Faceoffs

Ian Koller

2024-12-11

Read in previously prepared play-by-play data for every faceoff in the 2023/24 NHL season and every player who took part in those games:

```
library(tidyverse)
library(rstan)
mc.cores <- parallel::detectCores()
faceoffs <- read_csv("faceoffs.csv")
boxscore_players <- read_csv("boxscore_players.csv")
```

We need to scrape whether player shoots left or right building off previous scraping and get rid of All-Star game participants (since this game includes non-NHL players and would cause problems for inference to have players not connected to the network of player matchups):

```
boxscore_players_distinct <- boxscore_players %>%
  distinct(playerId, .keep_all = TRUE) %>%
  filter(as.numeric(str_remove(game_id, pattern = ("^[0-9]{4}"))) < 100000)
boxscore_players_distinct$url <-
  str_c("https://api-web.nhle.com/v1/player/",
        boxscore_players_distinct$playerId,"/landing")

for (i in 1:nrow(boxscore_players_distinct)){
  boxscore_players_distinct$shootsCatches[i] <-
    jsonlite::fromJSON(boxscore_players_distinct$url[i])$shootsCatches
}

boxscore_players_distinct <- boxscore_players_distinct %>%
  select(c(playerId, fullname, shootsCatches, team))

faceoffs <- faceoffs %>% left_join((boxscore_players_distinct %>%
                                   mutate(playerId = as.numeric(playerId)) %>%
                                    rename("p1_fullname" = "fullname",
                                           "p1_team" = "team",
                                           "p1_shoots" = shootsCatches)),
                                  by = c("player_1" = "playerId"))

faceoffs <- faceoffs %>% left_join((boxscore_players_distinct %>%
                                   mutate(playerId = as.numeric(playerId)) %>%
                                     rename("p2_fullname" = "fullname",
                                            "p2_team" = "team",
                                            "p2_shoots" = shootsCatches)),
                                  by = c("player_2" = "playerId"))
```

```
faceoffs <- faceoffs %>%
  mutate(p1_faceoff_side =
           yCoord/22 * ifelse(p1_team == homeTeam, 1, -1) *
           ifelse(homeTeamDefendingSide == "right", 1, -1) *
           ifelse(p1_shoots == "L", -1, 1) + 2) %>%
  mutate(p2_faceoff_side = yCoord/22 * ifelse(p2_team == homeTeam, 1, -1) *
           ifelse(homeTeamDefendingSide == "right", 1, -1) *
           ifelse(p2_shoots == "L", -1, 1) + 2)


unique_faceoff_player <- faceoffs %>% select(c(player_1, player_2)) %>%
  pivot_longer(cols = everything()) %>% group_by(value) %>%
  summarize(n_wins = sum(name == "player_1"), n = n(), prop = n_wins/n) %>%
  arrange(desc(n)) %>% mutate(index = as.numeric(rownames(.)))

unique_faceoff_player <- unique_faceoff_player %>%
  left_join((boxscore_players_distinct %>%
               mutate(playerId = as.numeric(playerId))),
            by = c("value" = "playerId"))

faceoff_matrix_maker <- faceoffs %>% select(c(player_1, player_2)) %>%
  left_join((unique_faceoff_player %>% select(c(value, index))),
            by = c("player_1" = "value")) %>%
  rename("player_1_index" = "index") %>%
  left_join((unique_faceoff_player %>% select(c(value, index))),
            by = c("player_2" = "value")) %>%
  rename("player_2_index" = "index") %>%
  select(c(player_1_index, player_2_index))
```

The Bradley-Terry model can run into problems when entries in a head-to-head matchup are not part of the connected network of all other entries. This could happen in our case if two players took a faceoff against each other and never against anyone else. While the nature of faceoffs makes this extremely improbable over a whole season, we verify just in case.

```
faceoff_matrix <- matrix(0, nrow = nrow(unique_faceoff_player) + 1,
                         ncol = nrow(unique_faceoff_player) + 1)

for (i in 1:nrow(faceoff_matrix_maker)){
  faceoff_matrix[faceoff_matrix_maker$player_1_index[i],
                 faceoff_matrix_maker$player_2_index[i]] <-
    faceoff_matrix[faceoff_matrix_maker$player_1_index[i],
                   faceoff_matrix_maker$player_2_index[i]] + 1
}
for (i in 2:nrow(faceoff_matrix)){
  for (j in 1:(i-1)){
    faceoff_matrix[i,j] <- faceoff_matrix[i,j] + faceoff_matrix[j,i]
  }
}

unaccounted_indices <- 2:nrow(unique_faceoff_player)
connected_indices <- 1
added_indices <- list()
removed_rows <- 1
```

```r
while (removed_rows > 0){
  for (i in unaccounted_indices){
    for (j in connected_indices){
      if ((faceoff_matrix[i,j]) > 0) {
        added_indices <- append(added_indices, i) %>% unique()
        unaccounted_indices <- setdiff(unaccounted_indices, i)
      }
    }
  }
  removed_rows <- length(added_indices) - length(connected_indices)
  connected_indices <- added_indices
}
```

All accounted for.

```r
faceoff_data <- list("N" = nrow(faceoffs), "K" = nrow(unique_faceoff_player),
                  "y" = rep(as.integer(1),nrow(faceoffs)),
                  "player_1" = as.integer(faceoff_matrix_maker$player_1_index),
                  "player_2" = as.integer(faceoff_matrix_maker$player_2_index))
```

The first fit will just include player variables, not accounting for handedness. We fit the model and simulate new observations (y_pred) from the parameters to test our method.

```r
basic_fit <- function(dat){
basic_bradley_terry_fit <- stan(model_code = "
data{
  int N;
  int K;
  array[N] int <lower=0, upper=1> y;
  array[N] int <lower=1, upper=K> player_1;
  array[N] int <lower=1, upper=K> player_2;
}
parameters{
  vector[K] theta;
}
model{
  theta ~ normal(0, 0.33);
  y ~ bernoulli_logit(theta[player_1] - theta[player_2]);
}
generated quantities{
  array[N] int y_pred = bernoulli_logit_rng(theta[player_1] - theta[player_2]);
}
", data = dat, cores = mc.cores)

faceoff_samples <- rstan::extract(basic_bradley_terry_fit)

return(faceoff_samples)
}
faceoff_samples <- basic_fit(faceoff_data)
```

```r
faceoff_quantiles <- apply(faceoff_samples$theta, 2, function(x)
  quantile(x, probs = seq(0.1,0.9,0.1)))
```

```
test <- unique_faceoff_player %>% bind_cols(t(faceoff_quantiles))
```

The coefficients can be interpreted as the proportional change in log odds of succcess when a given player is taking part in a faceoff. Negative coefficients are bad and positive coefficients are good. The Bayesian approach allows us to easily present the uncertainty in parameter samples as quantiles. The 50% quantile for instance represents the median parameter sample. This is valuable in a case like this where players have vastly different numbers of faceoffs taken.

We could easily represent these coefficients as the percentage of faceoffs won given base odds of probability of success/probability of failure = 1/1 (i.e. 50% chance of success):

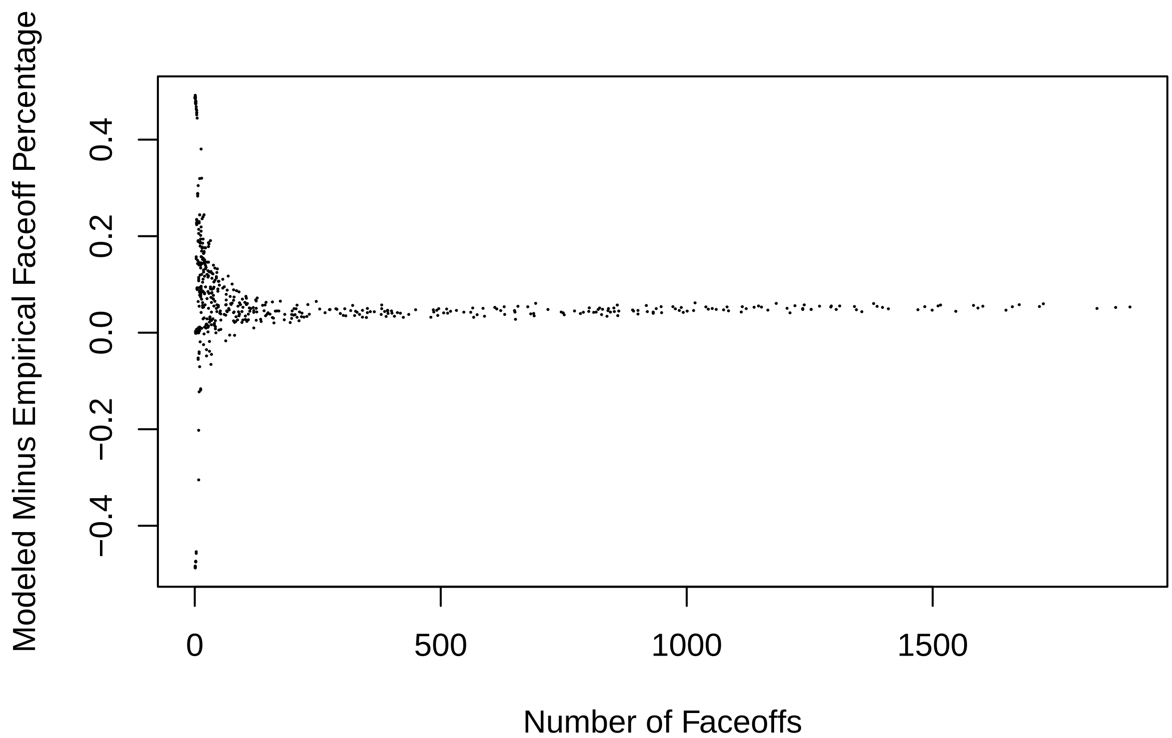
```
a <- exp(faceoff_samples$theta)/(1 +  exp(faceoff_samples$theta))
faceoff_percent_quantiles <- apply(a, 2, function(x)
  quantile(x, probs = seq(0.1,0.9,0.1)))
```

We see the necessity of modeling if we compare the difference between the empirical percentage of faceoffs won and the percentage won implied by our model:

```
unique_faceoff_percents <- unique_faceoff_player %>%
  bind_cols(t(faceoff_percent_quantiles)) %>% mutate(diff = `50%` - prop)
```

Observing the diff column here shows dramatic differences once we account for just the number of faceoffs taken leading to far greater uncertainty and potential large differences in opponent quality.

```
plot(unique_faceoff_percents$n, unique_faceoff_percents$diff,
      xlab = "Number of Faceoffs",
     ylab = "Modeled Minus Empirical Faceoff Percentage",
     pch = 20, cex = 0.1)
```



Not only does this plot show as expected that players with fewer than 50 faceoffs taken are exremely noisy, but there's a clear but very slight positive slope in difference for players with increasing faceoffs taken. Players who are good at faceoffs take a lot more of them, and face each other more often, so that the percentage won underrates the true strength of these players against the average player.

We can consider various posterior retrodictive summaries to compare how well our observations simulated from theta parameter samples compare to the observed data.

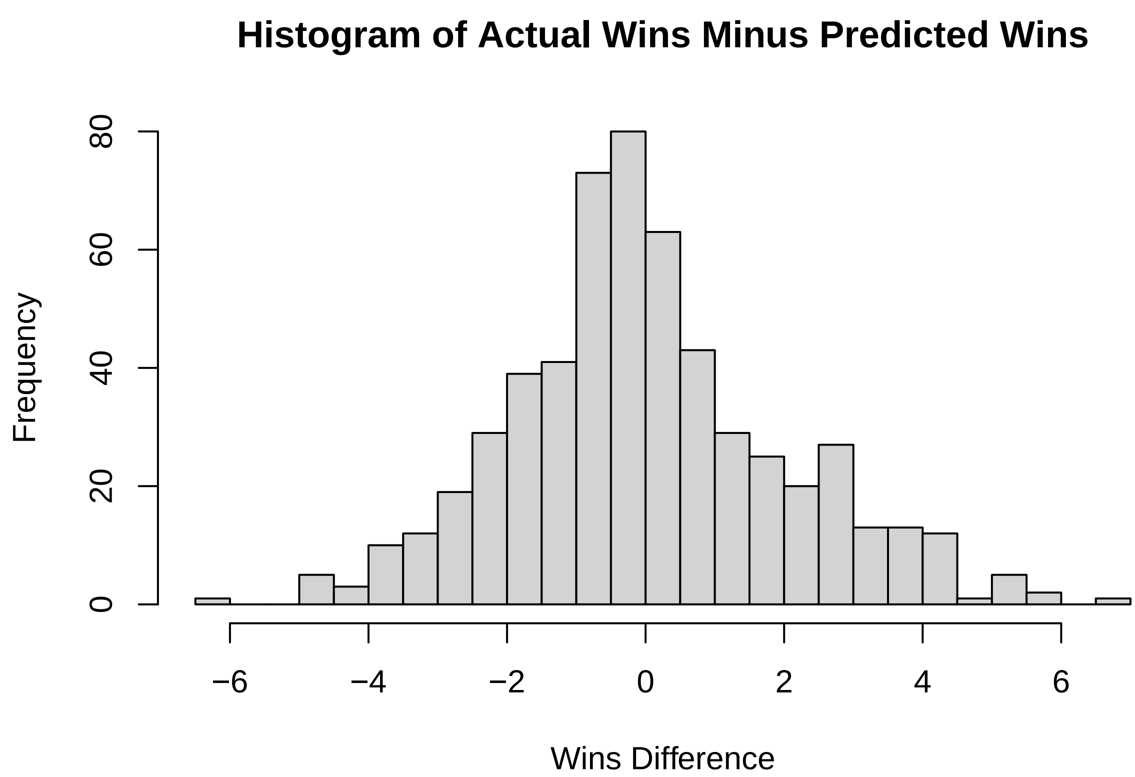
```
y_post_pred <- faceoff_samples$y_pred
sum(y_post_pred)/(4000 * nrow(faceoffs))
```

```
## [1] 0.5125904
```

```
player_wins <- bind_cols((faceoffs %>% select(c(player_1, player_2))),
          "player_1_wins" = apply(y_post_pred, 2,
                                function(x) sum(x))) %>%
  mutate("player_2_wins" = 4000 - player_1_wins)
player_wins <- player_wins %>%
  pivot_longer(cols = c(player_1_wins, player_2_wins)) %>%
  mutate(player = ifelse(name == "player_1_wins", player_1, player_2)) %>%
  select(-c(player_1, player_2))

player_wins_summary <- player_wins %>% group_by(player) %>%
  summarize(n_pred_wins = sum(value)/4000) %>%
  left_join(unique_faceoff_player %>%
            select(c(fullname, value, n_wins, n, prop)), by =
            c("player" = "value"))
```

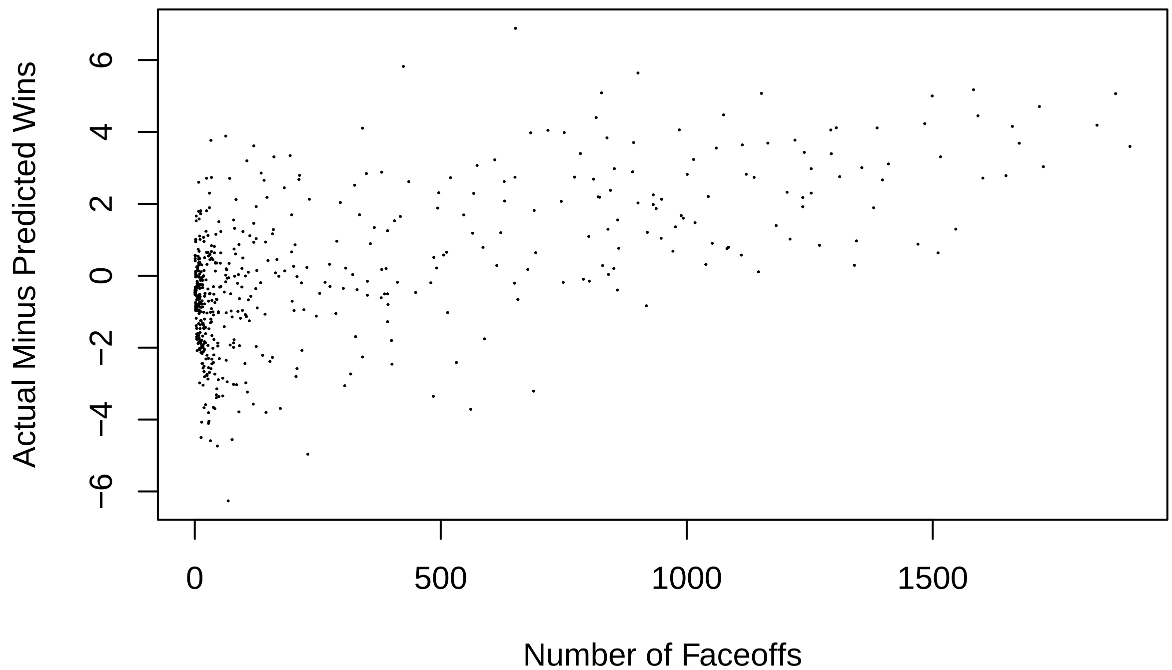
The raw residuals - the error from our predicted wins for each player minus the actual faceoff wins for that player - seem very reasonable here when bunched together in a histogram counting how many players are off by each amount - -2 wins, 0 wins, 2 wins, etc.

```
hist((player_wins_summary$n_wins - player_wins_summary$n_pred_wins),
     main = "Histogram of Actual Wins Minus Predicted Wins",
     xlab = "Wins Difference",
     breaks = 30)
```



However we can see a linear pattern in the residuals: players who take a lot of faceoffs are slightly but consistently underestimated.

```
plot(player_wins_summary$n,
     (player_wins_summary$n_wins - player_wins_summary$n_pred_wins),
      xlab = "Number of Faceoffs", ylab = "Actual Minus Predicted Wins",
     pch = 20, cex = 0.1)
```



I suspect that this is due to the fact that our prior assumes a neutral value for a player's strength at faceoffs, but the data is not informative for players who take very few faceoffs, who are also presumably weaker at them, or the coach would trust them to take more. Inferences for these players cannot be pulled away from the neutral prior by the weak data, so the model overestimates them, and when they go against a strong player, our model underestimates that strong player's chances. Even though the proportion of residual error is small for strong players with lots of data points (maybe 5 predicted wins fewer out of 1500+ faceoffs), we should still try to adjust our model.

We can start by simply putting a lower prior on theta. The 0.33 value for the standard deviation implies a success rate of about 8%. If we simply make the mean of our theta variable -0.33, implying a success rate of around 42%, we can hope that our data will resolve players with a lot of data close to their true value while not overestimating players who only take a few faceoffs.

```
lower_prior_fit <- function(dat){
bradley_terry_fit_lower_prior <- stan(model_code = "
data{
  int N;
  int K;
  array[N] int <lower=0, upper=1> y;
  array[N] int <lower=1, upper=K> player_1;
  array[N] int <lower=1, upper=K> player_2;
}
parameters{
  vector[K] theta;
}
model{
  theta ~ normal(-0.33, 0.33);
```

```
  y ~ bernoulli_logit(theta[player_1] - theta[player_2]);
}
generated quantities{
  array[N] int y_pred = bernoulli_logit_rng(theta[player_1] - theta[player_2]);
}
", data = dat, cores = mc.cores)
faceoff_samples_lower_prior <- rstan::extract(bradley_terry_fit_lower_prior)
return(faceoff_samples_lower_prior)
}

faceoff_samples_lower_prior <- lower_prior_fit(faceoff_data)
```

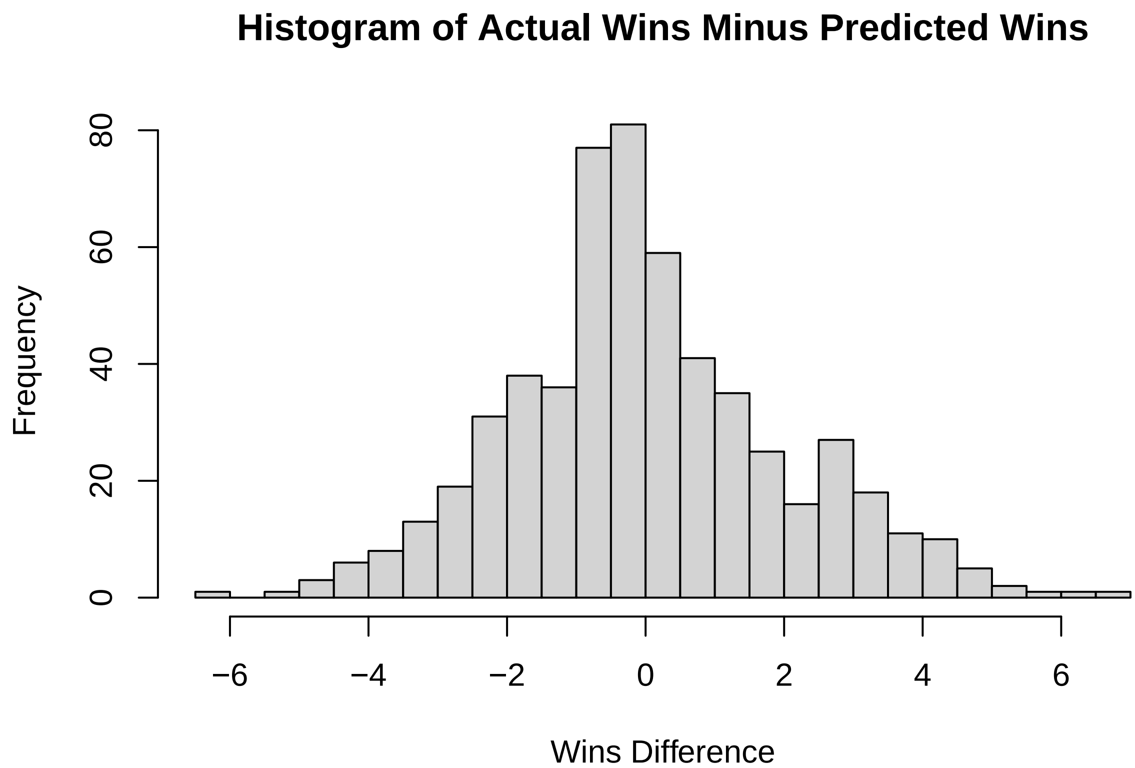
We can consider various posterior retrodictive summaries to compare how well our observations simulated from theta parameter samples compare to the observed data.

```
y_post_pred_lower_prior <- faceoff_samples_lower_prior$y_pred
sum(y_post_pred_lower_prior)/(4000 * nrow(faceoffs))
```

```
## [1] 0.5126059
```

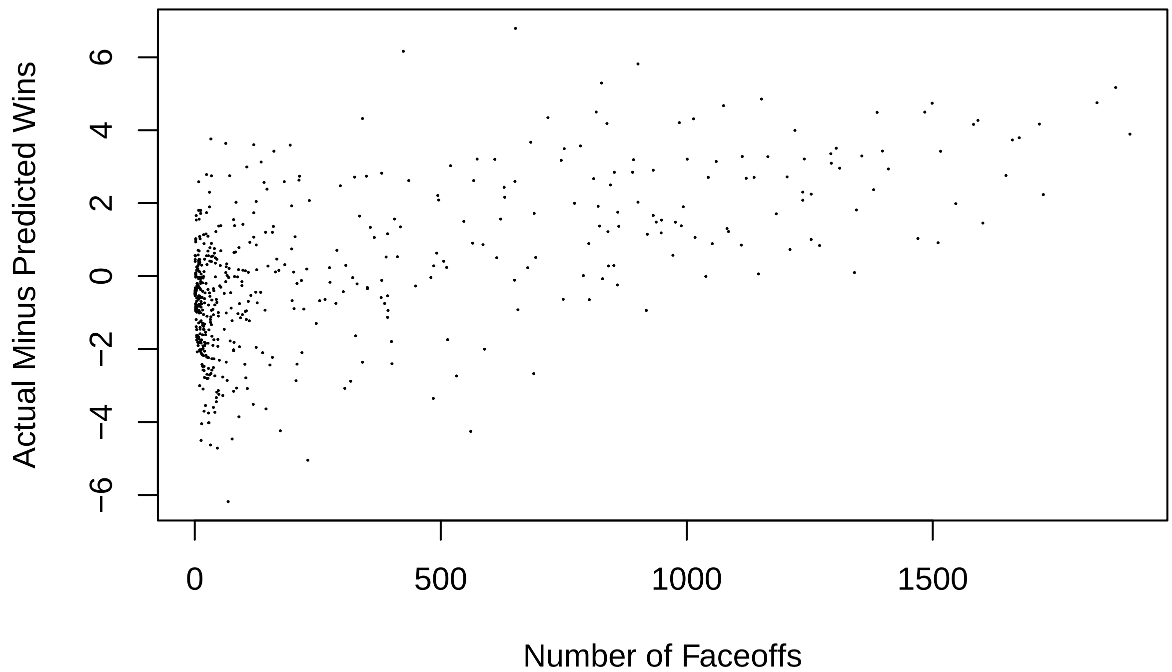
```
player_wins_lower_prior <- bind_cols(
  (faceoffs %>% select(c(player_1, player_2))),"player_1_wins" =
    apply(y_post_pred_lower_prior, 2, function(x) sum(x))) %>%
  mutate("player_2_wins" = 4000 - player_1_wins)
player_wins_lower_prior <- player_wins_lower_prior %>%
  pivot_longer(cols = c(player_1_wins, player_2_wins)) %>%
  mutate(player = ifelse(name == "player_1_wins", player_1, player_2)) %>%
  select(-c(player_1, player_2))

player_wins_summary_lower_prior <- player_wins_lower_prior %>%
  group_by(player) %>% summarize(n_pred_wins = sum(value)/4000) %>%
  left_join(unique_faceoff_player %>%
              select(c(fullname, value, n_wins, n, prop)), by =
              c("player" = "value"))
```

```
hist((player_wins_summary_lower_prior$n_wins -
        player_wins_summary_lower_prior$n_pred_wins),
     main = "Histogram of Actual Wins Minus Predicted Wins",
     xlab = "Wins Difference", breaks = 30)
```



The pattern persists for the most part

```
plot(player_wins_summary_lower_prior$n,
     (player_wins_summary_lower_prior$n_wins -
        player_wins_summary_lower_prior$n_pred_wins),
      xlab = "Number of Faceoffs", ylab = "Actual Minus Predicted Wins",
     pch = 20, cex = 0.1)
```



We alter our model so that our prior for our player coefficient is conditional on the number of faceoffs taken. Considering that we see a linear pattern in the residuals, a normal model for n_faceoffs taken may be appropriate:

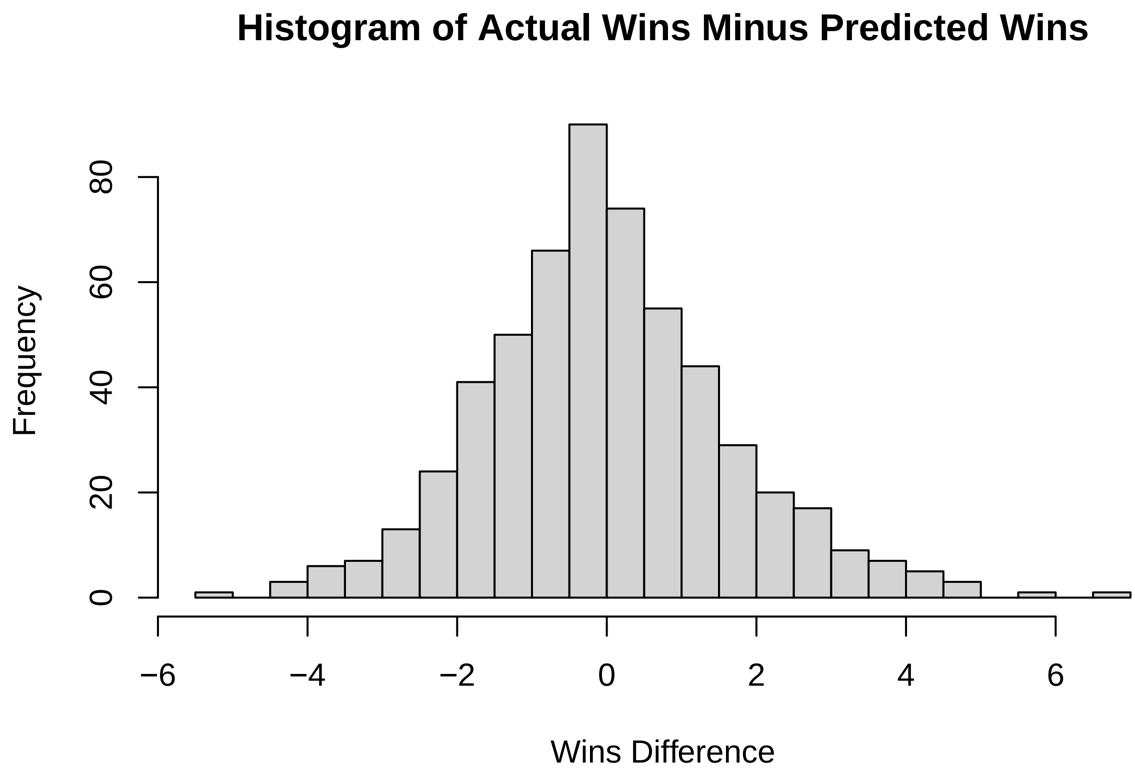
```
player_wins_summary <- player_wins_summary %>% arrange(desc(n)) %>%
  mutate(scaled_n = (n - (max(n)/2))/(max(n)/2))

faceoff_data_conditional <- list(
  "N" = nrow(faceoffs), "K" = nrow(unique_faceoff_player),
  "y" = rep(as.integer(1),nrow(faceoffs)),
  "player_1" = as.integer(faceoff_matrix_maker$player_1_index),
  "player_2" = as.integer(faceoff_matrix_maker$player_2_index),
  "N_taken" = player_wins_summary$scaled_n
)
```

```
conditional_fit <- function(dat){
bradley_terry_fit_conditional <- stan(model_code = "
data{
  int N;
  int K;
  array[N] int <lower=0, upper=1> y;
  array[N] int <lower=1, upper=K> player_1;
  array[N] int <lower=1, upper=K> player_2;
  vector<lower=-1, upper=1>[K] N_taken;
}
parameters{
  vector[K] theta;
```

```
  real beta;
}
transformed parameters{
  vector[K] theta_prior = beta*N_taken;
}
model{
  beta ~ normal(0.33,0.1);
  theta ~ normal(theta_prior, 0.33);
  y ~ bernoulli_logit(theta[player_1] - theta[player_2]);
}
generated quantities{
  array[N] int y_pred = bernoulli_logit_rng(theta[player_1] - theta[player_2]);
}
", data = dat, cores = mc.cores)

faceoff_samples_conditional <- rstan::extract(bradley_terry_fit_conditional)
return(faceoff_samples_conditional)
}
faceoff_samples_conditional <- conditional_fit(faceoff_data_conditional)
y_post_pred_conditional <- faceoff_samples_conditional$y_pred
sum(y_post_pred_conditional)/(4000 * nrow(faceoffs))
```

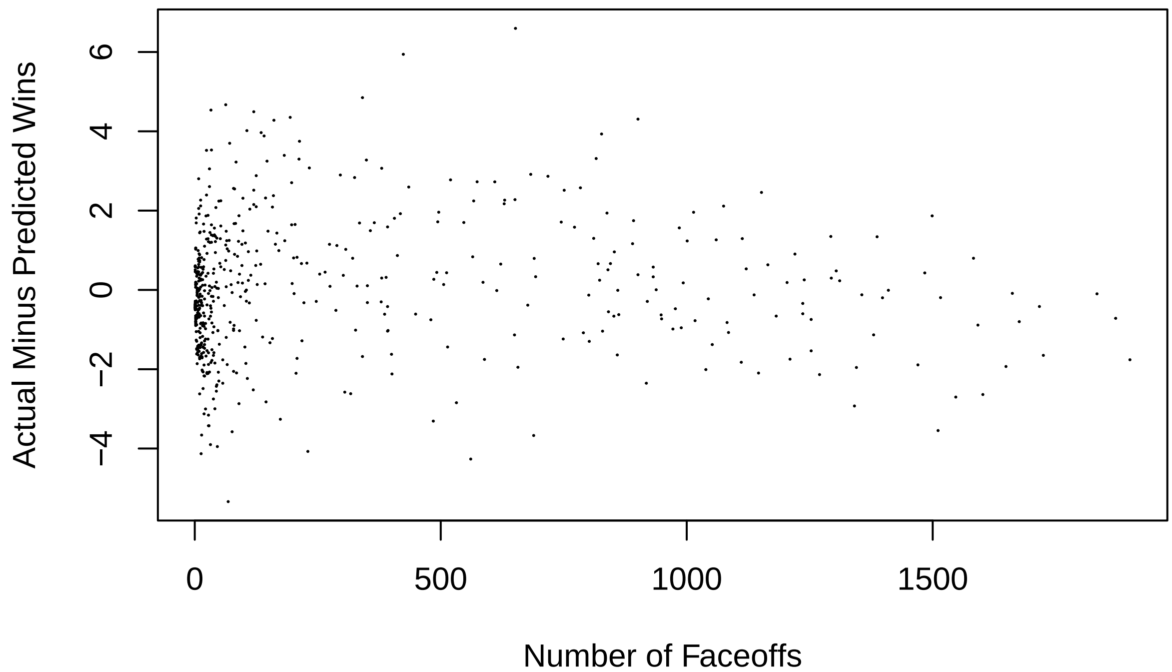
```
## [1] 0.5132783
```

```
player_wins_conditional <- bind_cols(
  (faceoffs %>% select(c(player_1, player_2))),
  "player_1_wins" = apply(y_post_pred_conditional, 2,function(x) sum(x))) %>%
  mutate("player_2_wins" = 4000 - player_1_wins)
player_wins_conditional <- player_wins_conditional %>%
  pivot_longer(cols = c(player_1_wins, player_2_wins)) %>%
  mutate(player = ifelse(name == "player_1_wins", player_1, player_2)) %>%
  select(-c(player_1, player_2))

player_wins_summary_conditional <- player_wins_conditional %>%
  group_by(player) %>% summarize(n_pred_wins = sum(value)/4000) %>%
  left_join(unique_faceoff_player %>%
              select(c(fullname, value, n_wins, n, prop)), by =
              c("player" = "value"))

hist((player_wins_summary_conditional$n_wins -
        player_wins_summary_conditional$n_pred_wins),
     main = "Histogram of Actual Wins Minus Predicted Wins",
     xlab = "Wins Difference",
     breaks = 30)
```



The residuals are in the same range but no longer have a pattern of underestimating the best players:

```
plot(player_wins_summary_conditional$n,
(player_wins_summary_conditional$n_wins -
        player_wins_summary_conditional$n_pred_wins),
 xlab = "Number of Faceoffs", ylab = "Actual Minus Predicted Wins",
     pch = 20, cex = 0.1)
```

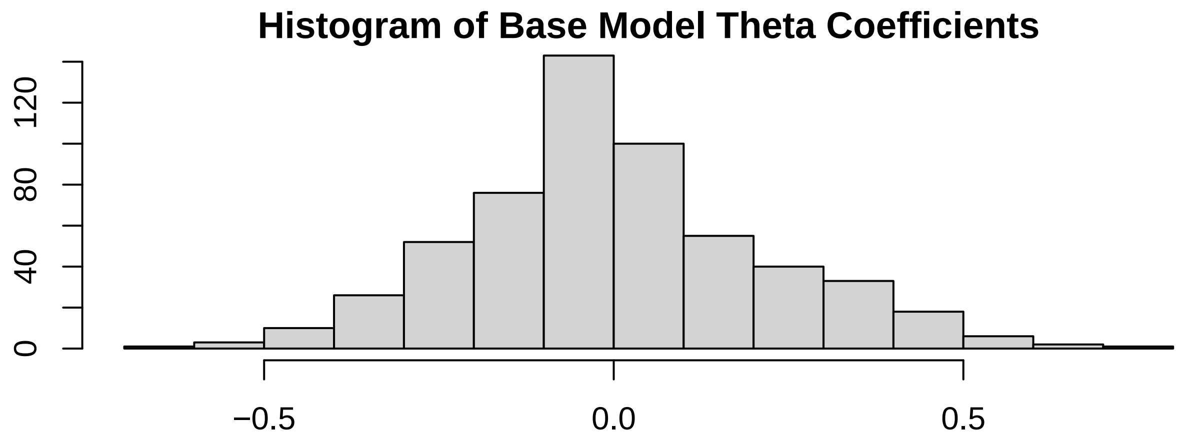


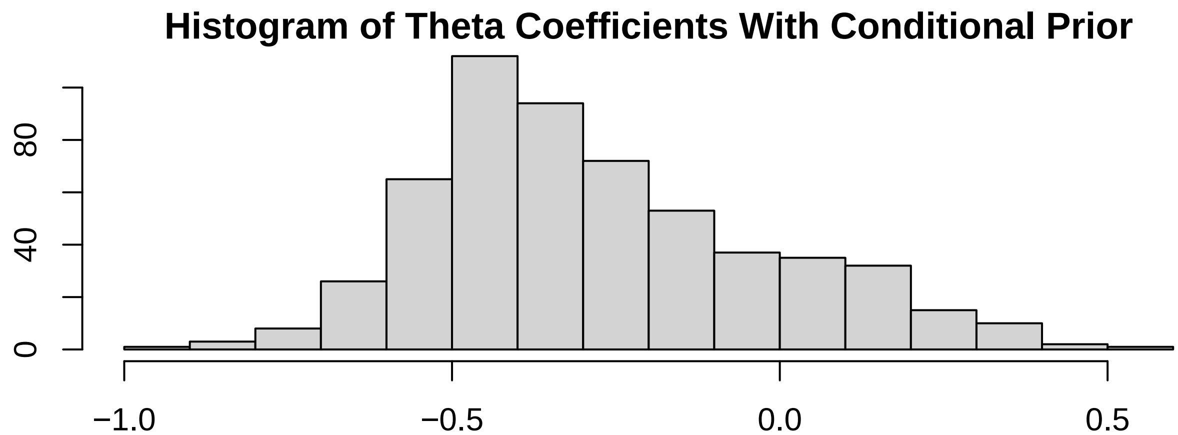
The parameters are no longer centered at zero due to our prior

```
cond_mean <- mean(faceoff_samples_conditional$theta)
cond_mean
```

```
## [1] -0.2799429
```

```
par(mar = c(2,2,1,1))
par(mfrow = c(2,1))
hist(apply(faceoff_samples$theta, 2, function(x) mean(x)),
     main = "Histogram of Base Model Theta Coefficients")
hist(apply(faceoff_samples_conditional$theta, 2, function(x) mean(x)),
     main = "Histogram of Theta Coefficients With Conditional Prior")
```





Recenter the parameters around the mean strength

```r
faceoff_quantiles_conditional <-
  apply(faceoff_samples_conditional$theta - cond_mean, 2, function(x)
    quantile(x, probs = seq(0.1,0.9,0.1)))

test <- unique_faceoff_player %>% bind_cols(t(faceoff_quantiles_conditional))

a3 <- exp(faceoff_samples_conditional$theta - cond_mean)/(1 +
exp(faceoff_samples_conditional$theta - cond_mean))
faceoff_percent_quantiles_conditional <- apply(a3, 2, function(x)
  quantile(x, probs = seq(0.1,0.9,0.1)))
print(test)
```

```
## # A tibble: 566 x 17
##      value n_wins     n  prop index fullname     shootsCatches team  `10%` `20%`
##      <dbl>  <int> <int> <dbl> <dbl> <chr>        <chr>         <chr> <dbl> <dbl>
##  1 8477934   1047  1901 0.551     1 LEON.DRAISA~ L             EDM   0.450 0.473
##  2 8471675   1090  1872 0.582     2 SIDNEY.CROS~ L             PIT   0.583 0.605
##  3 8476389   1074  1834 0.586     3 VINCENT.TRO~ R             NYR   0.584 0.609
##  4 8475745    881  1725 0.511     4 CHARLIE.COY~ R             BOS   0.312 0.335
##  5 8476468    973  1717 0.567     5 J.T..MILLER  L             VAN   0.517 0.544
##  6 8475158    897  1676 0.535     6 RYAN.O'REIL~ L             NSH   0.404 0.427
##  7 8471685    919  1662 0.553     7 ANZE.KOPITAR L             LAK   0.460 0.485
##  8 8480023    876  1649 0.531     8 ROBERT.THOM~ R             STL   0.341 0.364
##  9 8478493    796  1602 0.497     9 JOEL.ERIKSS~ L             MIN   0.236 0.259
## 10 8477496    884  1592 0.555    10 ELIAS.LINDH~ R             CGY   0.460 0.484
```

```
## # i 556 more rows
## # i 7 more variables: '30%' <dbl>, '40%' <dbl>, '50%' <dbl>, '60%' <dbl>,
## #   '70%' <dbl>, '80%' <dbl>, '90%' <dbl>
```

We see the necessity of modeling if we compare the difference between the empirical percentage of faceoffs won and the percentage won implied by our model:

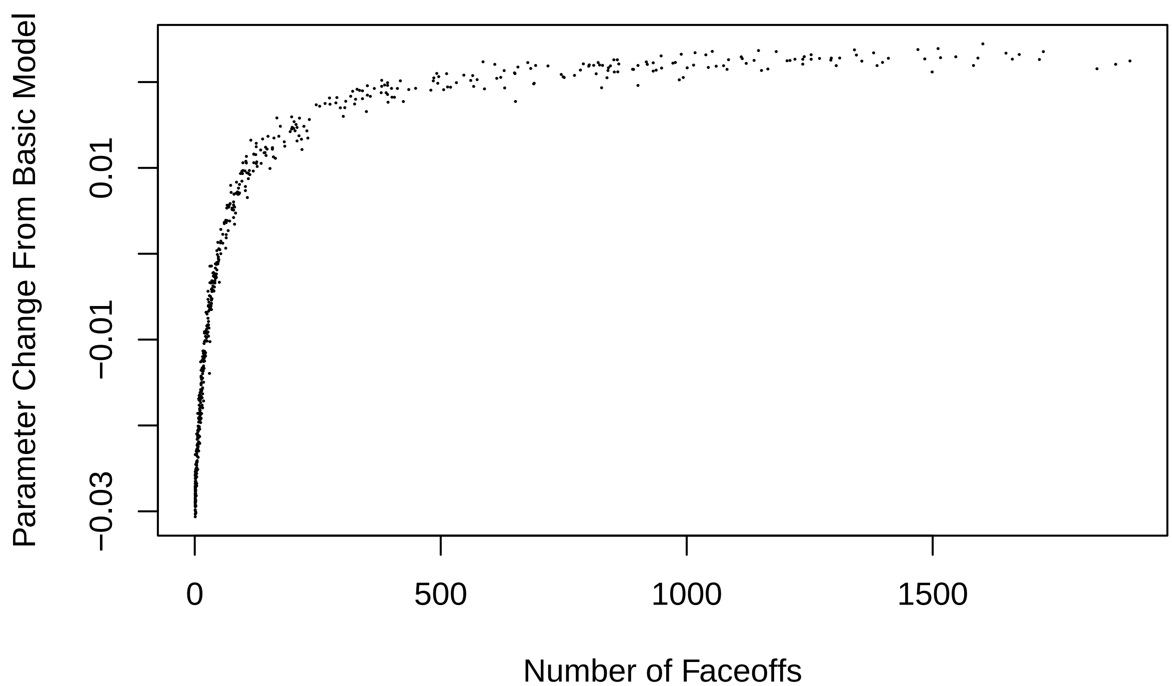
```
unique_faceoff_percents_conditional <- unique_faceoff_player %>% bind_cols(t(faceoff_percent_quantiles_c
  mutate(diff = `50%` - prop)
```

Under this model, the strongest players appear even stronger, and have a better chance of winning a faceoff against the perfect average player than under the previous model that underrated them.

```
unique_faceoff_conditional_percents <- unique_faceoff_player %>%
  bind_cols(t(faceoff_percent_quantiles[c(1,5,9),])) %>%
  rename_with(.cols = c(9:11), ~ paste(., "basic")) %>%
  bind_cols(t(faceoff_percent_quantiles_conditional[c(1,5,9),])) %>%
  rename_with(.cols = c(12:14), ~ paste(., "conditional")) %>%
  mutate(diff2 = `50% conditional` - `50% basic`)
```

Plot the difference in implied probability of winning a draw against an average player from the basic model to the conditional prior model against the number of faceoffs taken.

```
plot(unique_faceoff_conditional_percents$n,
     unique_faceoff_conditional_percents$diff2, pch = 20, cex = 0.1,
     xlab = "Number of Faceoffs", ylab = "Parameter Change From Basic Model")
```



Now we want to include whether a draw is on a player's strong or weak side to see how this affects the coefficients. We include a parameter for the number of faceoffs a player has taken and make this the prior for theta as in the last example.

```
faceoff_data_side <- list("N" = nrow(faceoffs),
"K" = nrow(unique_faceoff_player),
"y" = rep(as.integer(1),nrow(faceoffs)),
"player_1" = as.integer(faceoff_matrix_maker$player_1_index),
"player_2" = as.integer(faceoff_matrix_maker$player_2_index),
"player_1_side" = as.integer(faceoffs$p1_faceoff_side),
"player_2_side" = as.integer(faceoffs$p2_faceoff_side),
"N_taken" = faceoff_data_conditional$N_taken)

side_fit <- function(dat){
bradley_terry_side_fit <- stan(model_code = "
data{
  int N;
  int K;
  array[N] int <lower=0, upper=1> y;
  array[N] int <lower=1, upper=K> player_1;
  array[N] int <lower=1, upper=K> player_2;
  array[N] int <lower=1, upper=3> player_1_side;
  array[N] int <lower=1, upper=3> player_2_side;
  vector<lower=-1, upper=1>[K] N_taken;
}
parameters{
  vector[K] theta;
  vector[2] alpha;
  real beta;
}
transformed parameters{
  vector[3] alpha_trans = [alpha[1], 0, alpha[2]]';
  vector[K] theta_prior = beta*N_taken;
}
model{
  beta ~ normal(0.33,0.1);
  theta ~ normal(theta_prior, 0.33);
  alpha ~ normal(0, 0.33);
  y ~ bernoulli_logit((theta[player_1] + alpha_trans[player_1_side]) -
  (theta[player_2] + alpha_trans[player_2_side]));
}
generated quantities{
  array[N] int y_pred = bernoulli_logit_rng((theta[player_1] +
  alpha_trans[player_1_side]) - (theta[player_2] + alpha_trans[player_2_side]));
}
", data = dat, cores = mc.cores)

bradley_terry_side_samples <- rstan::extract(bradley_terry_side_fit)
return(bradley_terry_side_samples)
}
bradley_terry_side_samples <- side_fit(faceoff_data_side)

test <- apply(bradley_terry_side_samples$alpha, 2, function(x)
  quantile(x, probs = seq(0.1,0.9,0.1)))
```

```r
print(test)
```

```
##
##                [,1]         [,2]
##   10% -0.428852804 -0.173908584
##   20% -0.325092449 -0.074038773
##   30% -0.251162319  0.001547624
##   40% -0.183168712  0.065427118
##   50% -0.128564343  0.124492295
##   60% -0.066512916  0.186514070
##   70% -0.005120284  0.247697303
##   80%  0.068046547  0.319653240
##   90%  0.165866325  0.417575562
```

The quantiles are very wide but median on side value gives you a change in odds of success of exp(.13) = 1.16, equivalent to 53.2% probability of winning an otherwise 50-50 faceoff.

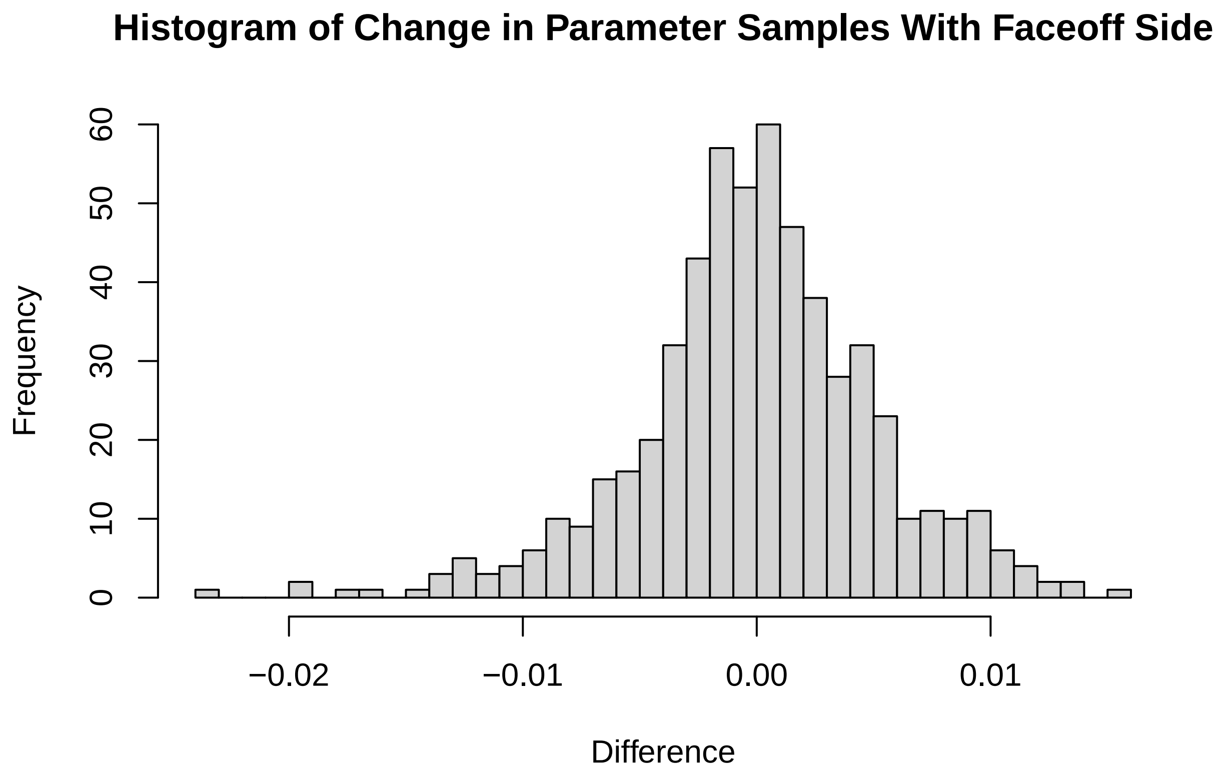
As before, the samples are not centered anymore:

```r
faceoff_quantiles_side <- apply(bradley_terry_side_samples$theta, 2, function(x)
  quantile(x, probs = seq(0.1,0.9,0.1)))
side_mean <- mean(bradley_terry_side_samples$theta)

a2 <- exp(bradley_terry_side_samples$theta - side_mean)/
  (1 +  exp(bradley_terry_side_samples$theta - side_mean))
faceoff_percent_quantiles_side <- apply(a2, 2, function(x)
  quantile(x, probs = seq(0.1,0.9,0.1)))

unique_faceoff_side_percents <- unique_faceoff_player %>%
  bind_cols(t(faceoff_percent_quantiles_conditional[c(1,5,9),])) %>%
  rename_with(.cols = c(9:11), ~ paste(., "no_side")) %>%
  bind_cols(t(faceoff_percent_quantiles_side[c(1,5,9),])) %>%
  rename_with(.cols = c(12:14), ~ paste(., "with_side")) %>%
  mutate(diff2 = `50% with_side` - `50% no_side`)

mean(abs(unique_faceoff_side_percents$diff2))
```

```
## [1] 0.003831033
```

So, only a slight average change in parameter sample mean from the model that didn't take faceoff side into consideration.

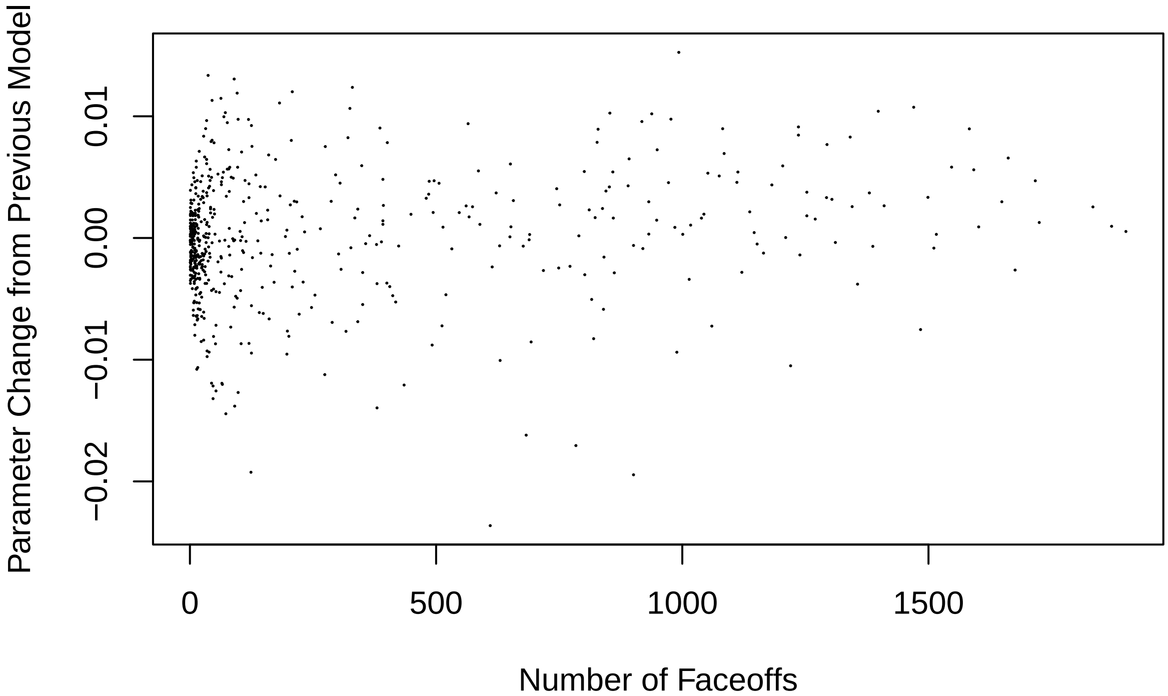
```r
hist(unique_faceoff_side_percents$diff2, breaks = 30,
     main = "Histogram of Change in Parameter Samples With Faceoff Side",
     xlab = "Difference")
```



Most players only see a very slight change in their faceoff strength with only a few seeing a change in over 1%

No strong pattern here:

```
plot(unique_faceoff_side_percents$n, unique_faceoff_side_percents$diff2,
     xlab = "Number of Faceoffs", ylab = "Parameter Change from Previous Model",
     pch = 20, cex = 0.1)
```




```
y_post_pred_side <- bradley_terry_side_samples$y_pred
```

```
sum(y_post_pred_side)/(4000 * nrow(faceoffs))
```

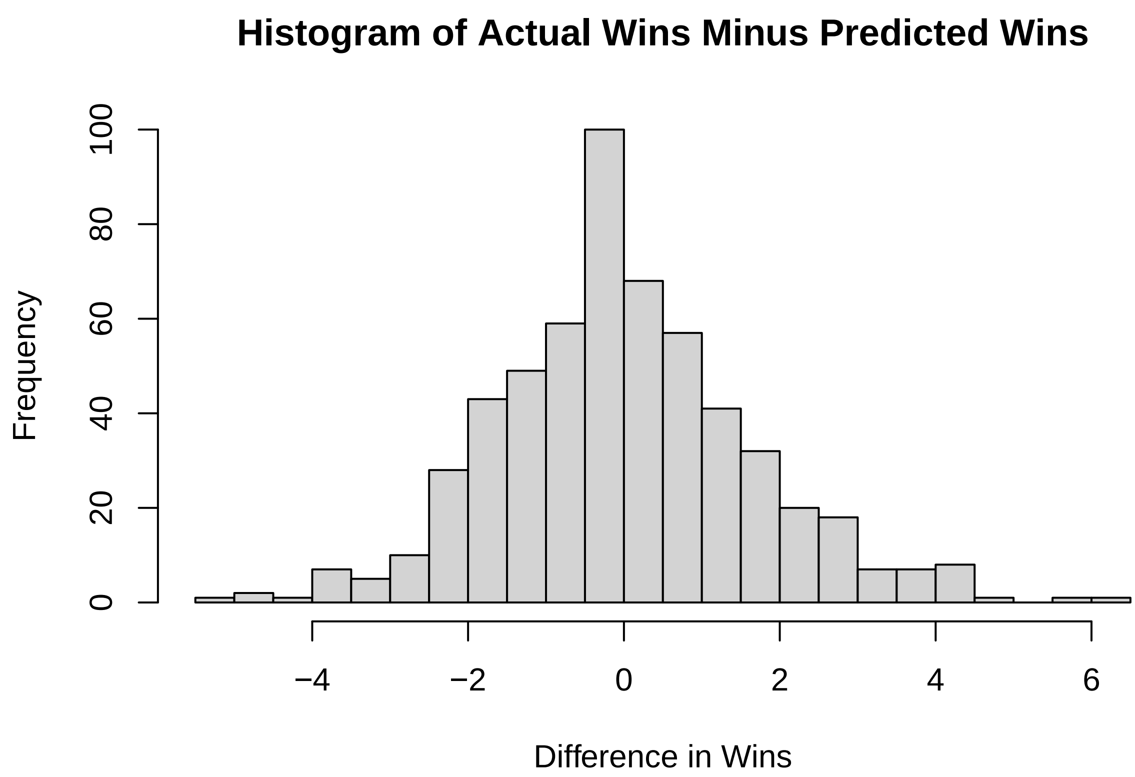
```
## [1] 0.516188
```

The model is making slightly more correct predictions than the previous model

```
player_wins_side <-
  bind_cols((faceoffs %>% select(c(player_1, player_2))),
            "player_1_wins" = apply(y_post_pred_side, 2, function(x) sum(x))) %>%
  mutate("player_2_wins" = 4000 - player_1_wins)
player_wins_side <- player_wins_side %>%
  pivot_longer(cols = c(player_1_wins, player_2_wins)) %>%
  mutate(player = ifelse(name == "player_1_wins", player_1, player_2)) %>%
  select(-c(player_1, player_2))

player_wins_summary_side <- player_wins_side %>%
  group_by(player) %>% summarize(n_pred_wins = sum(value)/4000) %>%
  left_join(unique_faceoff_player %>%
              select(c(fullname, value, n_wins, n, prop)), by =
              c("player" = "value"))
```

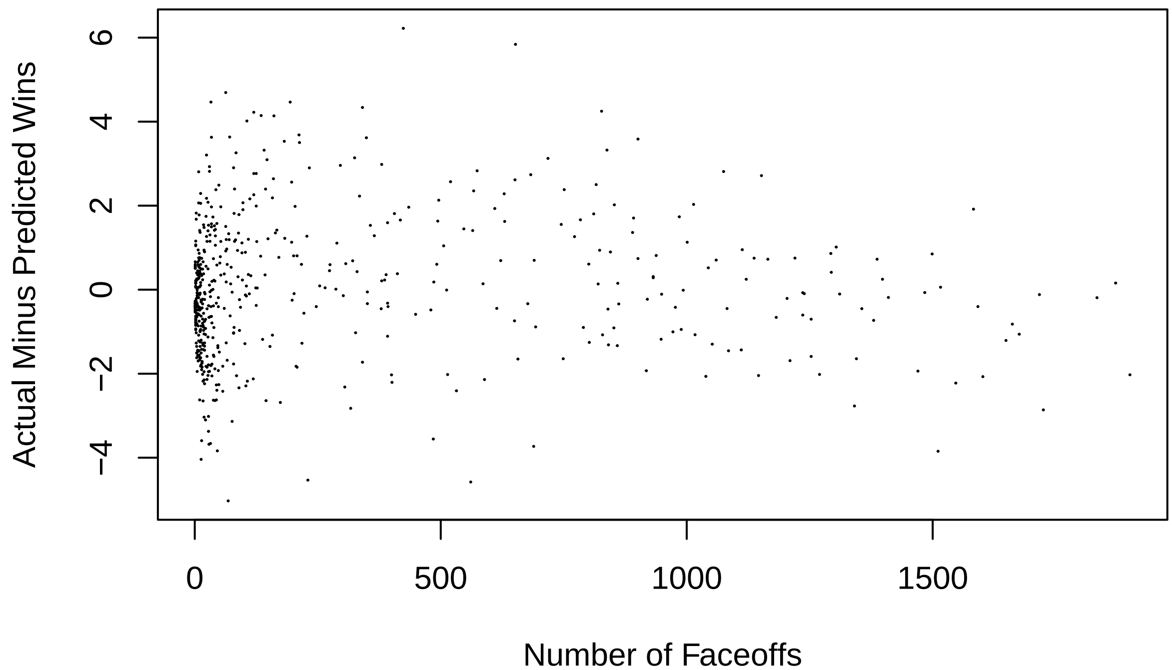
Looks quite similar to conditional model plot:

```r
hist((player_wins_summary_side$n_wins - player_wins_summary_side$n_pred_wins),
     main = "Histogram of Actual Wins Minus Predicted Wins",
     xlab = "Difference in Wins",
     breaks = 30)
```



Again quite similar

```r
plot(player_wins_summary_side$n, (player_wins_summary_side$n_wins - player_wins_summary_side$n_pred_wins
      xlab = "Number of Faceoffs", ylab = "Actual Minus Predicted Wins",
     pch = 20, cex = 0.1)
```



Finally satisfied with our model, we plot a player's predicted faceoff percentage against a perfectly average opponent accounting for any advantage in the side of the ice. Plotting the 10th to 90th percentiles of parameter samples for each player, with darker shades of red lying closer to the median, is an effective way to convey the uncertainty in a player's ability and in their rank compared to their peers. Players who took many faceoffs have narrower bands, while players who took few faceoffs have a very wide range in which their true percentage might lie.

```
Raw_vs_modeled_comparison <-
  bind_cols((unique_faceoff_player %>% select(c(fullname, prop))),
          t(faceoff_percent_quantiles_side)) %>% arrange(`50%`)

Raw_vs_modeled_comparison$index <- as.numeric(1:nrow(Raw_vs_modeled_comparison))
Raw_vs_modeled_comparison <-   as.data.frame(apply(Raw_vs_modeled_comparison, 2,
                                    function(x) rep(x, each = 2)))
Raw_vs_modeled_comparison[,c(2:12)] <-
  apply(Raw_vs_modeled_comparison[,c(2:12)], 2, function(x) as.numeric(x))

Raw_vs_modeled_comparison <-
  Raw_vs_modeled_comparison %>% mutate(
    index = ifelse(as.numeric(rownames(.)) %% 2 == 0, index + 0.5, index - 0.5))
par(mar=c(4, 1, 1, 3))
plot(1, type = "n", xlim = c(0.4, 0.8), ylim = c(518,566),
      main = "Top Faceoff Players in 23/24 Season",
     xlab = "Implied Neutral Win Percentage", yaxt = "n")
polygon(c(Raw_vs_modeled_comparison$`10%`,rev(Raw_vs_modeled_comparison$`90%`)),
      c(Raw_vs_modeled_comparison$index,rev(Raw_vs_modeled_comparison$index)),
```

```
      col = alpha("coral", 0.2), border = NA)
polygon(c(Raw_vs_modeled_comparison$`20%`,rev(Raw_vs_modeled_comparison$`80%`)),
        c(Raw_vs_modeled_comparison$index,rev(Raw_vs_modeled_comparison$index)),
        col = alpha("coral", 0.4), border = NA)
polygon(c(Raw_vs_modeled_comparison$`30%`,rev(Raw_vs_modeled_comparison$`70%`)),
        c(Raw_vs_modeled_comparison$index,rev(Raw_vs_modeled_comparison$index)),
        col = alpha("coral", 0.6), border = NA)
polygon(c(Raw_vs_modeled_comparison$`40%`,rev(Raw_vs_modeled_comparison$`60%`)),
        c(Raw_vs_modeled_comparison$index,rev(Raw_vs_modeled_comparison$index)),
        col = alpha("coral", 0.8), border = NA)
lines(Raw_vs_modeled_comparison$`50%`, Raw_vs_modeled_comparison$index,
      col = "red")

plot_labels <-
  Raw_vs_modeled_comparison$fullname[as.numeric(
    rownames(Raw_vs_modeled_comparison)) %% 2 == 0]
for (i in 517:566){
  text(x = 0.5, y = i, label = plot_labels[i], cex = 0.5)
}
```

Top Faceoff Players in 23/24 Season
MICHAEL.MCLEOD
JEFF.CARTER
NICO.STURM
JORDAN.STAAL
JOHN.TAVARES
VINCENT.TROCHECK
SIDNEY.CROSBY
CLAUDE.GIROUX
JAMIE.BENN
TOMAS.HERTL
J.T..MILLER
NICO.HISCHIER
ANZE.KOPITAR
ELIAS.LINDHOLM
ALEKSANDER.BARKOV
DYLAN.LARKIN
JEAN−GABRIEL.PAGEAU
LUKE.GLENDENING
KEVIN.HAYES
LEON.DRAISAITL
WILLIAM.KARLSSON
BOONE.JENNER
STEVEN.STAMKOS
TYLER.SEGUIN
CHRISTIAN.DVORAK
ERIK.HAULA
JACK.DRURY
SEAN.COUTURIER
BO.HORVAT
RYAN.O'REILLY
ANTON.LUNDELL
AUSTON.MATTHEWS
DYLAN.STROME
SEBASTIAN.AHO
SEAN.MONAHAN
CHRIS.TIERNEY
TEDDY.BLUEGER
ADAM.HENRIQUE
ROOPE.HINTZ
PAVEL.ZACHA
SAM.CARRICK
NOEL.ACCIARI
NICHOLAS.PAUL
ROBERT.THOMAS
NICK.SUZUKI
BLAKE.WHEELER
ANDREW.COPP
CASEY.CIZIKAS
MATT.DUCHENE
PIERRE−EDOUARD.BELLEMARE
0.4
0.5
0.6
0.7
0.8
Implied Neutral Win Percentage