Model 1 fixed

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 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

```r
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)
```

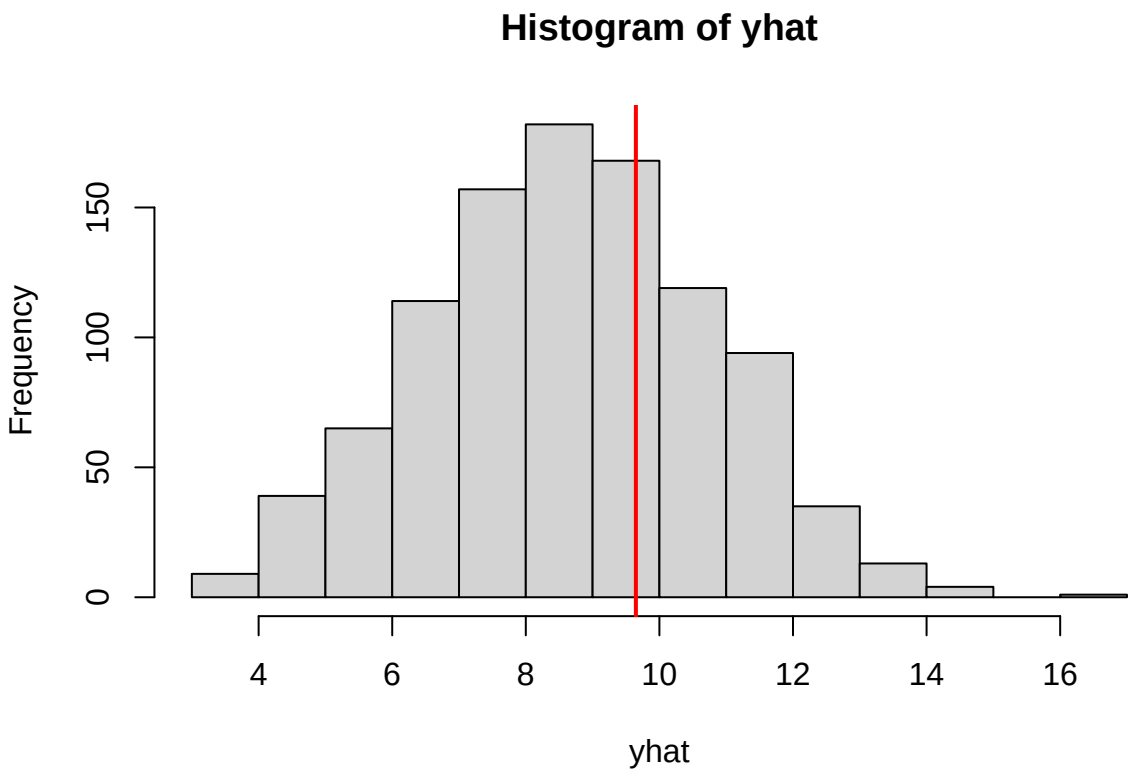
Metropolis Hastings implementation

```r
#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)
```

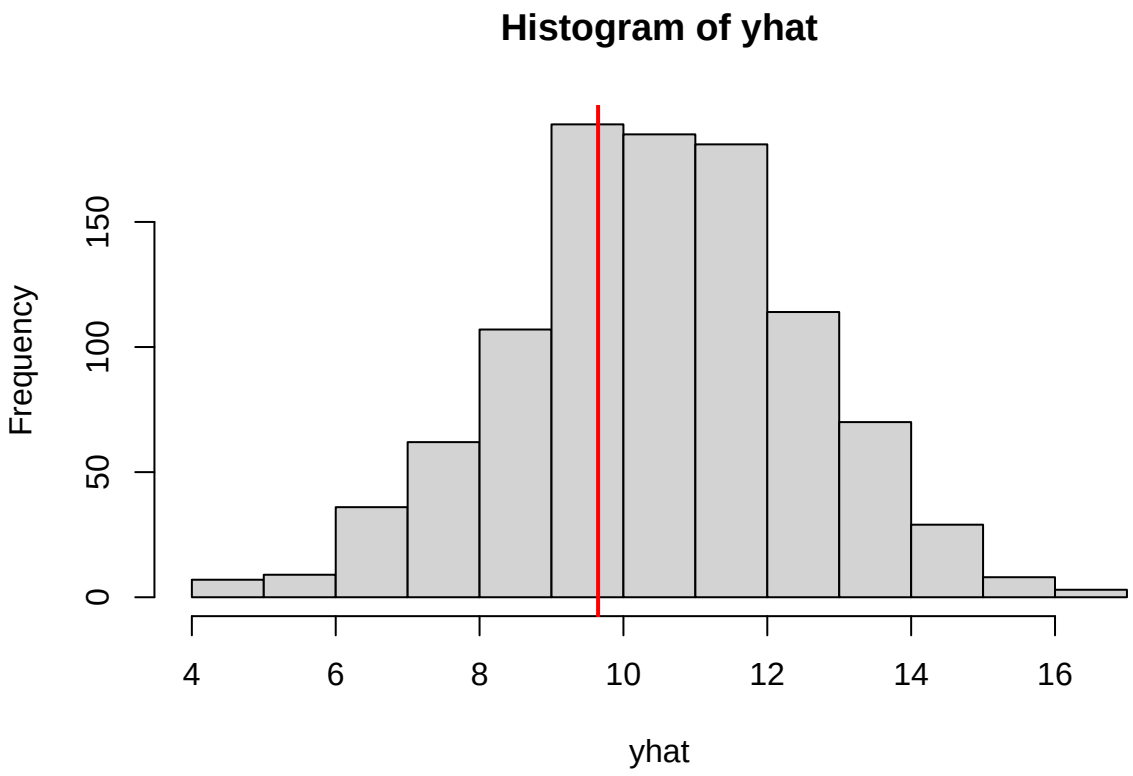
```
    if (log(runif(1)) < logr) current = proposed #comparitor
    samps[i] = current
  }
  paste("Acceptance Rate: ", length(unique(samps))/n)
  return(samps)
}

# 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.

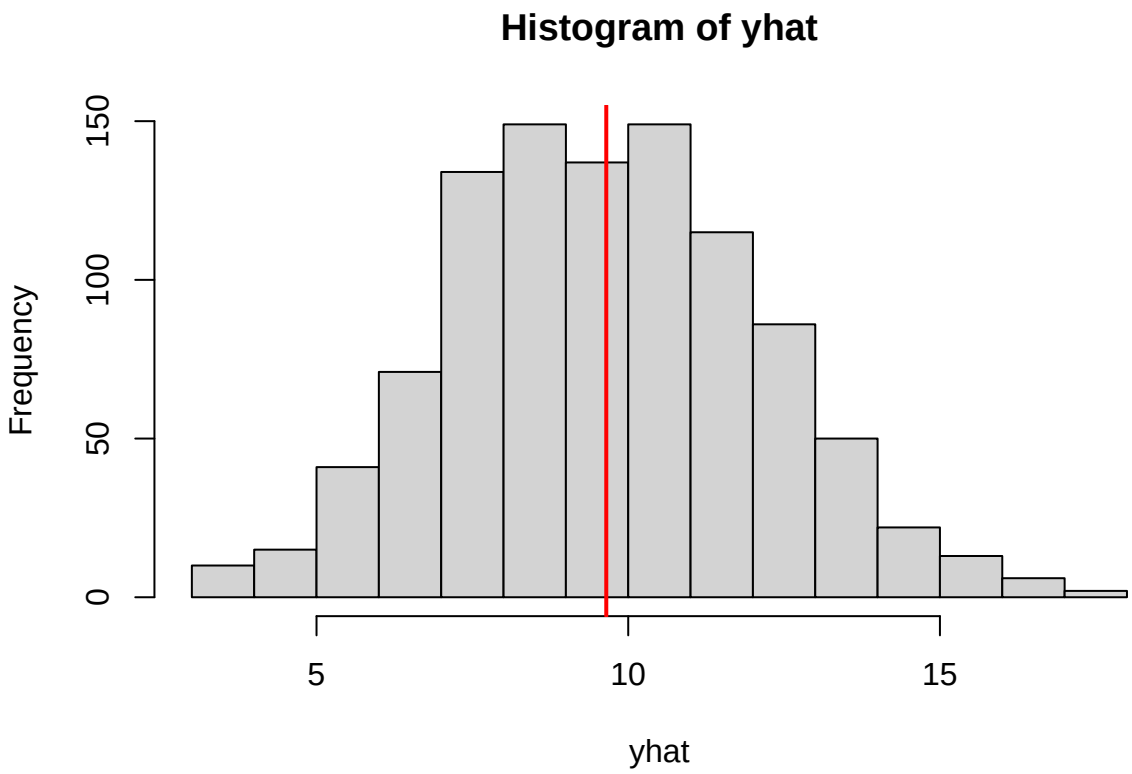
```
n_mean = round(mean(lebron_dat$FGA)) #NAs casue mean is not an integer
samps = MH_beta_grid_serach(n=n_mean)
yhat <- rbinom(1000, n_mean, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```




```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