# Actual Model

Ben Moolman, Craig Orman, Ethan Pross

```
## Warning: package 'dplyr' was built under R version 4.4.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Warning: package 'tidyverse' was built under R version 4.4.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

## Warning: package 'coda' was built under R version 4.4.2
```

```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.
```

```
# 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
```

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

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

### prepping the model

Situation:

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

```
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_mean = round(mean(lebron_dat$FGA)) #NAs casue mean is not an integer
n_median = median(lebron_dat$FGA)
n_max = max(lebron_dat$FGA)
```

## sample values for p!

```
n = n_mean
prop_sd = 0.4

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)
}


MH_beta_binom = function(current = 0.5, prop_sd) {
  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)
}
```

Now we use the power of functions

```
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])
  effect_sizes[i,2] = effectiveSize(samps)
}
best_sd = effect_sizes$sd[effect_sizes$effect_size == max(effect_sizes$effect_size)]
paste("max effective sample size: ", round(max(effect_sizes$effect_size),4), "at sd: ", best_sd)
```

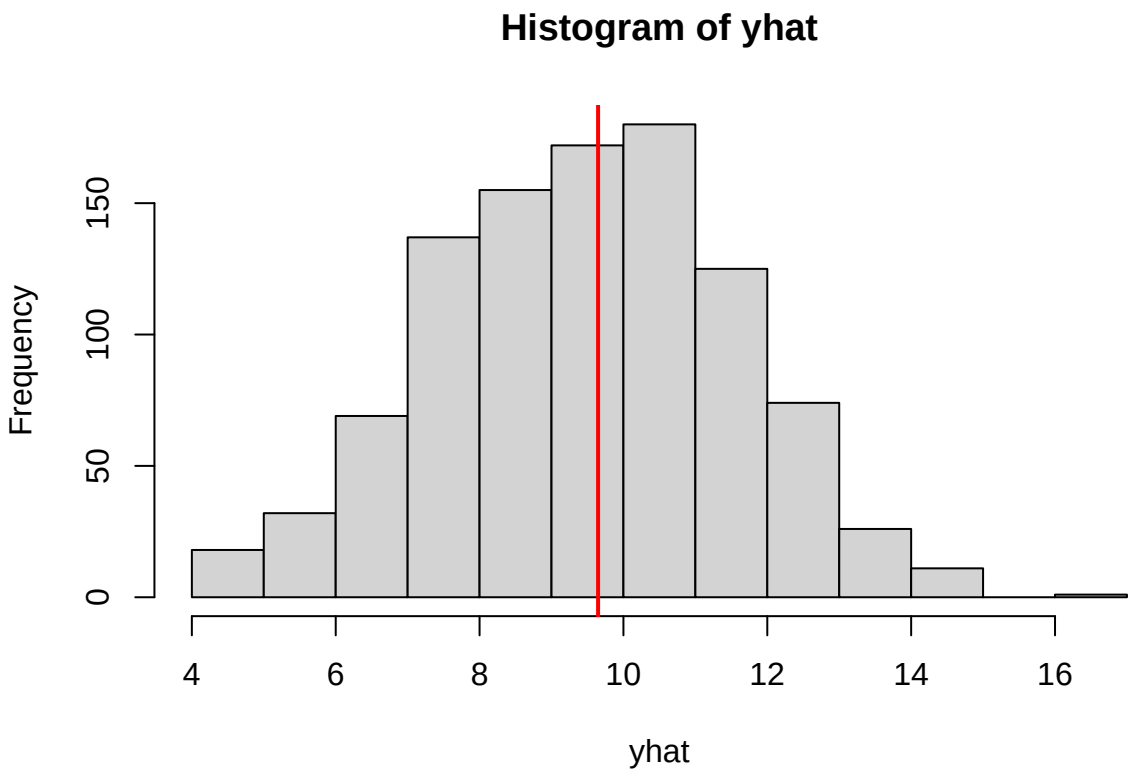
```
## [1] "max effective sample size:  210.0583 at sd:  4.19"
```

```
# effectiveSize(samps)
# hist(samps)
# abline(v = true_p, col = "red", lwd = 2)
# plot(as.mcmc(samps))
```

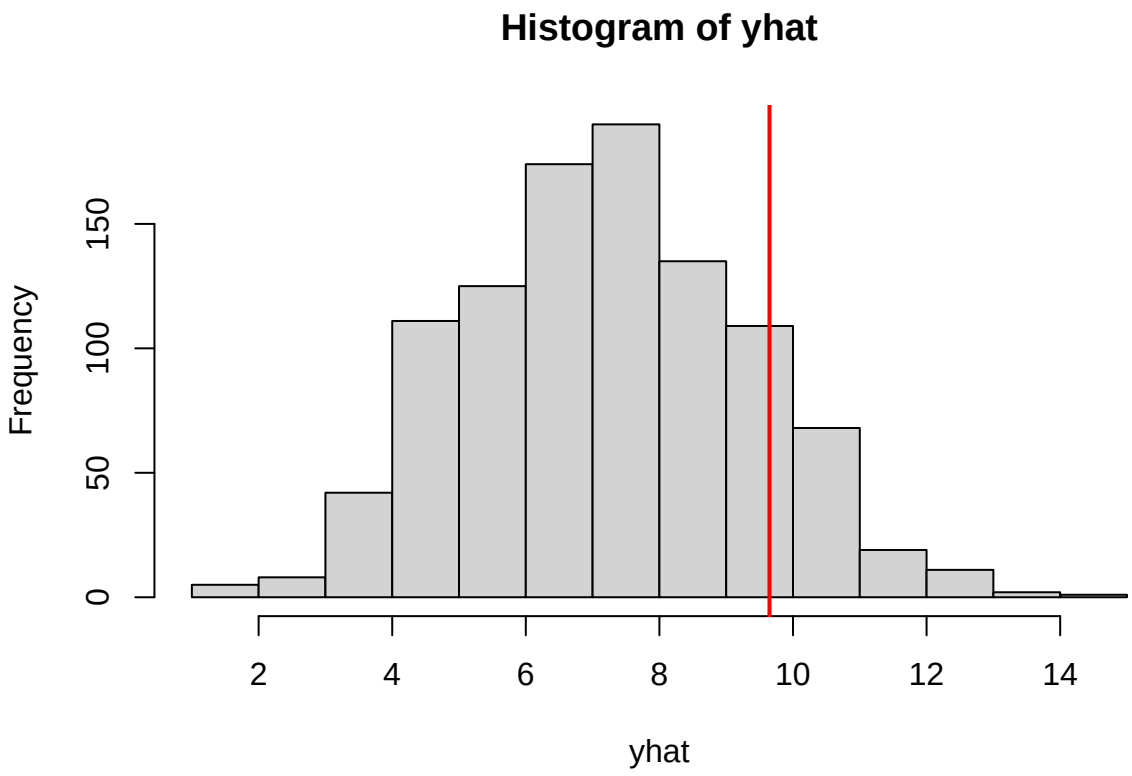
## Sample from the predictive posterior

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.

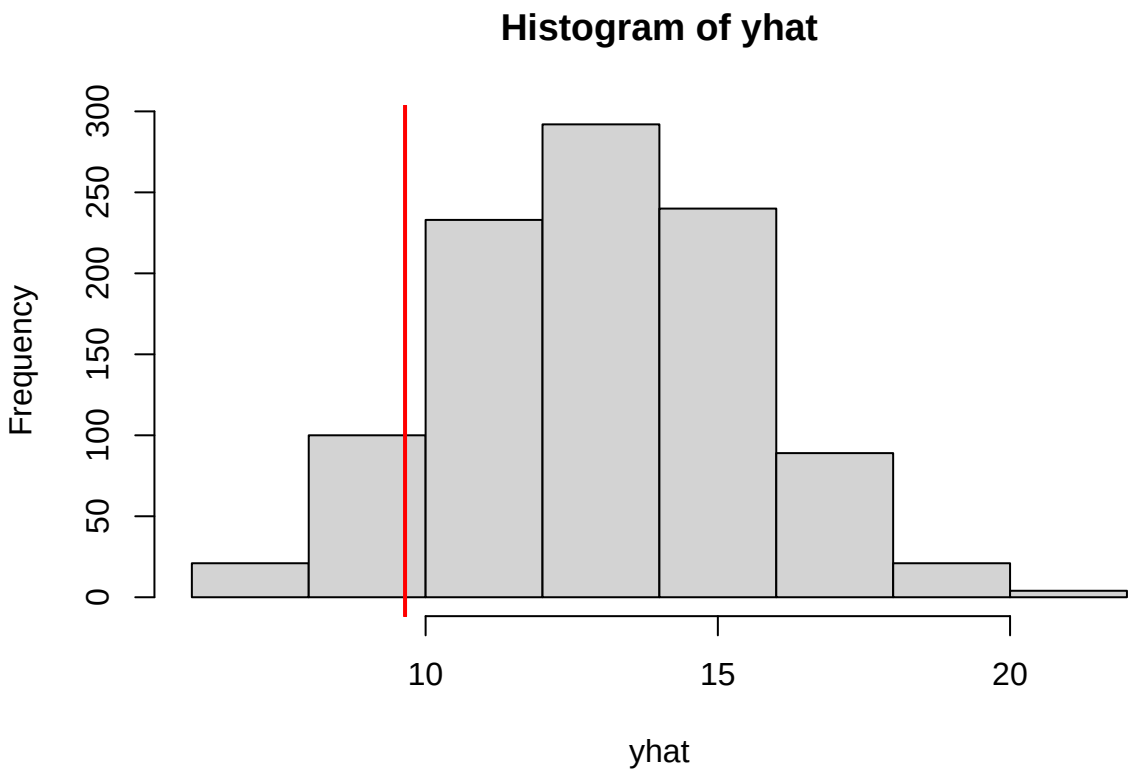
```
samps = MH_beta_binom(prop_sd = best_sd)
yhat <- rbinom(1000, n_mean, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```




```
samps = MH_beta_binom(prop_sd = best_sd)
yhat <- rbinom(1000, n_median, mean(samps))
hist(yhat)
abline(v = mean(Y), col = "red", lwd = 2)
```




```
samps = MH_beta_binom(prop_sd = best_sd)
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
200
250
300
10
15
20
yhat