Model 3 Neg Binom

Ben Moolman, Craig Orman, Ethan Pross

Beginning Data preparing

```
## Ben's DRTG code!
# Calculate total points per team per game
# here, datatest2 is the entire data frame that is not filtered for starters
# filtering dataset to remove NAs which arise a player doesnt record any minutes in the game (sitting o
team_points <- na.omit(original_tbl) %>%
  group_by(GAME_ID, TEAM_ID) %>%
  summarize(TeamPoints = sum(PTS), .groups = "drop")

#
team_points_opponent <- team_points %>%
  rename(OPP_TEAM_ID = TEAM_ID, OpponentPoints = TeamPoints)

# join and filter
team_vs_opponent <- team_points %>%
  inner_join(team_points_opponent, by = "GAME_ID") %>%
  filter(TEAM_ID != OPP_TEAM_ID)

# calculate average opponent points per team (our DRTG)
team_drtg <- team_vs_opponent %>%
  group_by(TEAM_ID) %>%
  summarize(DRTG_proxy = mean(OpponentPoints), n_games = n(), .groups = "drop")

range(team_drtg$DRTG_proxy)
```

```
## [1] 106.5244 123.0366
```

```
mean(team_drtg$DRTG_proxy)
```

```
## [1] 114.2114
```

DRTG data being joined to the starting data

```
## NOW ADDING DRTG vars to original_tbl


# Each team plays one opponent per game, so we pair them like this:
game_team_pairs <- original_tbl %>%
  select(GAME_ID, TEAM_ID) %>%
  distinct()
```

```r
# Join to get each game twice: once for each team, with their opponent
opponent_map <- game_team_pairs %>%
  inner_join(game_team_pairs, by = "GAME_ID") %>%
  filter(TEAM_ID.x != TEAM_ID.y) %>%
  rename(TEAM_ID = TEAM_ID.x, OPP_TEAM_ID = TEAM_ID.y)

# Add opponent DRTG to the mapping
opponent_map <- opponent_map %>%
  left_join(team_drtg %>% rename(OPP_TEAM_ID = TEAM_ID, OPP_DRTG = DRTG_proxy),
            by = "OPP_TEAM_ID")

# Join to add opponent DRTG column to the full data
starting_dat <- starting_dat %>%
  left_join(opponent_map, by = c("GAME_ID", "TEAM_ID"))

mean_drtg <- mean(team_drtg$DRTG_proxy)
starting_dat <- starting_dat %>%
  mutate(centered_OPP_DRTG = OPP_DRTG - mean_drtg)
```

Model 3 implementation

$$
\begin{aligned}
y_{ikj} &\sim Binom(n_{ikj}, p_{ik}) \\
p_{ik} &\sim \phi \times Beta(a, b) \\
n_{ijk} &\sim NegBinom(r, \theta) \\
r &\sim Gamma(2, 0.1) \\
\theta &\sim Beta(2, 2)
\end{aligned}
$$

Next we write out the full conditionals

$$
\begin{aligned}
y_{ikj}|... &\sim Bin(n_{ikj}, p_{ik}) \\
p_{ik}|... &\propto \phi p(y|p_{ik}, n_{ikj}) p(p_{ik}|a, b) \\
&\propto \min(1, Beta(\sum(y) + a, N - \sum(y) + b \qquad (1) \\
n_{ikj}|... &\propto p(y_{ikj}|p_{ik}, n_{ikj}) p(n_{ikj}|r, \theta) \\
&= \binom{n}{y} p^{\sum y} (1-p)^{N - \sum y} \frac{e^{-\sum y} \lambda^n}{n!} \qquad (2) \\
a, b|... &\propto p(p_{ik}|a, b) p(a, b) \\
&\propto \frac{1}{B(a, b)} p^{a-1} (1-p)^{b-1} \qquad (3) \\
r, \theta|... &\propto p(n_{ikj}|r, \theta) p(r, \theta) \\
&\text{for these I used the actual density functions}
\end{aligned}
$$

So for this, we kinda have a full thingy set up?

```r
# Warning: Produces lots of NaNs!
# Data and prior

# Initial values
lebron_dat = starting_dat[starting_dat$PLAYER_ID %in% 2544, ]
```

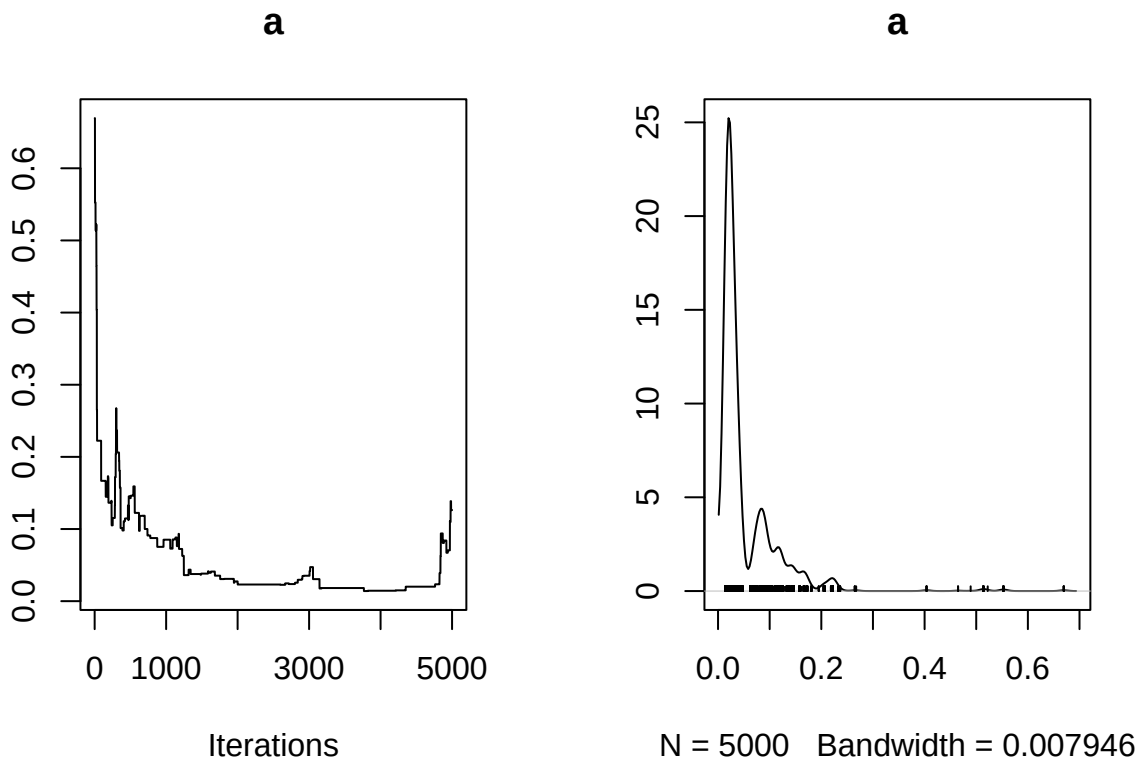
```r
y = lebron_dat$FGM
a = 1
b = 1
r = 20
theta = 0.5
eta = 2
DRTG = lebron_dat$centered_OPP_DRTG
psi=0.2

# Priors for r's gamma
Ga = 0.5
Gb = 0.1

# Priors for theta's beta
Ta = 2
Tb = 2


# Initial values for n's neg binom
n = 5
p = 0.1

# Save structures
n_iter = 5000
n_keep = matrix(NA, ncol=length(y), nrow=n_iter)
p_keep = matrix(NA, ncol=length(y), nrow=n_iter)
a_keep = numeric(n_iter)
b_keep = numeric(n_iter)
r_keep = numeric(n_iter)
theta_keep = numeric(n_iter)


# proposal variances (tuning paramaters)
prop_sd_a = 0.4
prop_sd_b = 0.4
prop_sd_eta = 0.4
prop_sd_r = 0.4

# full log conditionals

# a|... ~
log_fc_a = function(p, a, b) {
  if (a<0) return(-Inf)
  n = length(p)
  (a-1)*sum(log(p))-n*lbeta(a,b)-5/2*(a+b)
}
log_fc_b = function(p, a, b) {
  if (b<0) return(-Inf)
  n = length(p)
  (b-1)*sum(log(1-p))-n*lbeta(a,b)-5/2*(a+b)
}

log_fc_r = function(n, r, theta, Ga, Gb) {
```

```
  N = length(theta)
  return(sum(log(gamma(n+r))-log(r)) +
           r^(Ga-1)*N*log(theta)+log(r)-Gb*r)
}

log_fc_eta = function(N, r, theta, prop_sd_eta, eta) {
    sum(dnbinom(N, size=r, prob=theta, log=TRUE)) +
    dnorm(eta, mean=0, sd = prop_sd_eta, log=TRUE)
}


for (i in 1:n_iter){
  # generate p based on a and b, corresponds to option (1) above
  p = pmin(1, exp(psi * DRTG) * rbeta(length(y), a, b))

  #sample a from full conditional, corresponds to part (3) above
  a_prop = rnorm(1, a, prop_sd_a)
  logr = log_fc_a(p, a_prop, b) - log_fc_a(p, a, b)
  if (is.finite(logr) && log(runif(1)) < logr) {
      a <- a_prop
  }


  # sample b from full conditional, corresponds to part (3) above
  b_prop = rnorm(1, b, prop_sd_a) #take a proposal value
  logr = log_fc_b(p, a, b_prop)-log_fc_b(p, a, b) # this is what Dr. Niemi does!
  if (is.finite(logr) && log(runif(1)) < logr) {
      b <- b_prop
  } # basically, if its a real number, and it follows the symmetric MH thingy

  # generate n based on r and theta
  # Not sure if Im supposed to use the distribution of n or the conditional here...
  n = rnbinom(length(y), r, theta)


  # sample theta from full conditional using a weird logit function
  # This one is a train wreck
  eta_prop = rnorm(1, b, prop_sd_r)
  theta_prop = 1/(1+exp(-eta_prop))
  logr = log_fc_eta(length(y), r, theta_prop, prop_sd_eta, eta_prop) - log_fc_eta(length(y), r, theta, p
    if (is.finite(logr) && log(runif(1)) < logr) {
      eta <- eta_prop
      theta <- theta_prop
  }
  # sample r from full conditional
  # This apparently doesn't ever accept so its not actually doing anything
  r_prop = rnorm(1, b, prop_sd_r)
  logr = log_fc_r(n, r_prop, theta, Ga, Gb)-log_fc_r(n, r, theta, Ga, Gb)
  if (is.finite(logr) && log(runif(1)) < logr) {
      r <- r_prop
  }

  # Save our samples
```

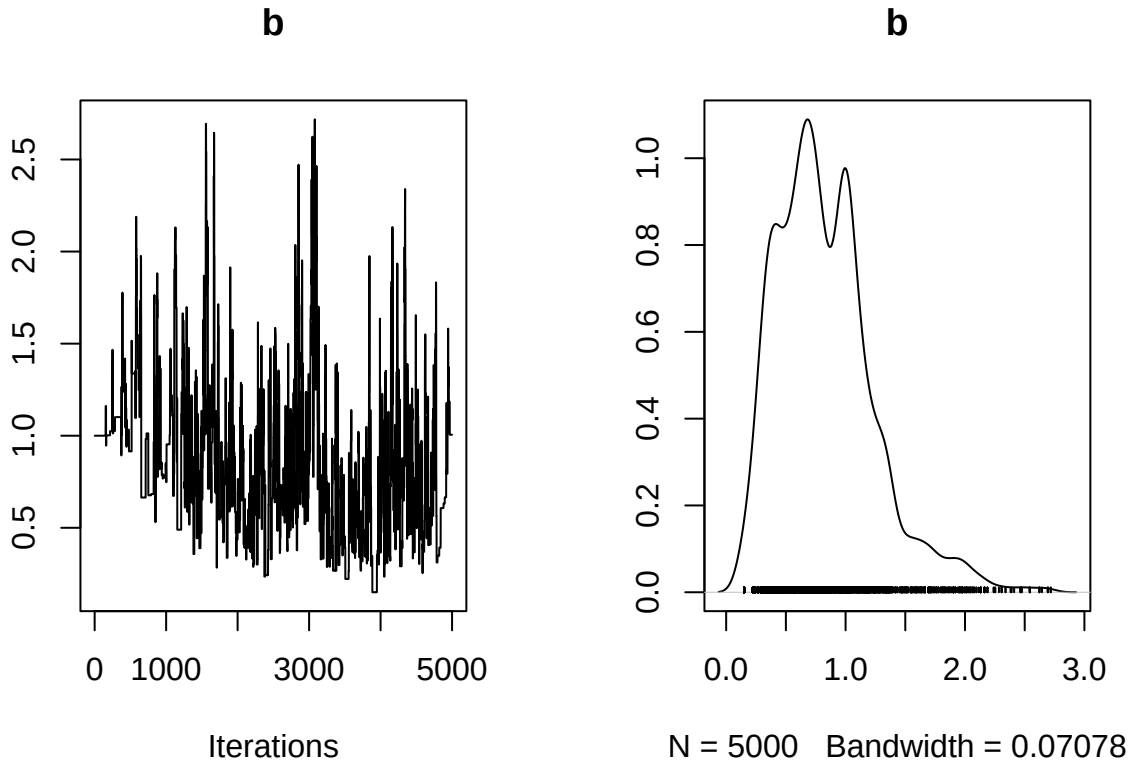
```
  n_keep[i,] = n
  p_keep[i,] = p
  a_keep[i] = a
  b_keep[i] = b
  r_keep[i] = r
  theta_keep[i] = theta
}
```

Some diagnostics to consider

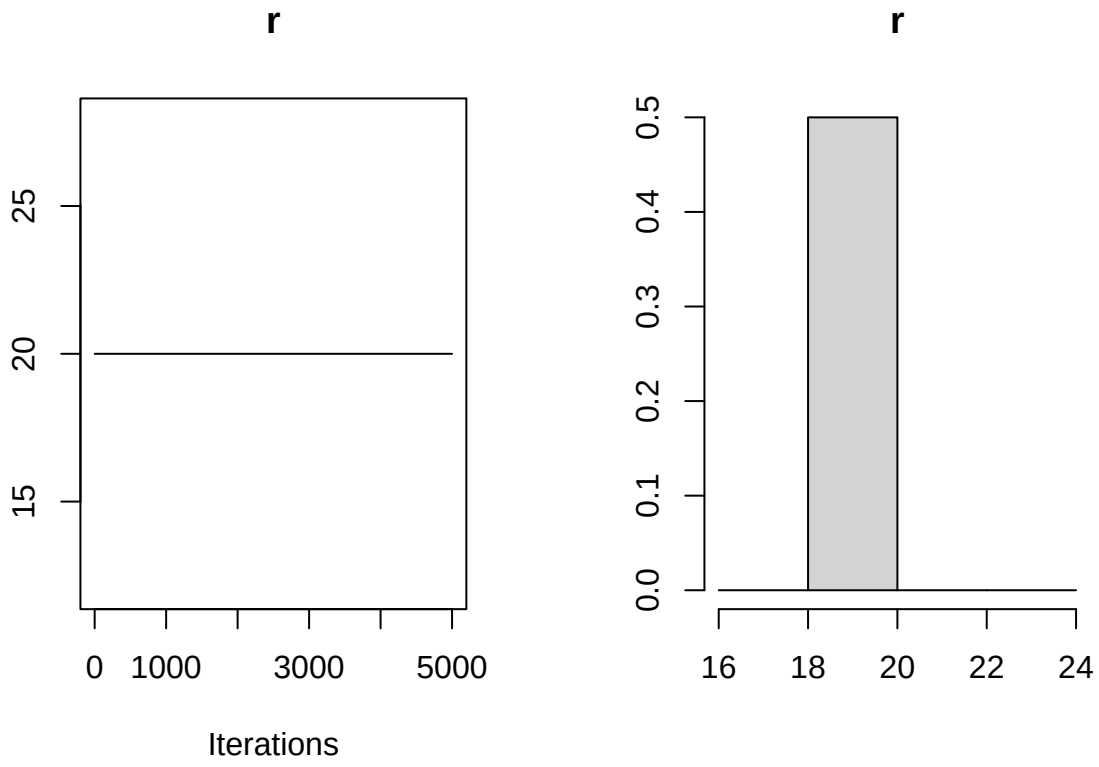
```
par(mfrow=c(4,2))
plot(as.mcmc(a_keep), main="a")
```



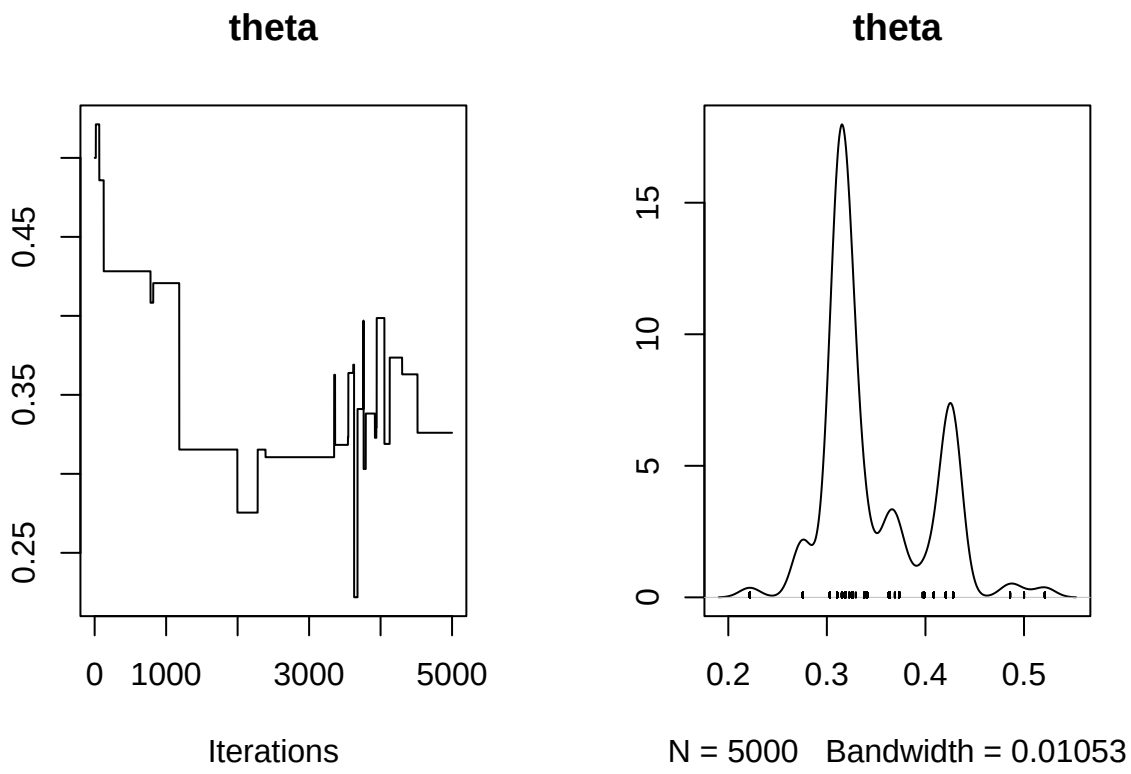

```
plot(as.mcmc(b_keep), main="b")
```




```
plot(as.mcmc(r_keep), main="r")
```

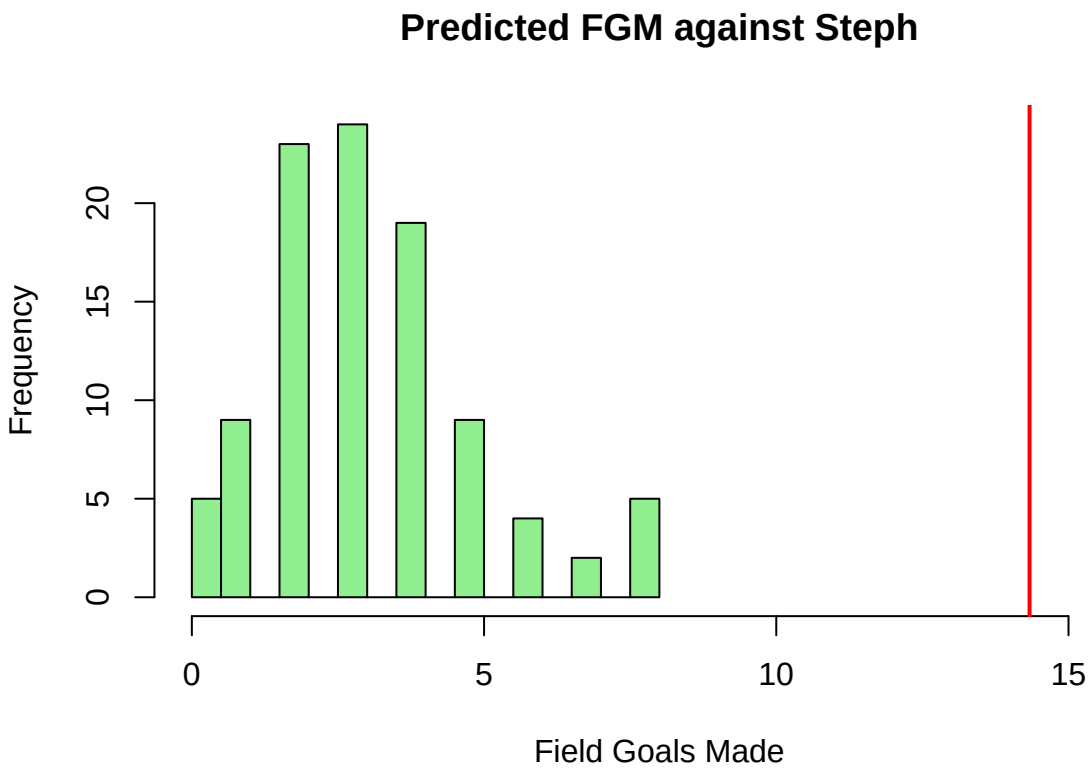



```
plot(as.mcmc(theta_keep), main="theta")
```

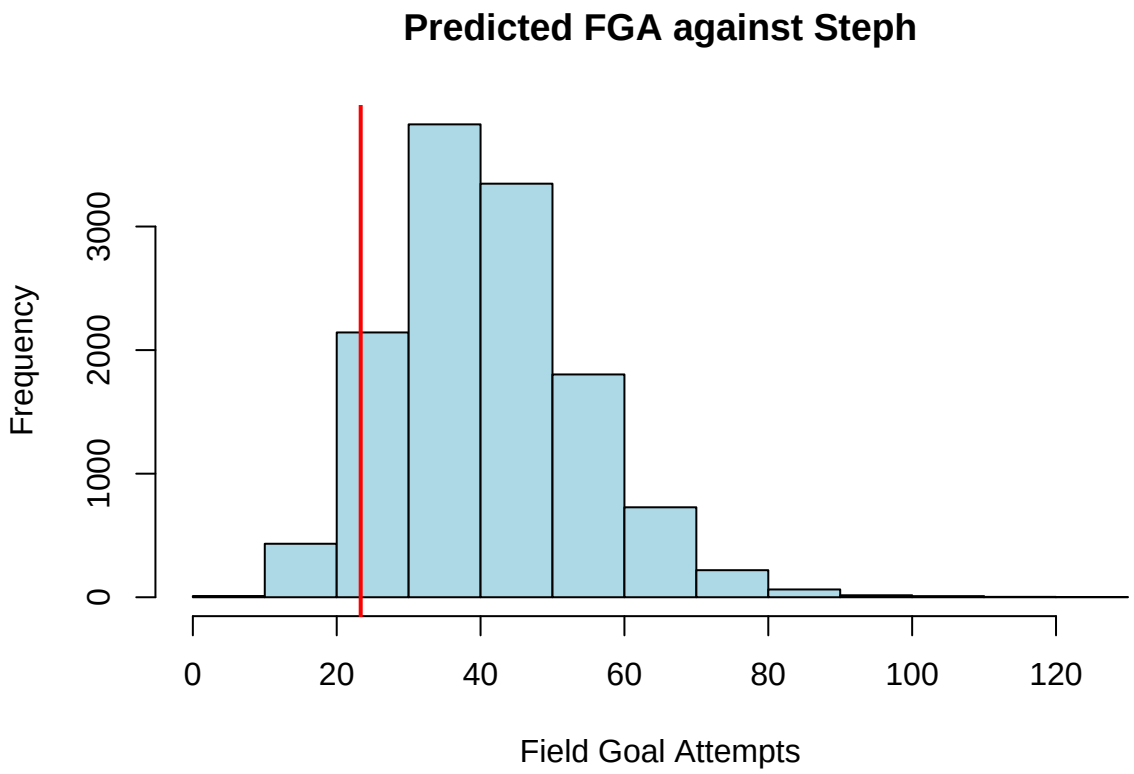


Find the columns pertaining to the steph v lebron games

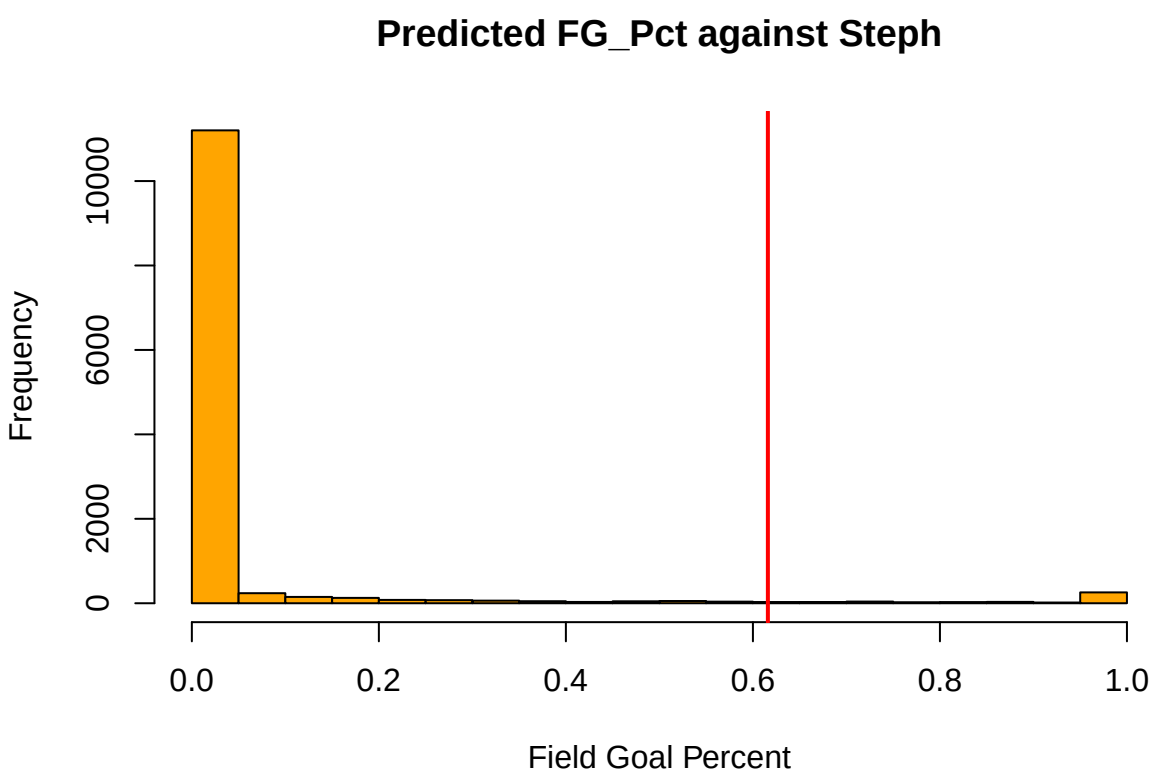
```
cols_of_interest = which(lebron_dat$GAME_ID %in%lebron_vs_steph_games, TRUE)
n_pred = round(mean(n_keep[n_iter, cols_of_interest]))
p_pred = mean(p_keep[n_iter, cols_of_interest]) # p's are pre scaled
samps = rbinom(100, n_pred, p_pred)
hist(samps,
     main = "Predicted FGM against Steph",
     xlab = "Field Goals Made",
     col = "lightgreen", breaks = 15,
     xlim = c(0, 16))
abline(v = mean(lebron_dat$FGM[lebron_dat$GAME_ID %in% lebron_vs_steph_games]), col = "red", lwd = 2)
```




```
hist(n_keep[800:n_iter, cols_of_interest],
     main = "Predicted FGA against Steph",
     xlab = "Field Goal Attempts",
     col = "lightblue", breaks = 15)
abline(v = mean(lebron_dat$FGA[lebron_dat$GAME_ID %in% lebron_vs_steph_games]), col = "red", lwd = 2)
```




```
hist(p_keep[800:n_iter, cols_of_interest],
     main = "Predicted FG_Pct against Steph",
     xlab = "Field Goal Percent",
     col = "orange", breaks = 15)
abline(v = mean(lebron_dat$FG_PCT[lebron_dat$GAME_ID %in% lebron_vs_steph_games]), col = "red", lwd = 2
```


Predicted FG_Pct against Steph
Frequency
0
2000
6000
10000
0.0
0.2
0.4
0.6
0.8
1.0
Field Goal Percent