Model 1 fixed

Ben Moolman, Craig Orman, Ethan Pross

Beginning Data preparing

```r
## 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
```

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

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

DRTG data being joined to the starting data

```r
## 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 1 implementation

Situation:

$$
\begin{aligned}
y_{ijk} &\sim Binom(n_{ijk}, p_{ik}) \\
n_{ijk} &\sim \text{fixed n, trying mean, median, max} \\
p_{ik} &= p_i \times \exp(\gamma(\text{DRTG}_k - \bar{\text{DRTG}})) \\
p_i &\sim Beta(5, 5)
\end{aligned}
$$

Set up

```r
# Filters for just LeBron games (71 games played by LeBron)
lebron_dat = starting_dat[starting_dat$PLAYER_ID %in% 2544, ]

# Gets centered_OPP_DRTG for each team LeBron played against (28 total teams)
lebron_team_drtg <- lebron_dat %>%
  group_by(OPP_TEAM_ID) %>%
  summarize(centered_OPP_DRTG = first(centered_OPP_DRTG), .groups = "drop") %>%
  arrange(OPP_TEAM_ID)

# Get the field goal percentage for LeBron over the 71 games
true_p = mean(lebron_dat$FG_PCT)
# Get the mean number of field goals attempted per game
n_median = median(lebron_dat$FGA)
```

Metropolis Hastings implementation to Obtain p_i

```r
log_q = function(theta, y, n) {
  if (theta < 0 || theta > 1) return(-Inf)
  sum(dbinom(y, size = n, prob = theta, log = TRUE)) + dbeta(theta, 5, 5, log = TRUE)
}

MH_beta_binom = function(current = 0.5, prop_sd = 0.05, n_vec, y_vec, n_iter = 1000) {
```

```r
  samps = rep(NA, n_iter)
  for (i in 1:n_iter) {
    proposed = rnorm(1, current, prop_sd)
    logr = log_q(proposed, y = y_vec, n = n_vec) - log_q(current, y = y_vec, n = n_vec)
    if (log(runif(1)) < logr) current = proposed
    samps[i] = current
  }
  return(samps)
}

MH_beta_grid_search = function(current = 0.5, n_vec, y_vec, n_iter = 1000) {
  vals <- seq(0.005, 1, by = 0.005)
  effect_sizes <- data.frame(sd = vals, ess = NA)

  for (i in seq_along(vals)) {
    samps <- MH_beta_binom(current = current, prop_sd = vals[i],
                           n_vec = n_vec, y_vec = y_vec, n_iter = n_iter)
    effect_sizes$ess[i] <- effectiveSize(samps)
  }

  best_sd <- effect_sizes$sd[which.max(effect_sizes$ess)]
  final_samps <- MH_beta_binom(current = current, prop_sd = best_sd,
                               n_vec = n_vec, y_vec = y_vec, n_iter = n_iter)

  return(list(
    samples = final_samps,
    best_sd = best_sd,
    ess_table = effect_sizes
  ))
}
```

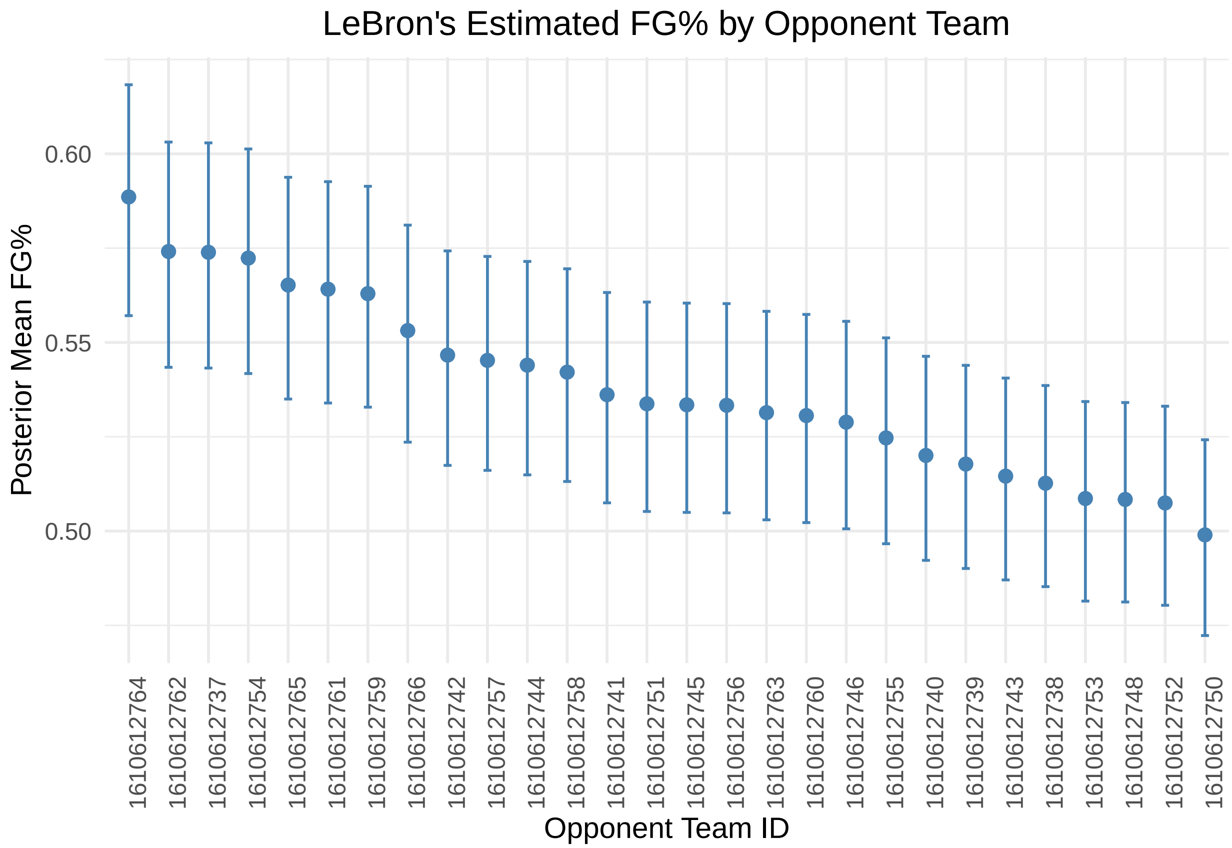
Now run MH

```r
Y <- lebron_dat$FGM
N_vec <- lebron_dat$FGA

mh_result <- MH_beta_grid_search(current = 0.5,
                                 y_vec = Y,
                                 n_vec = N_vec,
                                 n_iter = 1000)

p_samples <- mh_result$samples
mh_result$best_sd  # tuning parameter chosen
```

```
## [1] 0.055
```

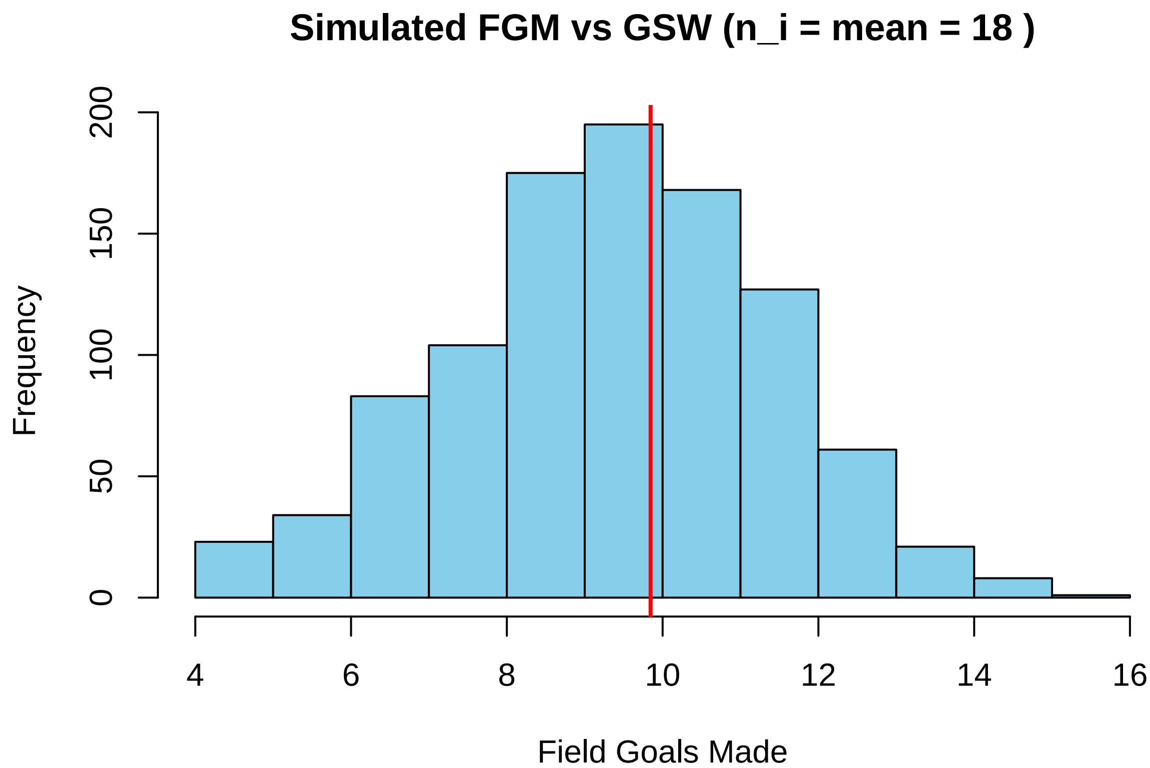
```r
drtg_vec <- lebron_team_drtg$centered_OPP_DRTG  # length = 28
gamma_val <- 0.01

# Compute p_ik matrix: rows = posterior draws, cols = opponent teams
p_ik_matrix <- outer(
  p_samples,
```

```r
  drtg_vec,
  FUN = function(p, drtg) p * exp(gamma_val * drtg)) |> pmin(1)  # clip to max 1

# Posterior mean FG% per opponent
p_ik_mean <- colMeans(p_ik_matrix)

# Credible intervals
p_ik_CI <- apply(p_ik_matrix, 2, quantile, probs = c(0.025, 0.975))

# Combine into a table
p_ik_summary <- data.frame(
  OPP_TEAM_ID = lebron_team_drtg$OPP_TEAM_ID,
  p_ik_mean = p_ik_mean,
  p_ik_lower = p_ik_CI[1,],
  p_ik_upper = p_ik_CI[2,]
)

ggplot(p_ik_summary, aes(x = reorder(as.factor(OPP_TEAM_ID), -p_ik_mean), y = p_ik_mean)) +
  geom_point(color = "steelblue", size = 2) +
  geom_errorbar(aes(ymin = p_ik_lower, ymax = p_ik_upper), width = 0.2, color = "steelblue") +
  labs(
    title = "LeBron's Estimated FG% by Opponent Team",
    x = "Opponent Team ID",
    y = "Posterior Mean FG%"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1),
    plot.title = element_text(hjust = 0.5)
  )
```

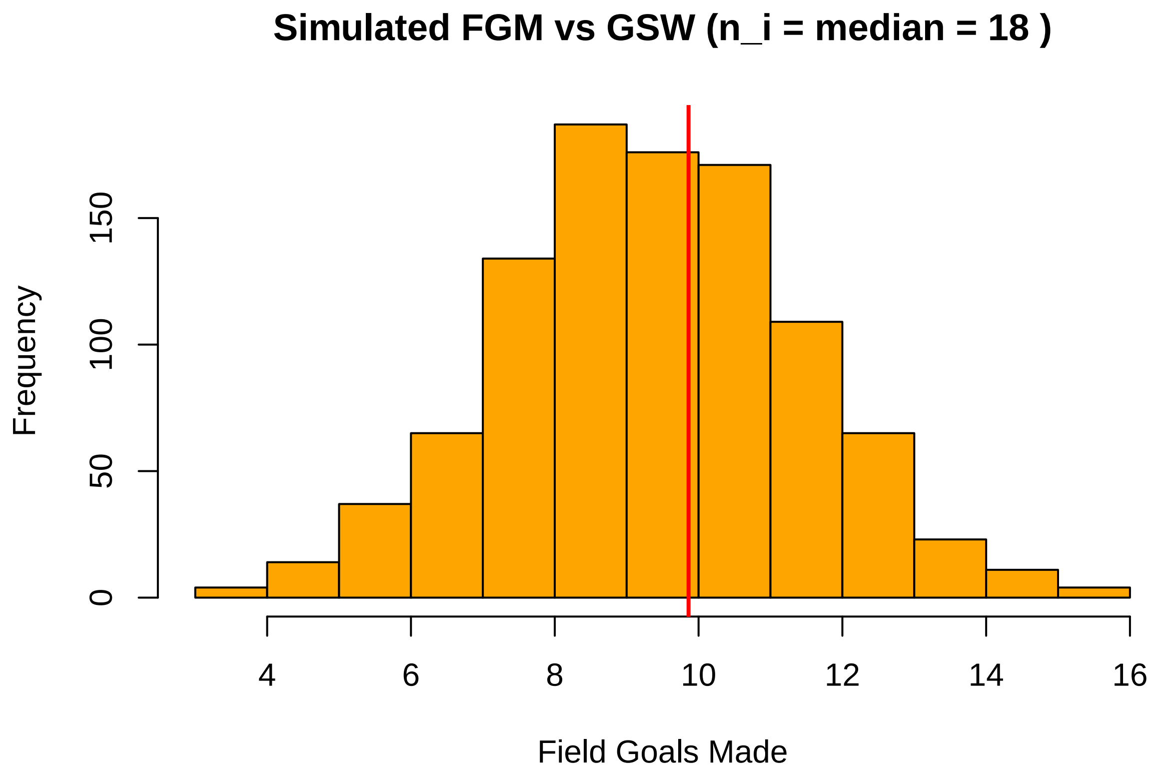


Now we will look at LeBron against Stephen Curry predictions (against Golden State Warriors, TEAM_ID = 1610612744)

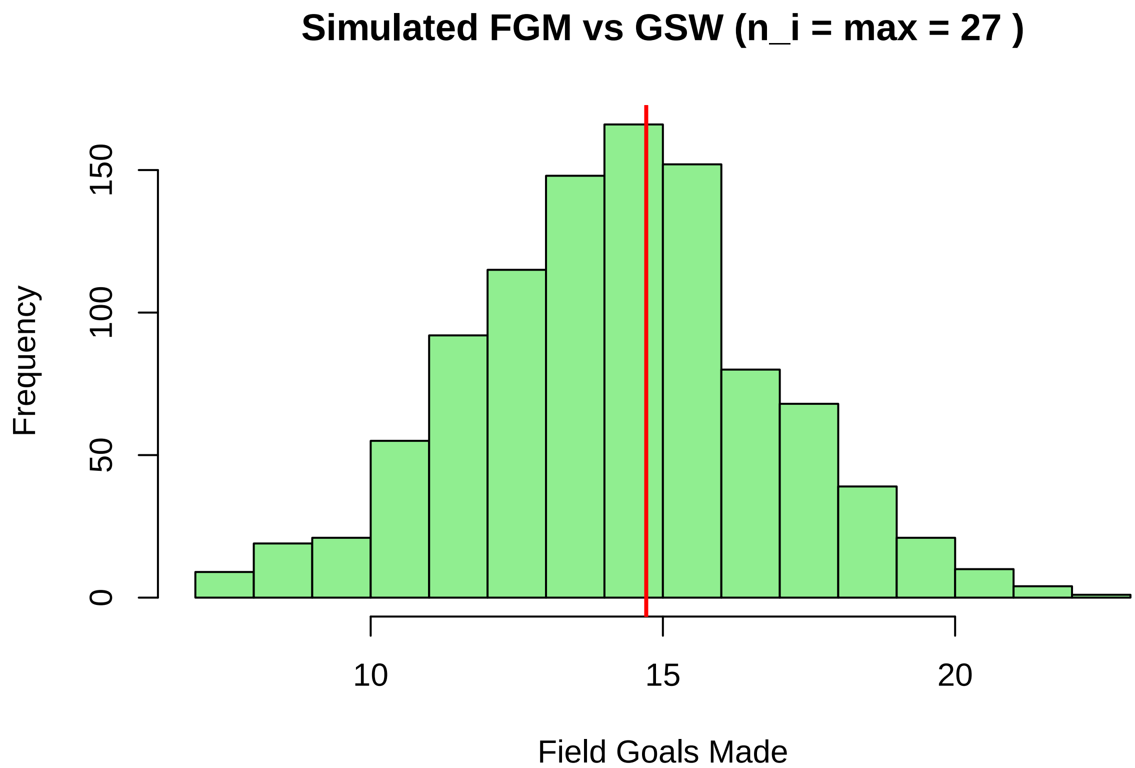
```r
# Get centered DRTG for GSW (Golden State Warriors)
drtg_gsw <- lebron_team_drtg$centered_OPP_DRTG[lebron_team_drtg$OPP_TEAM_ID == 1610612744]
gamma_val <- 0.01  # fixed tuning value
p_ik_gsw <- p_samples * exp(gamma_val * drtg_gsw)
p_ik_gsw <- pmin(p_ik_gsw, 1)  # ensure FG% stays within [0,1]

# FGAs for LeBron
n_mean <- round(mean(lebron_dat$FGA, na.rm = TRUE))
n_median <- round(median(lebron_dat$FGA, na.rm = TRUE))
n_max <- max(lebron_dat$FGA, na.rm = TRUE)

# Simulations using different n_i values
set.seed(5440)
fgm_sim_mean <- rbinom(length(p_ik_gsw), size = n_mean, prob = p_ik_gsw)
fgm_sim_median <- rbinom(length(p_ik_gsw), size = n_median, prob = p_ik_gsw)
fgm_sim_max <- rbinom(length(p_ik_gsw), size = n_max, prob = p_ik_gsw)

# Plot 1: Mean FGA
hist(fgm_sim_mean,
     main = paste("Simulated FGM vs GSW (n_i = mean =", n_mean, ")"),
     xlab = "Field Goals Made",
     col = "skyblue", breaks = 15)
abline(v = mean(fgm_sim_mean), col = "red", lwd = 2)
```




```r
# Plot 2: Median FGA
hist(fgm_sim_median,
     main = paste("Simulated FGM vs GSW (n_i = median =", n_median, ")"),
     xlab = "Field Goals Made",
     col = "orange", breaks = 15)
abline(v = mean(fgm_sim_median), col = "red", lwd = 2)
```




```r
# Plot 3: Max FGA
hist(fgm_sim_max,
     main = paste("Simulated FGM vs GSW (n_i = max =", n_max, ")"),
     xlab = "Field Goals Made",
     col = "lightgreen", breaks = 15)
abline(v = mean(fgm_sim_max), col = "red", lwd = 2)
```



Interpretation and Generalization

This simulation estimates the **posterior predictive distribution** of field goals made (FGM) for a given player i (in this case, LeBron James) when matched against a specific opponent team k (here, the Golden State Warriors). The distribution reflects:

- Uncertainty in the player's overall shooting ability p_i,
- Game-to-game variability in performance, and
- Adjustments for the defensive strength of the opposing team via the centered DRTG metric.

This framework can be generalized to *any* player i and *any* opponent k in the dataset. By sampling from the posterior of p_i, scaling it by the opponent's centered DRTG using γ, and drawing from a binomial distribution with a fixed n_i, we obtain predictive in-game outcomes tailored to individual matchups.

Model 1 implementation

Situation:

$$\begin{aligned} y_{ijk} &\sim Binom(n_{ijk}, p_{ik}) \\ n_{ijk} &\sim \text{by model} \\ p_{ik} &= p_i \times \exp(\gamma(\text{DRTG}_k - \bar{\text{DRTG}})) \\ p_i &\sim Beta(5, 5) \end{aligned}$$

Set up

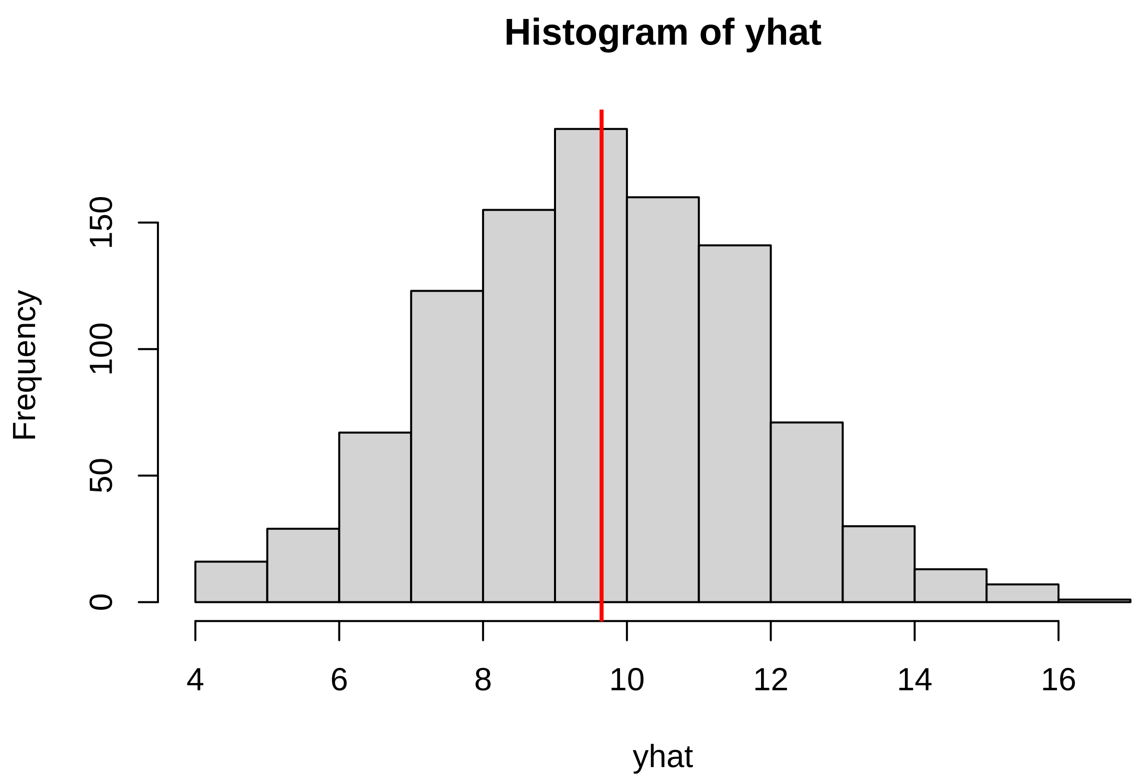
```
lebron_dat = starting_dat[starting_dat$PLAYER_ID %in% 2544, ]
model_1_dat = lebron_dat[lebron_dat$GAME_ID %in% lebron_vs_steph_games,] # This basically turned out to
Y = lebron_dat$FGM
N = nrow(lebron_dat)
true_p = mean(lebron_dat$FG_PCT)
n_median = median(lebron_dat$FGA)
```

Metropolis Hastings implementation

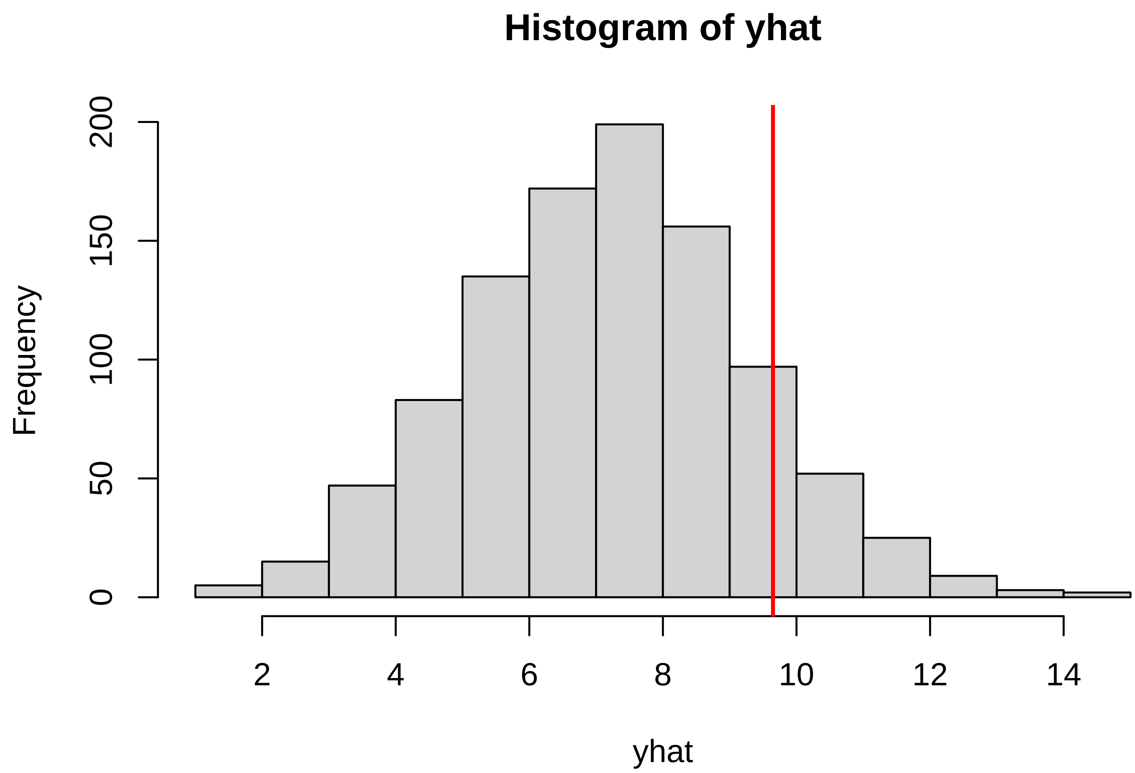
```
#This is the uhhhh posterior I think
log_q = function(theta, y=3, n=10) {
  if (theta<0 | theta>1) return(-Inf)
  (y-0.5)*log(theta)+(n-y-0.5)*log(1-theta)
}

# This runs the Metropolis hastings algorithm
MH_beta_binom = function(current = 0.5, prop_sd, n = n) {
  current = 0.5 # Initial value
  samps = rep(NA,N)
  for (i in 1:N) {
    proposed = rnorm(1, current, prop_sd) # tuning parameter goes here
    logr = log_q(proposed, y=Y[i], n=n)-log_q(current, y=Y[i], n=n)
    if (log(runif(1)) < logr) current = proposed #comparitor
    samps[i] = current
  }
  paste("Acceptance Rate: ", length(unique(samps))/n)
  return(samps)
}

# This is such a grid search of the MH using a variety of Proposed SDs
# and choosing the best one to maximize the effective sample size
MH_beta_grid_serach = function(current = 0.5, n) {
  vals = seq(from=0.01, to = 20, by = 0.01)
  effect_sizes = data.frame(sd = vals, effect_size = NA)
  for (i in 1:length(vals)) {
    samps = MH_beta_binom(prop_sd = vals[i], n=n)
    effect_sizes[i,2] = effectiveSize(samps)
  }
  best_sd = effect_sizes$sd[effect_sizes$effect_size == max(effect_sizes$effect_size)] #select best sd
  return(MH_beta_binom(prop_sd = best_sd, n=n))
}
```

So basically, I don't think I did the predictive posterior correctly. And also, we have low ESS and the y's don't fit the best! For shame team, for shame.

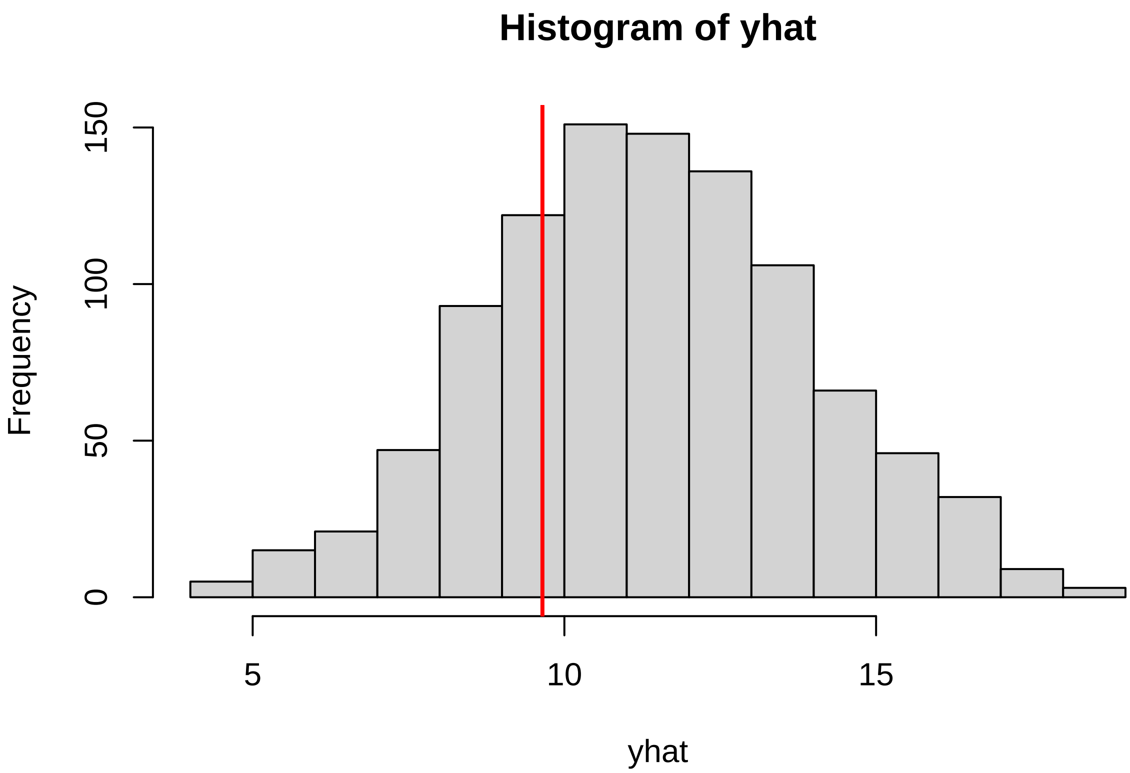
```
n_mean = round(mean(lebron_dat$FGA)) #NAs casue mean is not an integer
samps = MH_beta_grid_serach(n=n_mean)
yhat <- rbinom(1000, n_mean, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```




```r
n_median = median(lebron_dat$FGA)
samps = MH_beta_grid_serach(n=n_median)
yhat <- rbinom(1000, n_median, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```




```r
n_max = max(lebron_dat$FGA)
samps = MH_beta_grid_serach(n=n_max)
yhat <- rbinom(1000, n_max, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```


Histogram of yhat
Frequency
0
50
100
150
5
10
15
yhat