A Bayesian Approach to Predicting NFL Quarterback Scores in Fanduel Tournaments

*STAT 578, Fall 2017, **Team 5**: Aaron Ray, Kiomars Nassiri, Michael Chan*

October 25, 2017

Project Description

The National Football League (NFL), being one of the major professional sports leagues in North America, has a wide audience. participates in the NFL craze by competing in fantasy football tournaments organized by the daily fantasy site, "FanDuel.com". Participants in a **Fantasy Football** game act as the managers of a virtual footbal team and try to maximize their points by picking up the best line-up. Points are given based on actual performance of players in real-world competition. For the purpose of this project we have chosen to work with the data gathered from the **FanDuel** internet company. We will leverage a Hierarchical Bayesian approach with the Markov Chain Monte Carlo method to predict the fantasy points likely to be scored by an NFL quarterback in any given game. The goal is to predict the points scored by each player given certain prior conditions and predictor variables that will assist our model in providing credible posterior prediction intervals.

The analysis is inspired by the study presented in the article, **Bayesian Hierarchical Modeling Applied to Fantasy Football Projections for Increased Insight and Confidence**, by Scott Rome.

Team Members

- **Aaron Ray** (aaronwr2@illinois.edu)*
- **Kiomars Nassiri** (nassiri2@illinois.edu)
- **Michael Chan** (mhchan3@illinois.edu)

*Contact Person

Dataset Description

Team has set up a process to gather the historical data from the RotoGuru website. The following is the code used to get the data from RotoGuru:

```
# Scrape rotoguru1 site for weekly FanDuel stats and bind each week's data to the
# pre-defined dataframe, 'd'.

for(year in 2014:2017){
  for(week in 1:16){
    page = read_html(
      gsub(" ","",
           paste("http://rotoguru1.com/cgi-bin/fyday.pl?week=",week,"&year=",
                 year ,"&game=fd&scsv=1")
      ))
    dtext = page %>% html_nodes("pre") %>% html_text(trim = TRUE)
    dtable = read.table(text=dtext, sep = ";", header=TRUE, col.names = cnames,
```

```
                              quote=NULL)
    d = rbind(d,dtable)
  }
}
```

Data cleaning is performed using R routines. Some data cleaning tasks are needed to calculate Player rank.

Response Variables

- `FanDuelPts`: Points position at the end of a single game

Predictor Variables

- `6GmAvgOppPAP`: The six game average Opposing Points Allowed to Position (OppPAP) by the current player's opposing defense. For example, if the Buffalo Bills defense allowed a total of 30 points per game to wide receivers for six games straight, then this number would equal to the average of 30 for any wide receiver facing the Bills defense.
- `Position`: The position the player plays
- `Opponent`: The team that the player plays against
- `Rank`: The rank of a player based on recent performance

Analysis Ideas

At the lowest level, we model the performance (`FanDuelPts`) as normally-distributed around a true value:

$y|\alpha, \beta_{defense}, \beta_{home}, \beta_{away}, \sigma_r^2 \sim N(\alpha + X_{defense}.\beta_{defense} + X_{home}.\beta_{home} + X_{away}.\beta_{away}, \sigma_y^2 I)$

where

$\alpha =$ `6GmAvgOppPAP`

$\beta_{defense,t,p} =$ defense coefficient against team t for position p

$\beta_{home,p,r} =$ home coefficient for position p and a rank r player

$\beta_{away,p,r} =$ Away coefficient for position p and a rank r player

$y =$ `FanDuelPts`

$x_{t,p} =$ interaction indicator term for team t, position p

$x_{p,r} =$ interaction indicator term for rank r, position p

At higher level, we model the defense effect, $\beta_{defense}$, as how good a player is when playing against a particular team. We pool the effect based on the position of the player. That is, the defense coefficient is normally distributed from the same position specific distribution.

$\beta_{defense,t,p} \sim N(\delta_p, \sigma_\delta^2)$

For the home and away game effect, β_{home} and β_{away}, we model the effect for player of the same rank has the same distribution.

$\beta_{home,r} \sim N(\eta_r, \sigma_\eta^2)$

$\beta_{away,r} \sim N(\rho_r, \sigma_\rho^2)$

We will approximate non informative prior using:

$\sigma_y \sim Inv - gamma(0.0001, 0.0001)$

$\sigma_\delta \sim N(0, 100^2)$

$\sigma_\eta \sim N(0, 100^2)$

$\sigma_\rho \sim N(0, 100^2)$

Here is the JAGS model:

```
model {
  for (i in 1:length(y)) {
    y[i] ~ dnorm(inprod(X.defense[i, ], beta.defense)
                 + inprod(X.home[i, ], beta.home)
                 + inprod(X.away[i, ], beta.away), sigmasqinv)
  }

  # The entry of the beta.defense corresponds to Opponent:Position
  # In our model, we pool the beta.defense based on position.
  # i.e. All defense effects of the same position are drawn from the same distribution
  for (p in 1:Num.Position) {
    for (t in 1:Num.Opponent) {
      beta.defense[(p-1) * Num.Opponent + t] ~ dnorm(delta[p], 1/100^2)
    }
    delta[p] ~ dnorm(0, 1/100^2)
  }

  # The entry of the beta.home and beta.away corresponds to Rank:Position
  # In our model, we pool the beta.home/away based on rank
  for (r in 1:Num.Rank) {
    for (t in 1:Num.Position) {
      beta.home[(r-1) * Num.Position + t] ~ dnorm(eta[r], 1/100^2)
      beta.away[(r-1) * Num.Position + t] ~ dnorm(rho[r], 1/100^2)
    }
    eta[r] ~ dnorm(0, 1/100^2)
    rho[r] ~ dnorm(0, 1/100^2)
  }

  sigmasqinv ~ dgamma(0.0001, 0.0001)
  sigmasq <- 1/sigmasqinv
}
```

Sample Data

```
fdp <- read.csv("fdp.csv", sep = '\t', header = TRUE)

head(fdp)
```

```
##   Week Year TeamGameId PlayerId             Name Position  Team Home.Game
## 1   16 2015        200     1060 Hasselbeck, Matt       QB Colts         0
## 2   15 2015        184     1060 Hasselbeck, Matt       QB Colts         1
## 3   14 2015        134     1060 Hasselbeck, Matt       QB Colts         0
## 4   13 2015        112     1060 Hasselbeck, Matt       QB Colts         0
```

```
## 5   12 2015         85     1060 Hasselbeck, Matt       QB Colts         1
## 6   11 2015         30     1060 Hasselbeck, Matt       QB Colts         0
##     Opponent OpponentId X6GmAvgOppPAP X6GmStDOppPAP FanDuelPts FanDuelSalary
## 1   Dolphins       7016          21.2           5.6       3.96          6000
## 2     Texans       7032          14.5           8.3       8.98          6400
## 3    Jaguars       7014          24.2           6.7       9.08          6600
## 4   Steelers       7024          20.8           8.0       6.86          6500
## 5 Buccaneers       7029          19.7           8.3      20.30          6400
## 6    Falcons       7002          17.5           7.2      15.32          6300
```

Derived data

```r
#Rank player based on current FanDuelPts

#(This is cheating as it is looking at future, we'll fix it after more data
# clean up work - This is done only for getting some data to play with in this
# preminlinary research)

year_week = unique(fdp[,c('Year','Week')])
rank_column = "FanDuelPts"
fdp['Rank'] = NA
for (i in 1:nrow(year_week)) {
  fdp_year_week = fdp[fdp$Year == year_week[i, 'Year'] & fdp$Week == year_week[i, 'Week'], ]
  fdp_year_week_quantile = quantile(fdp_year_week[rank_column], c(0.25, 0.5, 0.75), na.rm = TRUE)

  fdp[fdp$Year == year_week[i, 'Year'] & fdp$Week == year_week[i, 'Week']
      & fdp[rank_column] < fdp_year_week_quantile[1], 'Rank'] = 'Rank4'
  fdp[fdp$Year == year_week[i, 'Year'] & fdp$Week == year_week[i, 'Week']
      & fdp[rank_column] >= fdp_year_week_quantile[1]
      & fdp[rank_column] < fdp_year_week_quantile[2], 'Rank'] = 'Rank3'
  fdp[fdp$Year == year_week[i, 'Year'] & fdp$Week == year_week[i, 'Week']
      & fdp[rank_column] >= fdp_year_week_quantile[2]
      & fdp[rank_column] < fdp_year_week_quantile[3], 'Rank'] = 'Rank2'
  fdp[fdp$Year == year_week[i, 'Year'] & fdp$Week == year_week[i, 'Week']
      & fdp[rank_column] >= fdp_year_week_quantile[3], 'Rank'] = 'Rank1'
}
```

```r
fdp['Locality'] = 'Away'
fdp[fdp$Home.Game == 1, 'Locality'] = 'Home'
```

** The following R code is for reference and preliminary research **

```r
fdp_train=fdp[fdp$Year == 2016, ]

X.defense = model.matrix(~ 0 + Opponent:Position , data=fdp_train)
X.home = model.matrix(~ 0 + Rank:Position , data=fdp_train)
X.away = model.matrix(~ 0 + Rank:Position , data=fdp_train)

X = cbind(X.defense, X.home, X.away)

Num.Opponent = length(unique(fdp[, "Opponent"]))
Num.Position = length(unique(fdp[, "Position"]))
Num.Rank = length(unique(fdp[, "Rank"]))
```

```r
library(rjags)
set.seed(20171008)
```

```r
# Initialization List for the 4 chains
jags.inits=list(
  list( sigmasqinv=    0.01,
        .RNG.name = "base::Mersenne-Twister", .RNG.seed = 20171008 ),
  list( sigmasqinv=    0.01,
        .RNG.name = "base::Mersenne-Twister", .RNG.seed = 20171008 + 1 ),
  list( sigmasqinv=0.000001,
        .RNG.name = "base::Mersenne-Twister", .RNG.seed = 20171008 + 2 ),
  list( sigmasqinv=0.000001,
        .RNG.name = "base::Mersenne-Twister", .RNG.seed = 20171008 + 3 )
)

data.jags <- list(
  y= fdp_train$FanDuelPts,
  X.defense = X.defense,
  X.home = X.home,
  X.away = X.away,
  Num.Position=Num.Position,
  Num.Opponent=Num.Opponent,
  Num.Rank=Num.Rank
)
```

```r
m <- jags.model("fdp.bug", data.jags, inits = jags.inits, n.chains=4, n.adapt = 1000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7207
##    Unobserved stochastic nodes: 255
##    Total graph size: 1759735
##
## Initializing model
```

```r
update(m, 2500) # burn-in

x <- coda.samples(m1, c("delta","beta.defense", "sigmasq"), n.iter=5000)
```