Actual Model

Ben Moolman, Craig Orman, Ethan Pross

Beginning Data upload

```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
team_points <- original_tbl %>%
  group_by(GAME_ID, TEAM_ID) %>%
  summarize(TeamPoints = sum(PTS), .groups = "drop")

# tidying
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)
```

```
## Warning in inner_join(., team_points_opponent, by = "GAME_ID"): Detected an unexpected many-to-many r
## i Row 1 of ‘x‘ matches multiple rows in ‘y‘.
## i Row 1 of ‘y‘ matches multiple rows in ‘x‘.
## i If a many-to-many relationship is expected, set ‘relationship =
##   "many-to-many"‘ to silence this warning.
```

```r
# 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] NA NA
```

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

```
## [1] NA
```

prepping the model

Situation:

$$y_{ijk} \sim Binom(n_{ijk}, p_{ik})$$
$$n_{ijk} \sim \text{by model}$$
$$p_{ik} \sim Beta(5,5)$$

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

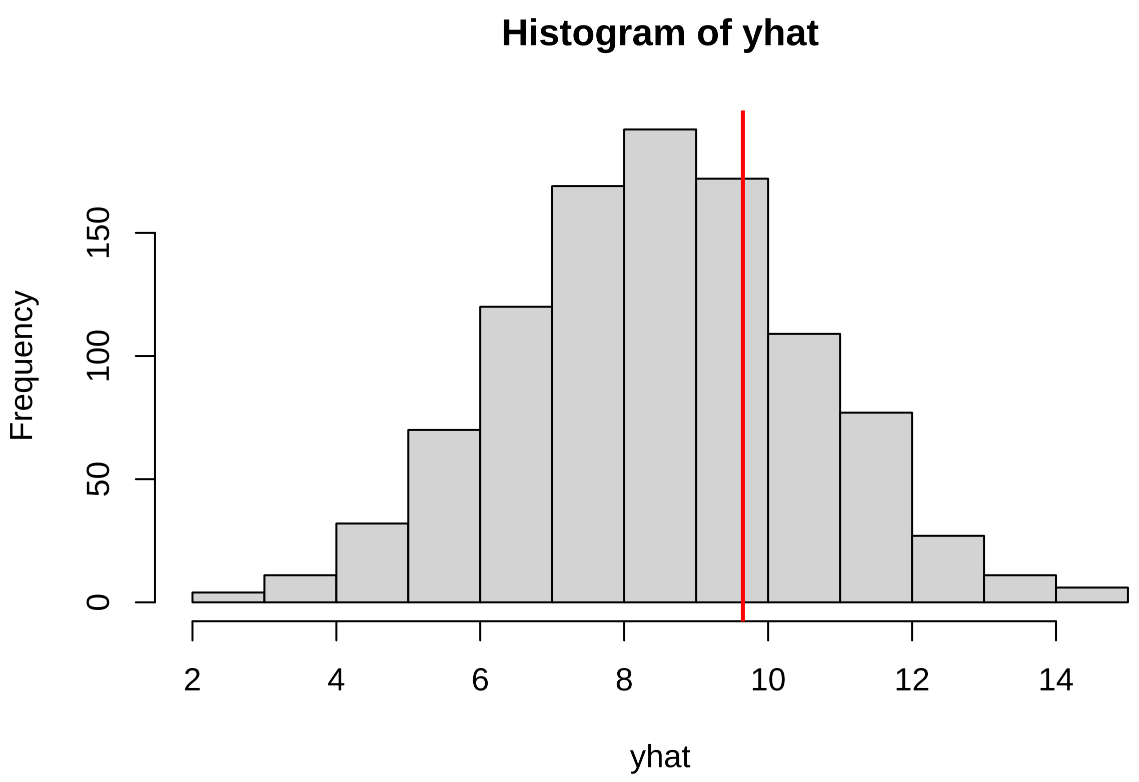
sample values for p!

```
#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 is 0.4
    logr = log_q(proposed, y=Y[i], n=n)-log_q(current, y=Y[i], n=n)
    if (log(runif(1)) < logr) current = proposed
    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 max
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)]
  return(MH_beta_binom(prop_sd = best_sd, n=n))
}
```

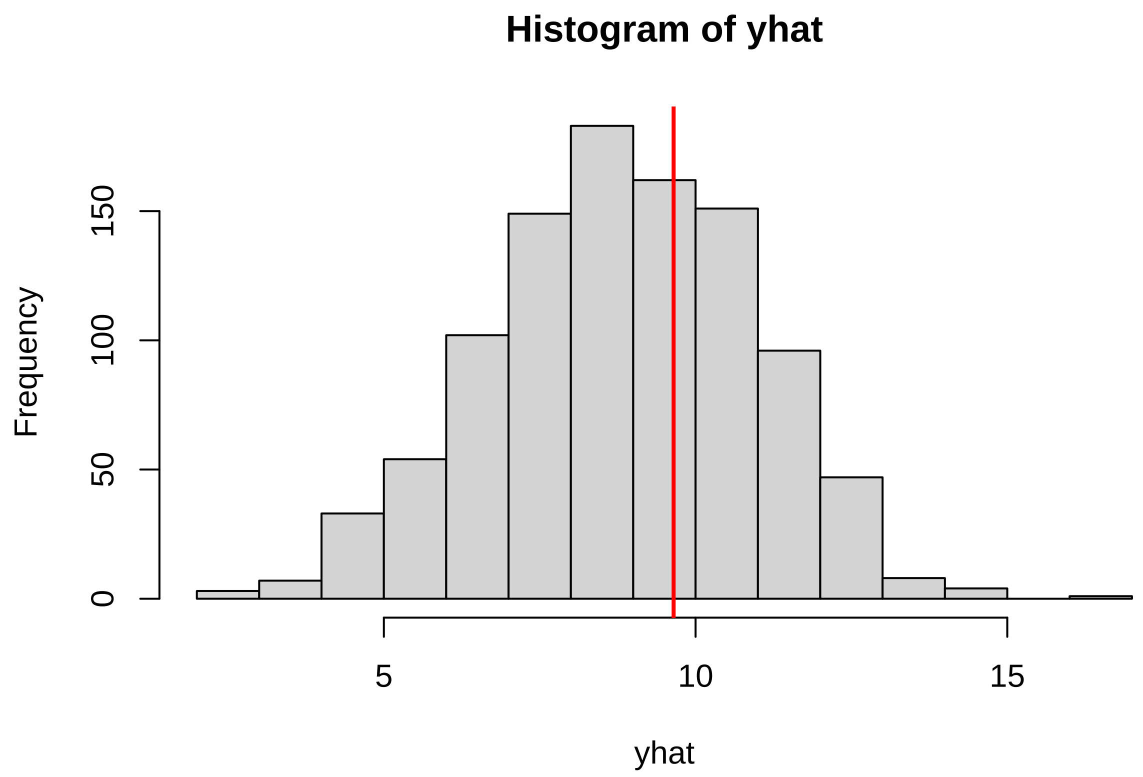
Model 1 using N as fixed

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.

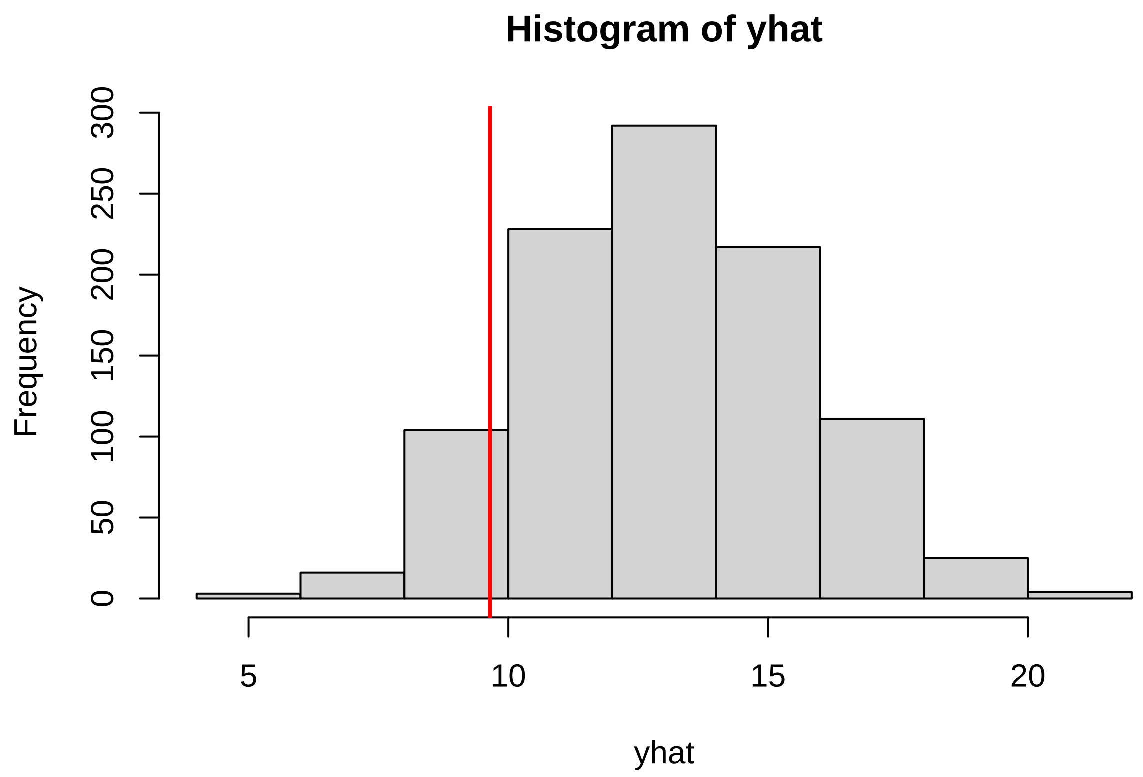
```r
n_mean = round(mean(lebron_dat$FGA)) #NAs casue mean is not an integer
samps = MH_beta_grid_serach(n=n_mean)
yhat <- rbinom(1000, n_mean, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```



```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)
```



```
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)
```



Model 2 using N as poisson

Model 3 using N as negbinom