

In this parameterization, $E[Y_i] = \frac{r(1-p_i)}{p_i}$ and $Var(y_i) = \frac{r(1-p_i)}{p_i^2}$.

(a) If we plan to replace the Poisson distribution with the above Negative Binomial distribution in our regression, we would like to keep the same link function, namely that

$log(\lambda_i) = log(E[Y_i]) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$

Equating the mean of the Poisson with the mean of the Negative Binomial, express p_i as a function of λ_i and r.

Equating the Poisson mean with the Negative Binomial mean, we get

$$\begin{aligned} E[Y_i] = \lambda_i &= \frac{r(1-p_i)}{p_i} \\ p_i \lambda_i &= r(1-p_i) \\ \frac{p_i}{(1-p_i)} &= \frac{r}{\lambda_i} \end{aligned}$$

(b) Using your answer to part (a), rewrite the $Var(Y_i)$ in the Negative Binomial case in terms of λ_i and r.

$$\begin{aligned} Var[Y_i] &= \frac{r(1-p_i)}{p_i^2} \\ \frac{p_i^2}{(1-p_i)} &= \frac{r}{\lambda_i} + \frac{r^2}{\lambda_i^2} \end{aligned}$$

(c) The model introduced in Question 4 contains two Poisson regressions, one for the Home Goals, and one for the Away Goals. Replace each of these with Negative Binomial's and use your answer to part (a) to connect the parameters of the Negative Binomial to the existing link functions for $log(\lambda_{ij})$ and $log(\theta_{ij})$.

$$\begin{aligned} H_i &\sim NegBinomial(p_h, r_h) \\ A_i &\sim NegBinomial(p_a, r_a) \\ p_h &= \frac{r_h}{r_h + \lambda_{ij}} \\ p_a &= \frac{r_a}{r_a + \theta_{ij}} \\ r_h &\sim Uniform(0, 50) \\ r_a &\sim Uniform(0, 50) \\ \lambda_{ij} &= e^{\mu + a_i - d_j + \gamma} \\ \theta_{ij} &= e^{\mu + a_j - d_i} \end{aligned}$$

The priors on r_h and r_a are a Uniform with an upper bound of 50, which as noted in [1] is not restrictive. Simon Jackman notes that as $r \to \infty$, the Negative Binomial tends to the Poisson.

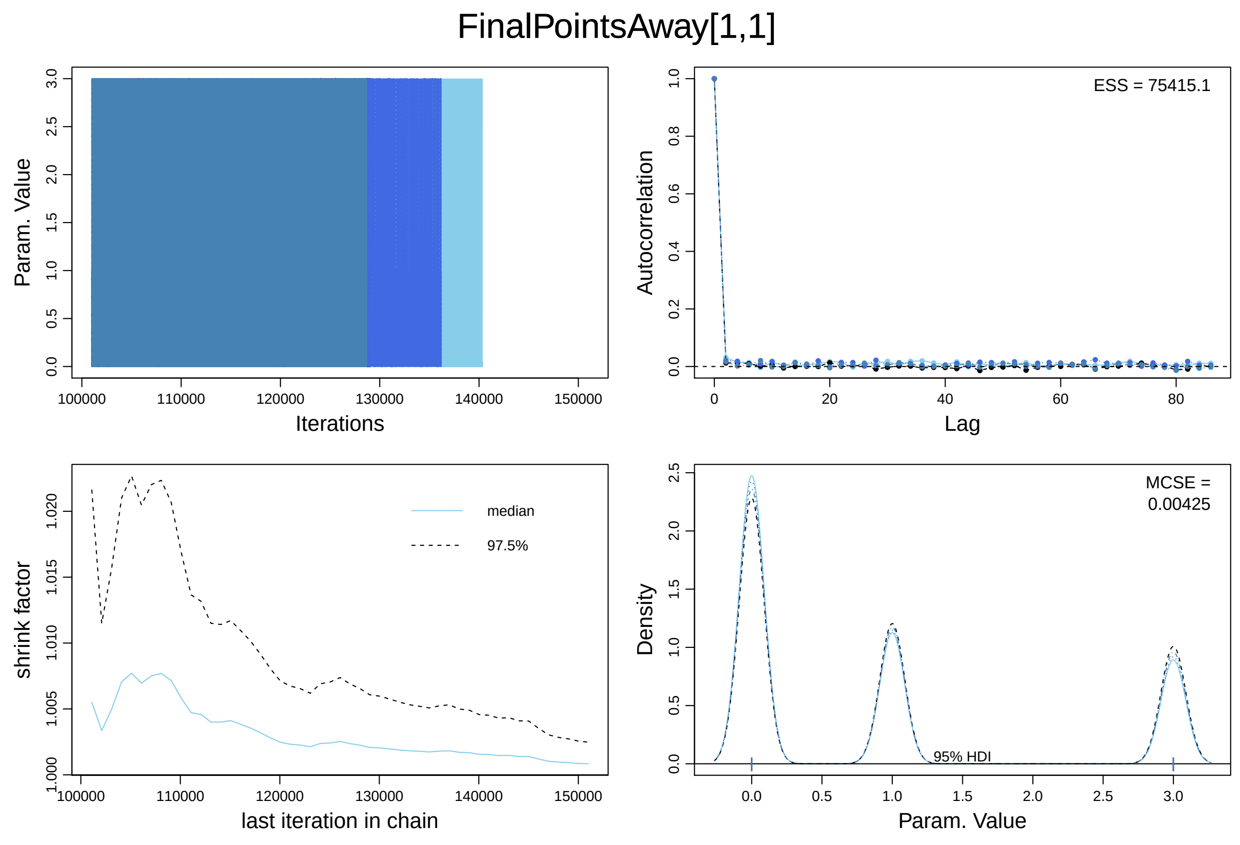
```
# -1 Away win, 0 tie, 1 Home win
teams <- unique(c(epl$Home.Team, epl$Away.Team))

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
    }
  }
}

fileName <- "Final.6.c.jags"

modelString ="
model {
  for(i in 1:120) {
    H[i] ~ dnegbin(ph[i], rh[i])
    A[i] ~ dnegbin(pa[i], ra[i])

    ph[i] <- rh[i] / (rh[i] + lambda[HomeTeam[i], AwayTeam[i]])
    pa[i] <- ra[i] / (ra[i] + theta[HomeTeam[i], AwayTeam[i]])

    rh[i] ~ dunif(0, 50)
    ra[i] ~ dunif(0, 50)
  }

  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(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(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: 1083
##    Total graph size: 27679
##
## Initializing model
```

```
update(epl.model, n.iter=100000)

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


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

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



(d) Run the Negative Binomial model for the EPL data, and answer the same questions as posed in Question 4 (b) through (f) and Question 5 (h) through (l). Are your answers any different?

4 (b) Using the posterior means as a judge, which is the best offensive team? Which is the worst offensive team? Which is the best defensive team? Which is the worst defensive team?

```
final.points.away.means <- smry$quantiles[1:400, 3]
final.points.home.means <- smry$quantiles[401:800, 3]
final.points.away.means <- smry$quantiles[1:400, 3]

league.rank.means <- smry$quantiles[801:820, 3]
league.ranks.means <- smry$quantiles[821:1220, 3]

sim.goals.away.means <- smry$quantiles[1221:1620, 3]
sim.goals.home.means <- smry$quantiles[1621:2020, 3]

sim.points.away.means <- smry$quantiles[2021:2420, 3]
sim.points.home.means <- smry$quantiles[2421:2820, 3]

total.points.means <- smry$quantiles[2821:2840, 3]


a.means <- smry$quantiles[2841:2860, 3]
d.means <- smry$quantiles[2881:2880, 3]

gamma.means <- smry$quantiles[2881, 3]

rank.a.means <- smry$quantiles[3242:3261, 3]
rank.d.means <- smry$quantiles[3262:3281, 3]

ranks.a.means <- smry$quantiles[3282:3681, 3]
ranks.d.means <- smry$quantiles[3682:4081, 3]

best.a <- which(a.means == max(a.means))
worst.a <- which(a.means == min(a.means))

best.d <- which(d.means == max(d.means))
worst.d <- which(d.means == min(d.means))

best.a.team.name <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == best.a])
best.d.team.name <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == best.d])

worst.a.team.name <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == worst.a])
worst.d.team.name <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == worst.d])
```

The best offensive team is Liverpool, and the worst offensive team is Hull City.

The best defensive team is Arsenal, and the worst defensive team is Burnley.

■

4 (c) Rather than using the posterior means to evaluate the offensive and defensive quality of the teams, we could look at where each team's α and δ values rank compared to all the other teams. In JAGS, the rank function takes a vector and returns another vector whose entries are the size ranks of the the elements of the input vector, with 1 indicating that this was the smallest element, 2 indicating the next smallest, and so on.

```
best.rank <- which(rank.d.means == max(rank.d.means))
worst.rank <- which(rank.d.means == min(rank.d.means))

best.team.name.rank1 <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == best.rank[[
worst.team.name.rank <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == worst.rank[
```

The best defensive team by rank is Tottenham Hotspur, and the worst defensive team by rank is West Ham United.

■

(d) In your JAGS model, add a line defining a vector of parameters called `rank.a` so that the kth element of `rank.a`, `rank.a[k]` is the rank of team k's offensive strength compared to all other teams, with 1 indicating they were the best offensive team (α_k is the biggest of all α's), and 20 indicating that they were the worst offensive team (α_k is the smallest of all α's). Using the posterior mean ranks for each team, which is the best offensive team? Which is the worst offensive team?

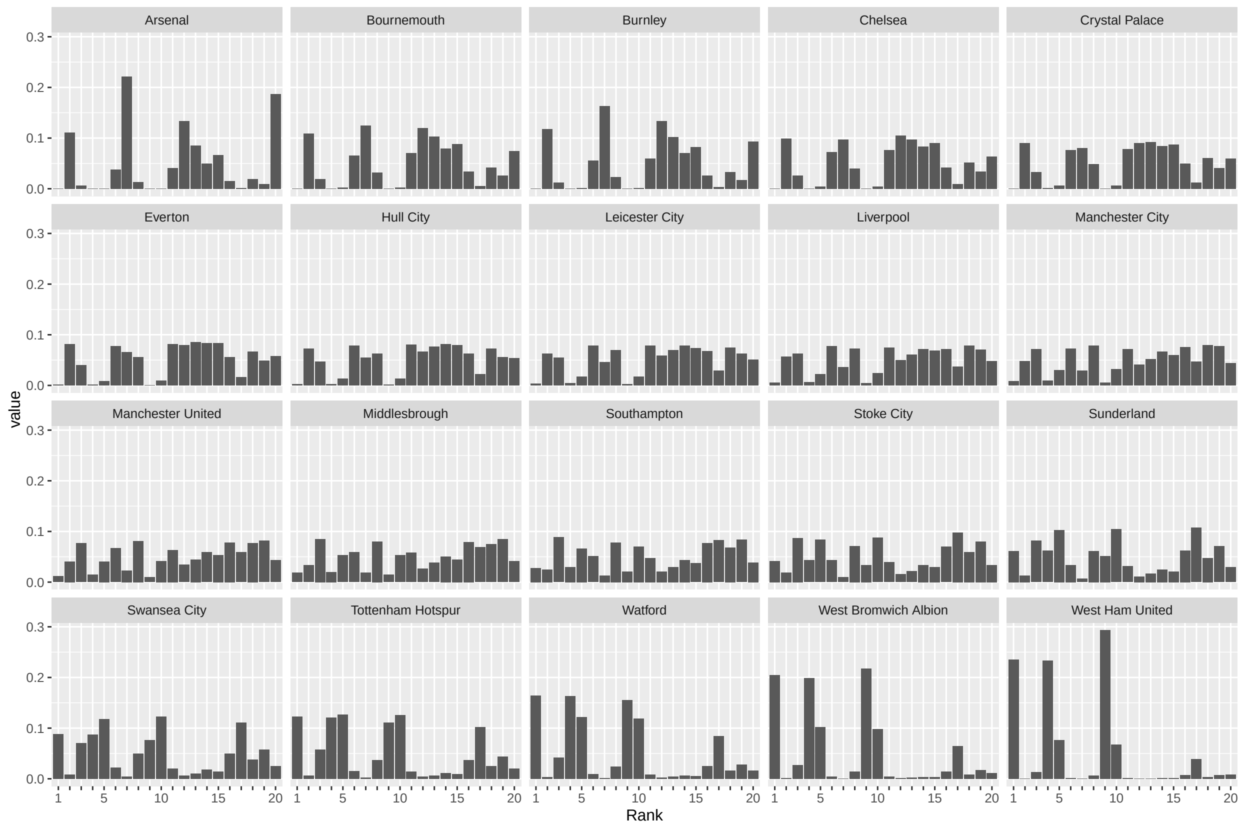
```
best.rank <- which(rank.a.means == max(rank.a.means))
worst.rank <- which(rank.a.means == min(rank.a.means))

best.team.name.rank1 <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == best.rank[[
worst.team.name.rank <- as.character(teams.with.names$Home.Team[teams.with.names$Home.ID == worst.rank[
```

The best defensive team by rank is Liverpool, and the worst defensive team by rank is Hull City.

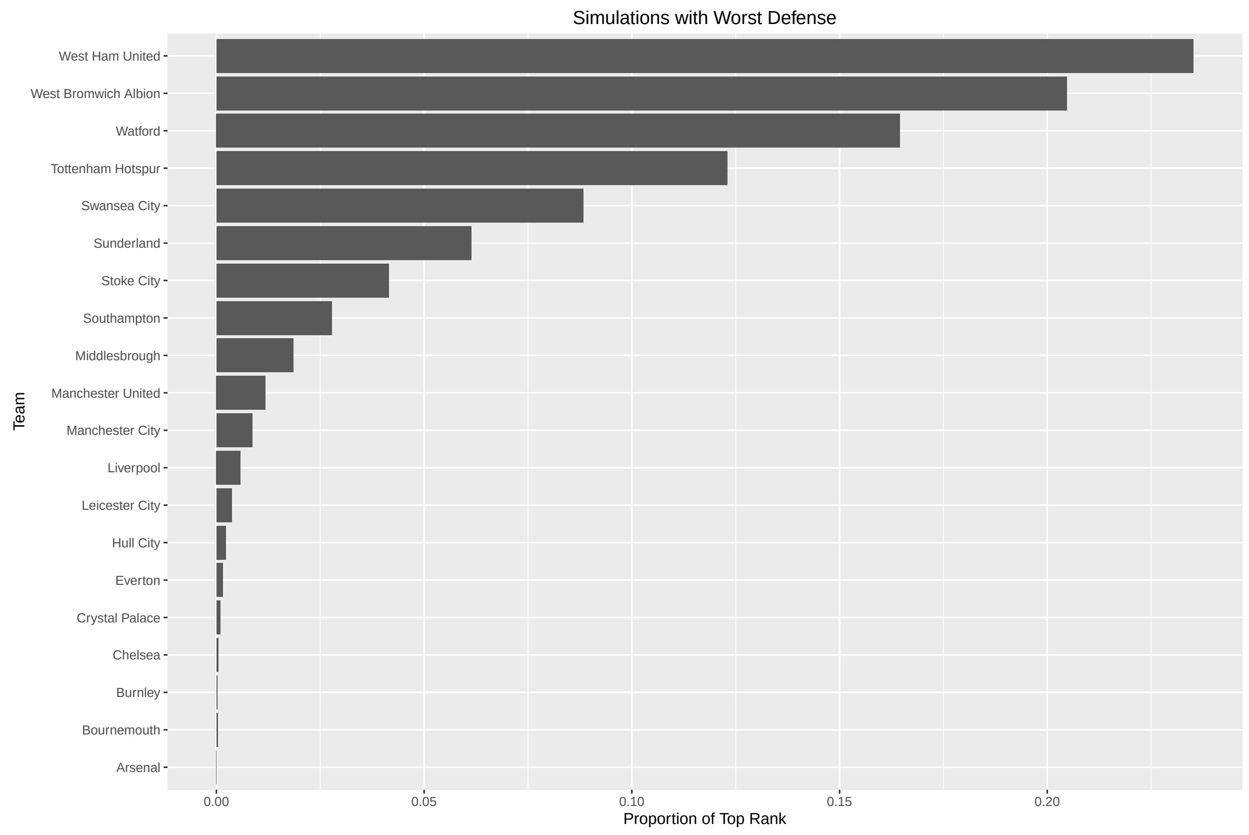
■

(e) Yet another way to evaluate the teams, would be to see in what proportion of simulations were the team's parameters ranked in a particular spot. In your JAGS model, add code to define `ranks.d[k,m]` which takes the value 1 is the defensive rank of team k is equal to m, and 0 otherwise (hint, check out the equals function in JAGS). This should be done for all teams and all possible ranks. With `ranks.d[k,m]` defined this way, the posterior mean of `ranks.d[k,m]` would be equal to the proportion of simulations in which team k's defensive rank was equal to m. Which team had the highest proportion of rank 1 for the defensive parameter? Which team had the highest proportion of rank 20 for the defensive parameter?

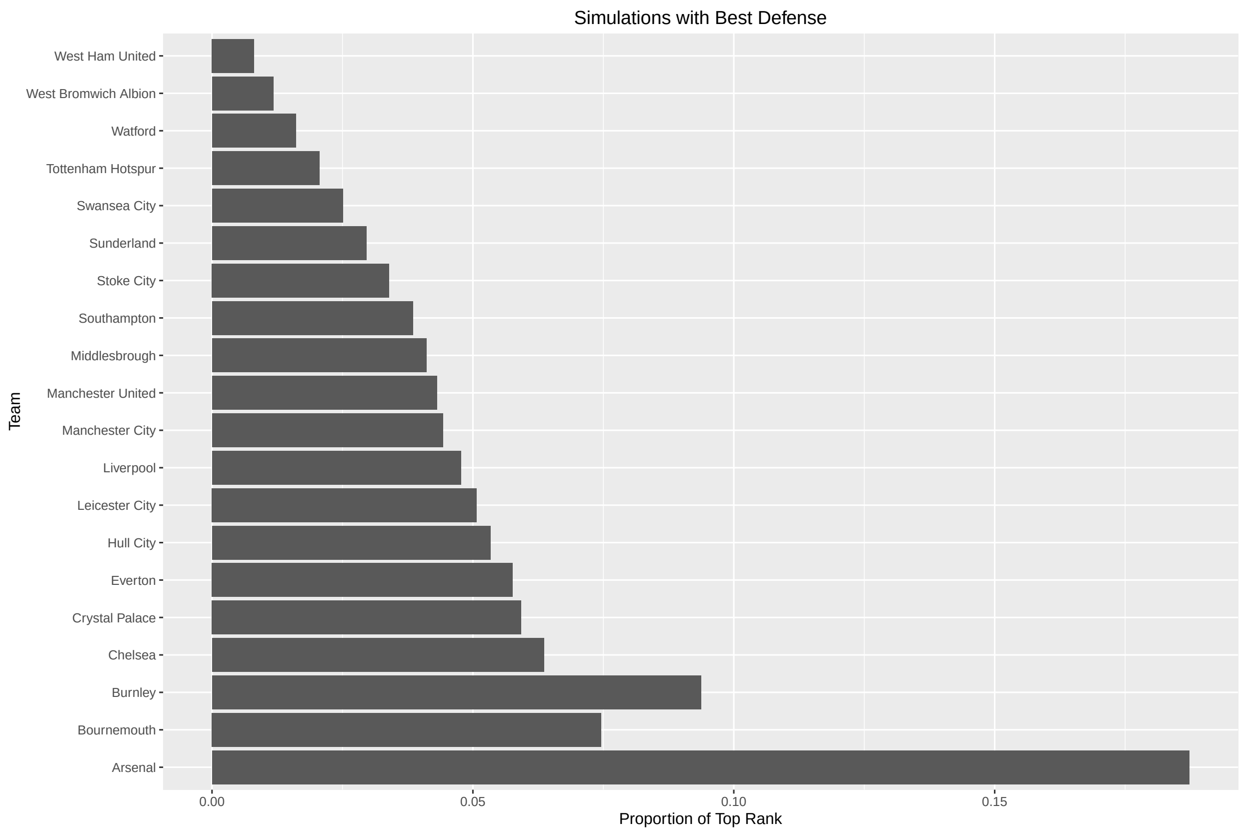
```r
ranks.df <- data.frame(teams.with.names)
ranks.df$rank1 <- 0
ranks.df$rank2 <- 0
ranks.df$rank3 <- 0
ranks.df$rank4 <- 0
ranks.df$rank5 <- 0
ranks.df$rank6 <- 0
ranks.df$rank7 <- 0
ranks.df$rank8 <- 0
ranks.df$rank9 <- 0
ranks.df$rank10 <- 0
ranks.df$rank11 <- 0
ranks.df$rank12 <- 0
ranks.df$rank13 <- 0
ranks.df$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("ranks.a[", i, ",", j, "]")])
  }
}
ranks.sub.df <- ranks.df[,-1]

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)

ranks.sub.df.rank1$Home.Team <-
  factor(ranks.sub.df.rank1$Home.Team, levels(ranks.sub.df.rank1$Home.Team[order(ranks.sub.df.rank1$ranl

ranks.sub.df.rank20$Home.Team <-
  factor(ranks.sub.df.rank20$Home.Team, levels(ranks.sub.df.rank20$Home.Team[order(-ranks.sub.df.rank20$

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



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

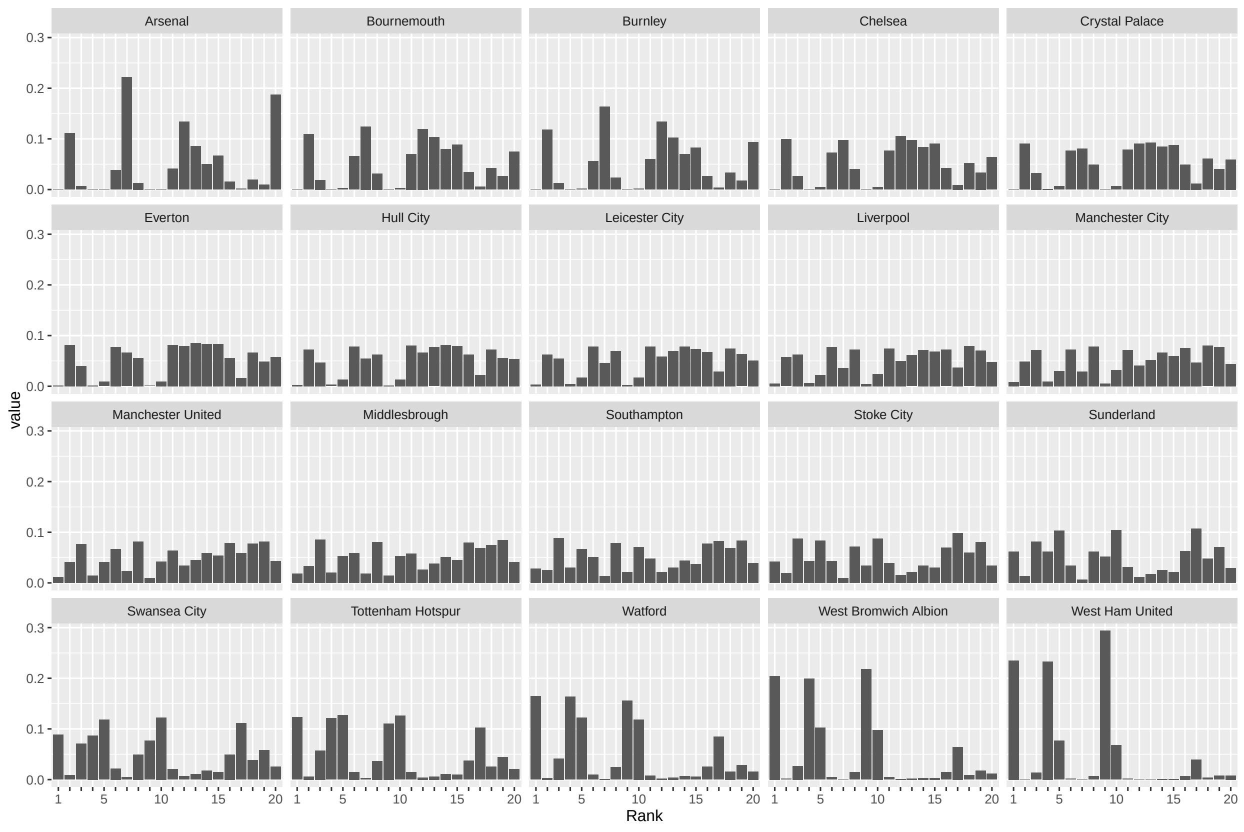


■

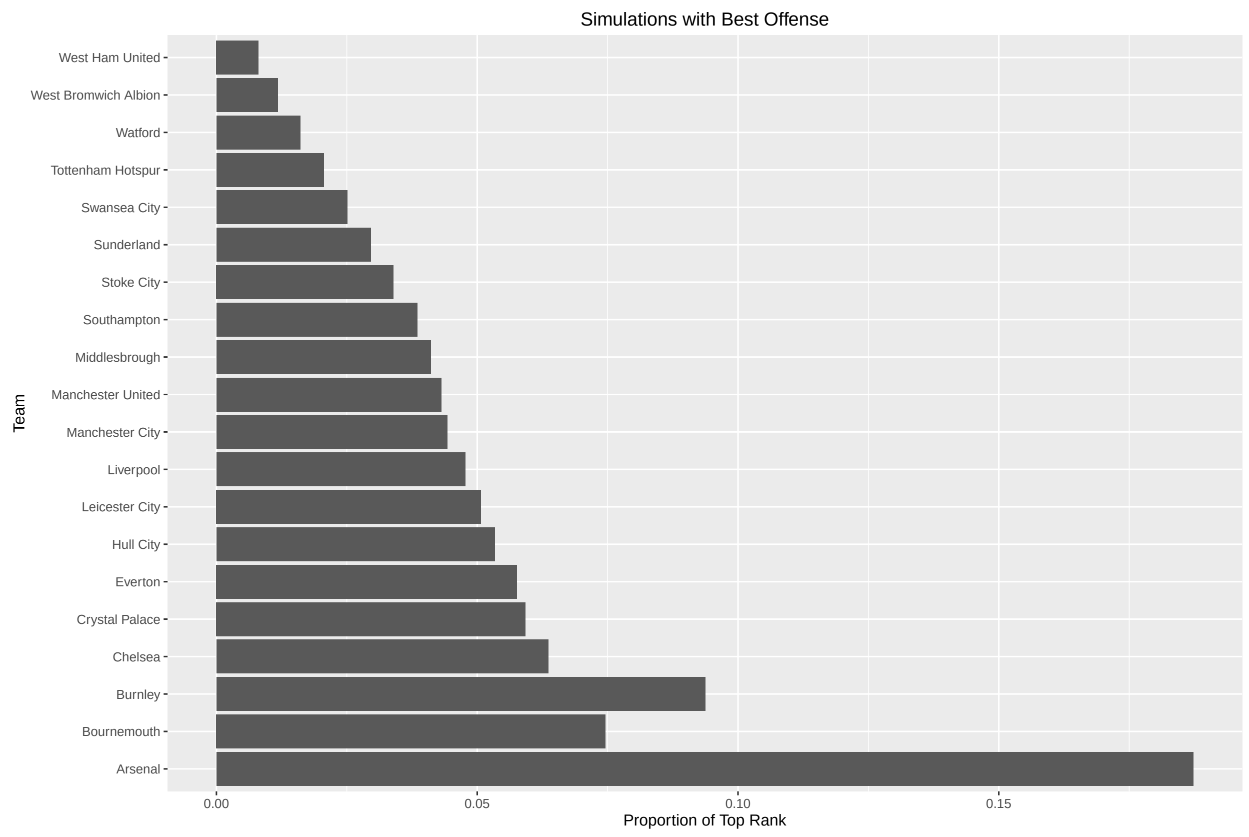
(f) Repeat part (e) for the offensive parameter, defining `ranks.a[k.m]`.

Show all working.

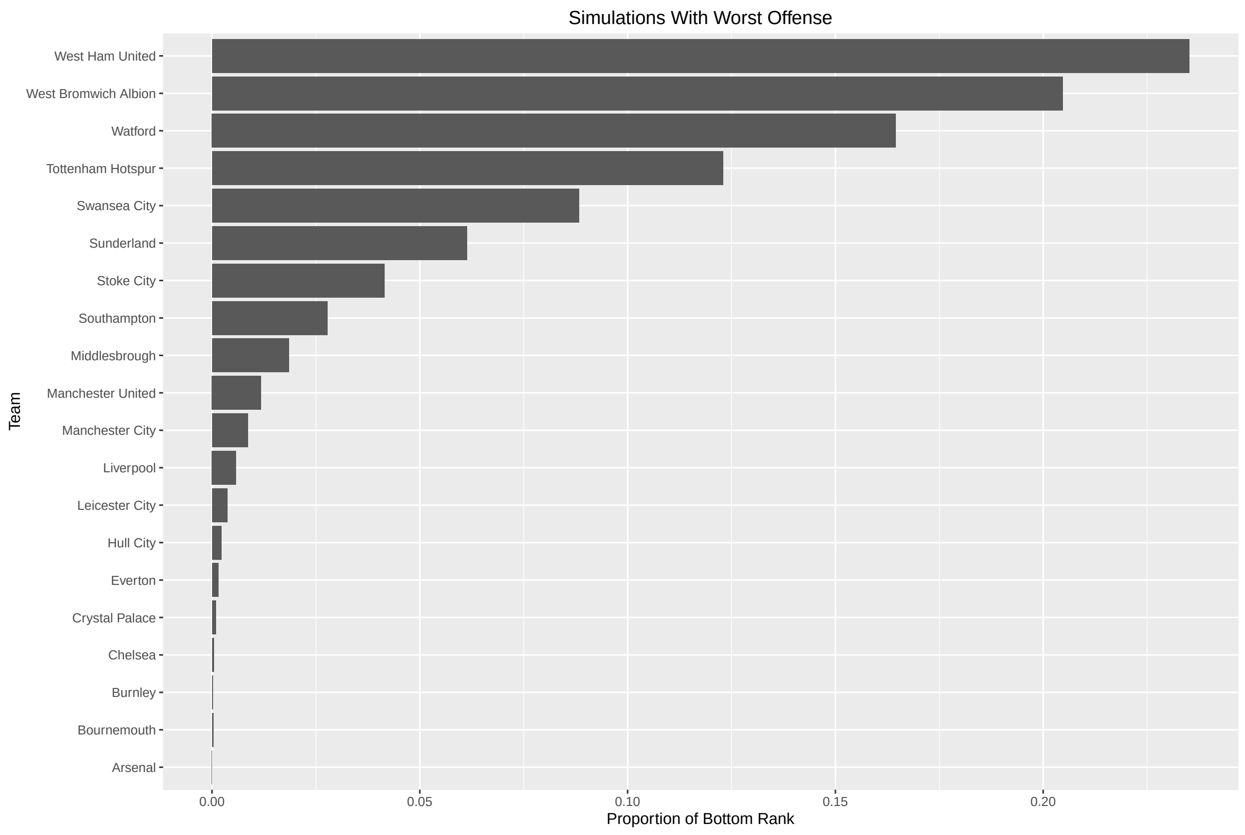
```
ranks.df <- data.frame(teams.with.names)
ranks.df$rank1 <- 0
ranks.df$rank2 <- 0
ranks.df$rank3 <- 0
ranks.df$rank4 <- 0
ranks.df$rank5 <- 0
ranks.df$rank6 <- 0
ranks.df$rank7 <- 0
ranks.df$rank8 <- 0
ranks.df$rank9 <- 0
ranks.df$rank10 <- 0
ranks.df$rank11 <- 0
ranks.df$rank12 <- 0
ranks.df$rank13 <- 0
ranks.df$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("ranks.a[", i, ",", j, "]")])
  }
}
ranks.sub.df <- ranks.df[,-1]

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

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



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



■

5 (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.

```r
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?

```r
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))
```

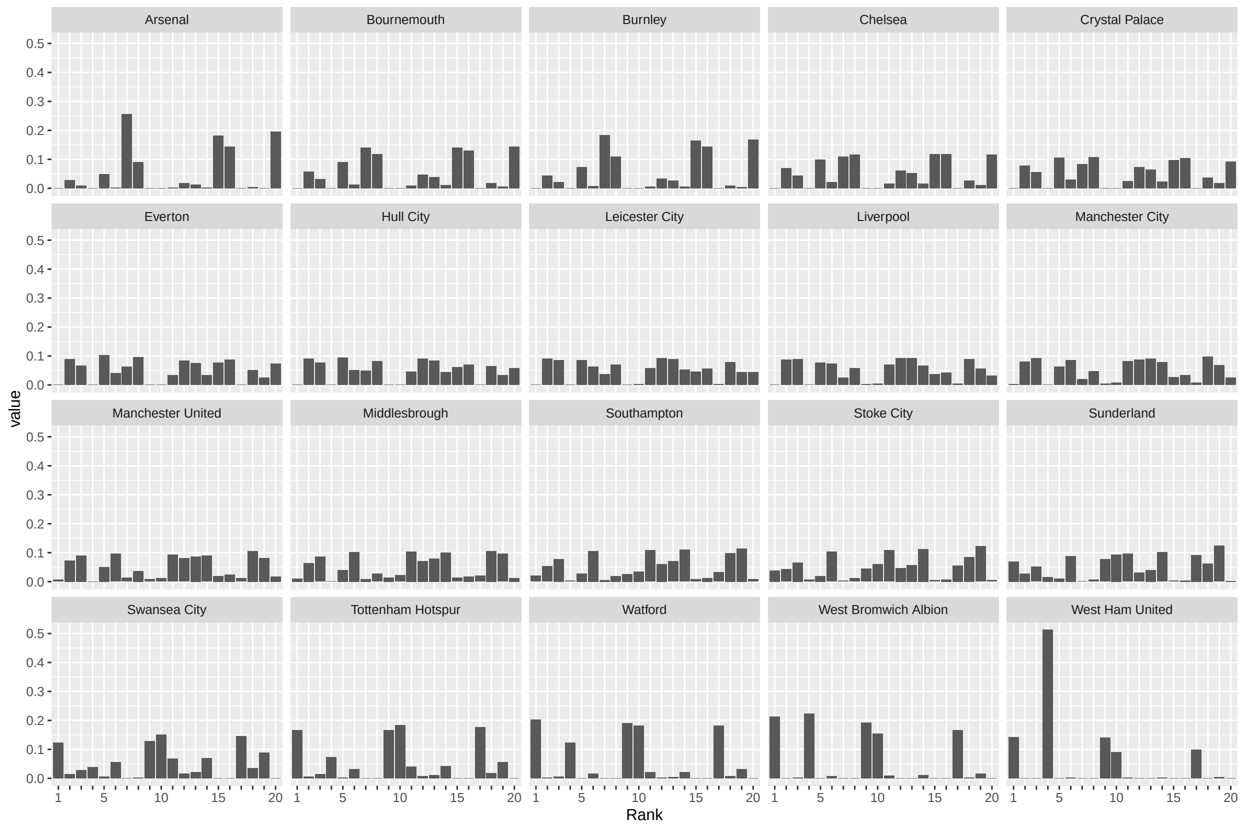
The top four teams ranked by final points are Chelsea, Arsenal, Liverpool, Manchester City.

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

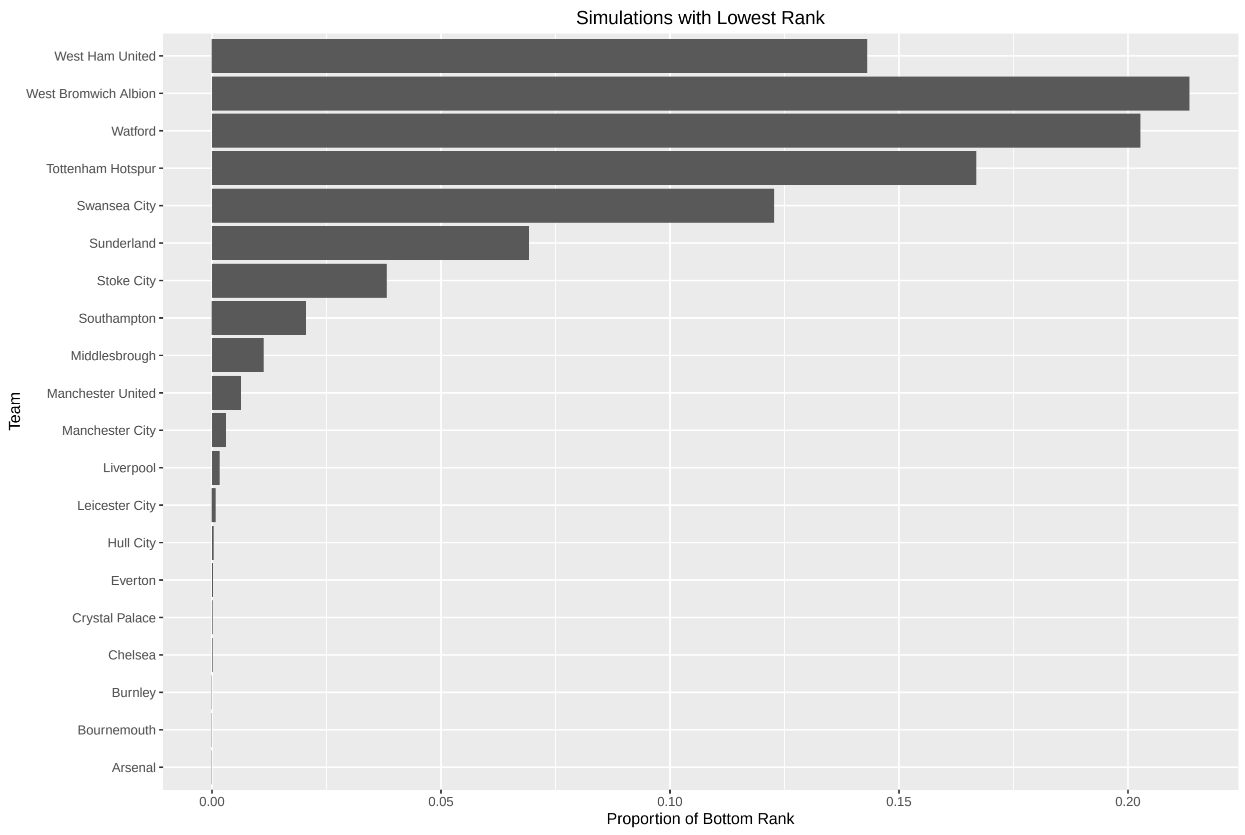
(j) Just like in the previous question, we could look to see in what proportion of simulations was a team's point total ranked in a particular spot. In your JAGS model, add code to define `LeagueRanks[k,m]` which takes value 1 if the rank of the points total for team k is equal to m and 0 otherwise.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

```r
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
```

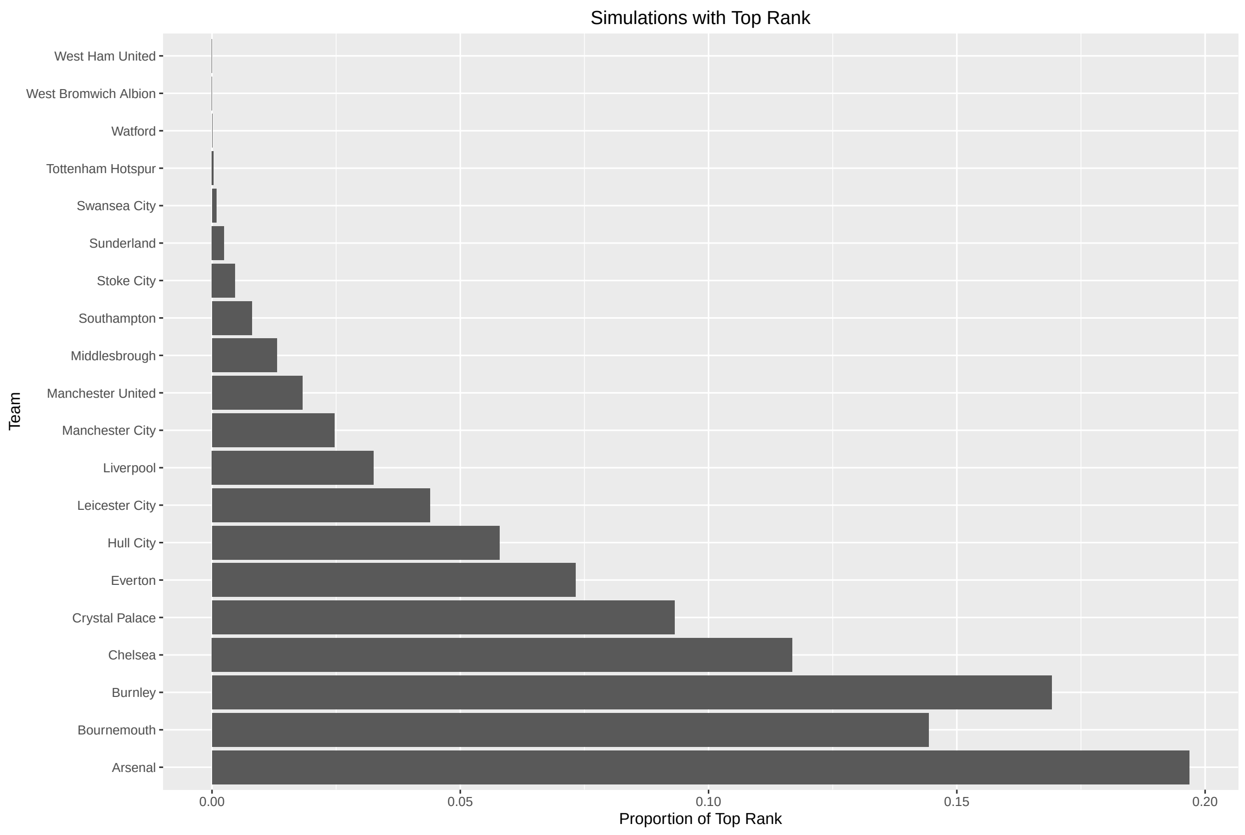
```
for (i in 1:20) {
  for (j in 1:20) {
    ranks.df[i, paste0("rank", j)] <- mean(epl.samples.M[,paste0("LeagueRanks[", i, ",", j, "]")])
  }
}
ranks.sub.df <- ranks.df[,-1]
ranks.sub.df <- ranks.sub.df[, -2]
ranks.sub.df <- ranks.sub.df[, -2]

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.

```r
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     0.28072
## 2 Swansea City     0.27300
## 3   Stoke City     0.26756
## 4  Southampton     0.25431
```

(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.

```r
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        0.21603
## 2              Watford        0.21166
## 3    Tottenham Hotspur        0.18759
```

References

[1] Jackman, Simon *Bayesian Analysis for the Social Sciences*
John Wiley & Sons, Ltd 2009

■