Broncos Technical Assesment

Ethan Straub

2025-07-01

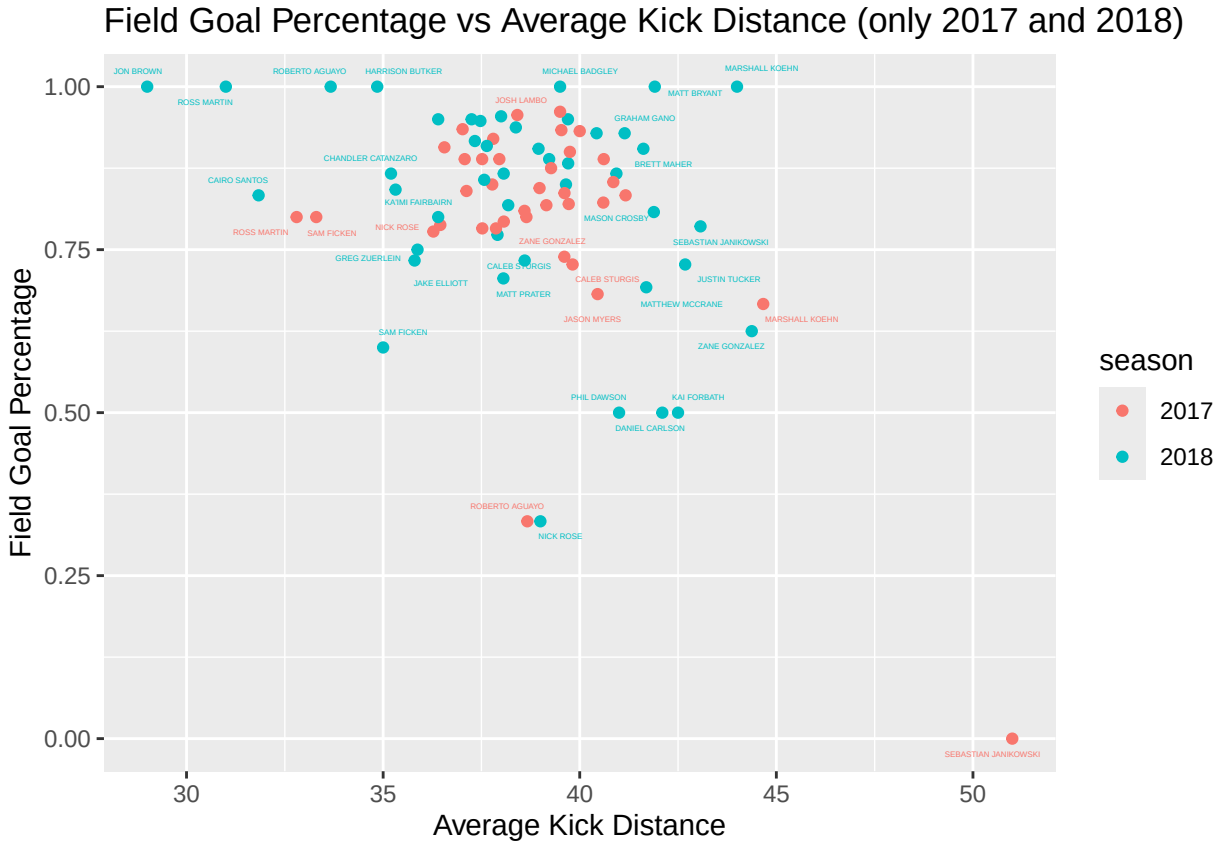
The code chunk below loads in the data and does transformations, summarizations, and filtering for the purposes of visualizations and model fitting. Check out the comments for more information. Players whose last kick was in 2018 and who attempted at least 5 field goals were included in the model.

```
library(dplyr)
# Loading in data
kickers = read.csv("kickers.csv")
data = read.csv("field_goal_attempts.csv")

# changing results to 0/1 variable instead of missed/made
data$field_goal_result = ifelse(data$field_goal_result == "Made", 1, 0)

# combining the kickers and field goal data into one data frame
fullData = left_join(data, kickers, by = "player_id")

# summarize data for each kicker
reducedData = fullData %>% group_by(player_name) %>% summarize(nkicks = n(),
                                                               fgpercent = mean(field_goal_result),
                                                               lastkickYear = max(season),
                                                               meanDistance = mean(attempt_yards))

# combine the summarized data with the combined data
fullData = left_join(fullData, reducedData, by = "player_name")

# create a vector of players that will be included in the model.
#Only players whose last field goal was in 2018 and have kicked at least 5 field goals are included
currentPlayers = reducedData$player_name[reducedData$lastkickYear>2017 & reducedData$nkicks > 4]

# use the chosen names to filter data
fullData_filtered = fullData %>% filter(player_name %in% currentPlayers)

# summarize data of the chosen players for visualizations
reducedData_filtered = fullData_filtered %>% group_by(player_name) %>%
  summarize(nkicks = n(), fgpercent = mean(field_goal_result),
            lastkickYear = max(season), meanDistance = mean(attempt_yards))
semireducedData_filtered = fullData_filtered %>%
  group_by(player_name, season) %>% summarize(nkicks = n(),
                                              fgpercent = mean(field_goal_result),
                                              lastkickYear = max(season),
                                              meanDistance = mean(attempt_yards))
```

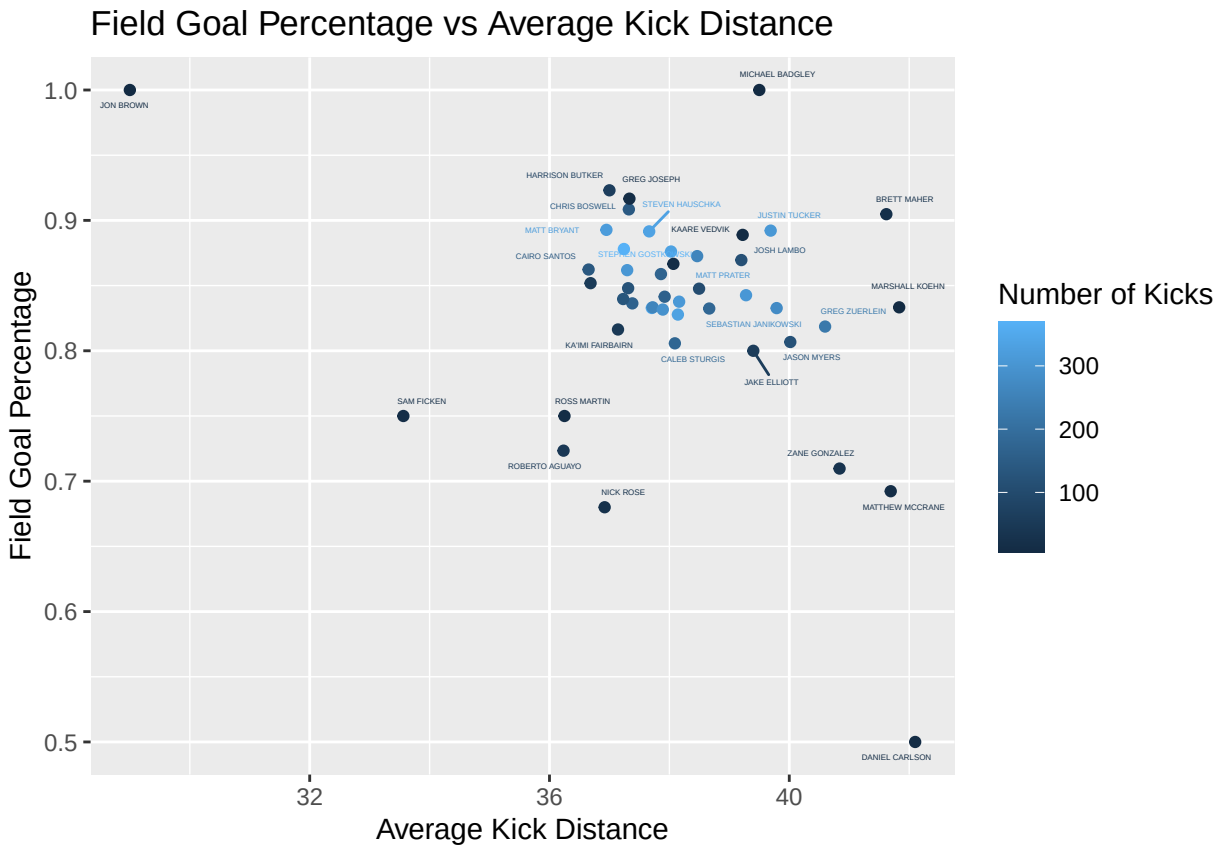
These next two graphs are visualizations of each chosen kicker's performance. The first graph uses data from only 2017 and 2018. The second graph uses data from all years in the dataset. These graphs were used to get

a general sense of the abilities of these kickers and to make sure the results of my model made sense.

```
# graph basics
library(ggplot2)
library(ggrepel)
only2018 = semireducedData_filtered %>% filter(season >= 2017)
only2018$season = as.factor(only2018$season)
graph1 = ggplot(only2018, aes(x = meanDistance, y = fgpercent, color = season)) +
  geom_point() + geom_text_repel(aes(label = player_name), size = 1) +
  labs(title = "Field Goal Percentage vs Average Kick Distance (only 2017 and 2018)",
       x = "Average Kick Distance",
       y = "Field Goal Percentage")
graph1
```




```
graph2 = ggplot(reducedData_filtered, aes(x = meanDistance, y = fgpercent,
                                          color = nkicks)) + geom_point() +
  geom_text_repel(aes(label = player_name), size = 1) +
  labs(title = "Field Goal Percentage vs Average Kick Distance",
       x = "Average Kick Distance", y = "Field Goal Percentage",
       color = "Number of Kicks")
graph2
```



The code chunk below sets up the imputs to the stan() function that runs a Hierarchical Bayesian State-Space model.

```
# removing unneeded columns from the data
df_filtered = fullData_filtered %>% select(week, attempt_yards, player_name, field_goal_result)

# creating an id for each player
df_filtered$id = as.integer(factor(df_filtered$player_name))

# fitting state space model
library(rstan)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

# model prep
file_code = "
data {
  int<lower=1> N;                  // total number of observations
  int<lower=1> K;                  // number of kickers
  int<lower=1> T[K];               // number of kicks per kicker
  int<lower=0, upper=1> y[N];      // miss = 1, make = 0
  vector[N] distance;              // kick distances
  int<lower=1, upper=K> kicker[N]; // kicker ID for each observation
}

parameters {
  real beta;                       // global distance effect
```

```
  real mu_alpha;                    // mean initial ability
  real<lower=1e-2> tau_alpha;       // SD of initial ability
  real<lower=0> sigma_alpha;        // SD of random walk process
  vector[K] alpha1;
  vector[sum(T)] alpha_raw;         // raw latent ability innovations
}

transformed parameters {
  vector[sum(T)] alpha;             // latent ability trajectories
  {
    int pos = 1;
    for (k in 1:K) {
      alpha[pos] = alpha1[k];  // initial ability
      for (t in 2:T[k]) {
        alpha[pos + t - 1] = alpha[pos + t - 2] + sigma_alpha * alpha_raw[pos + t - 1];
      }
      pos += T[k];
    }
  }
}

model {
  beta ~ normal(-0.1, 0.5);
  mu_alpha ~ normal(0, 1);
  tau_alpha ~ normal(0, 2);
  sigma_alpha ~ normal(0, 1);
  alpha1 ~ normal(mu_alpha, tau_alpha);
  alpha_raw ~ normal(0, 1);

  for (n in 1:N) {
    y[n] ~ bernoulli_logit(alpha[n] + beta * distance[n]);
  }
}
"

# rearranging data so that kickers are arranged together alphabetically
df = df_filtered %>% arrange(id)

# creating a variable that is the number of kicks that a player has made
T_k = df %>% count(id) %>% pull(n)

# creating a list for stan() input
stan_data <- list(
  N = nrow(df),
  K = length(unique(df$id)),
  T = T_k,
  y = df$field_goal_result,
  distance = as.vector(scale(df$attempt_yards)),
  kicker = df$id
)
```

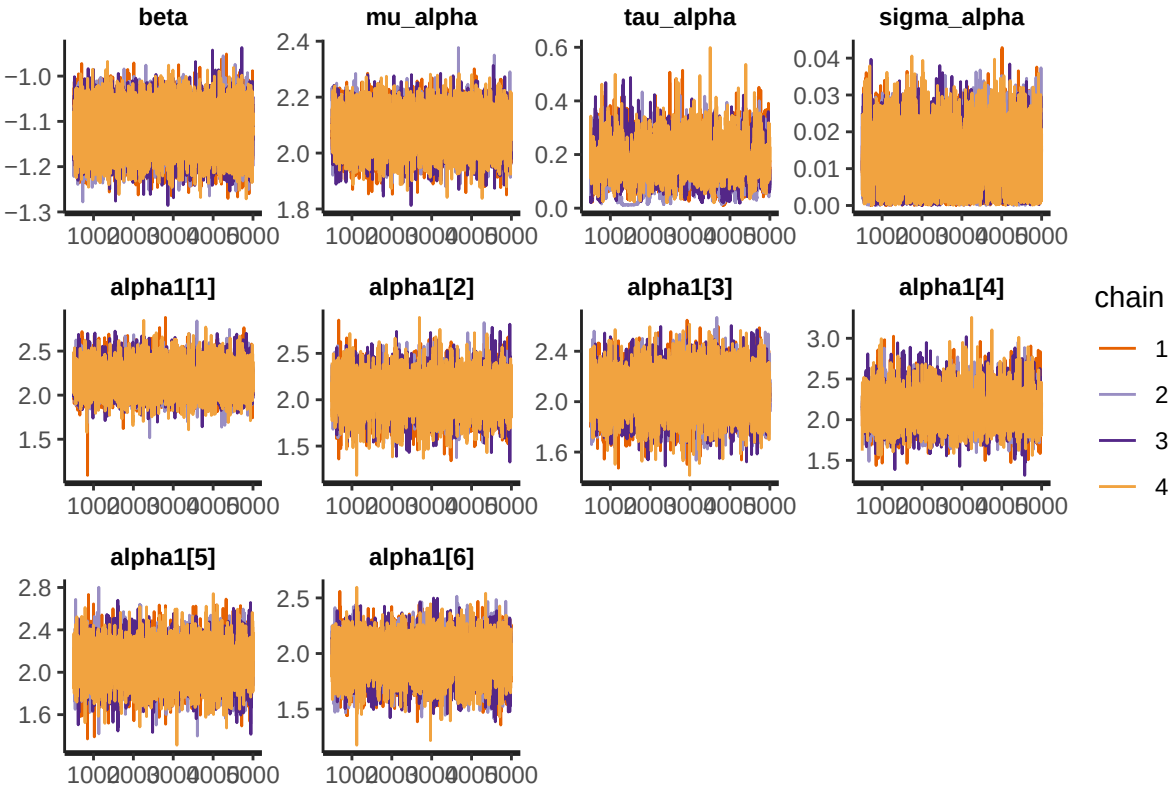
The code chunk below runs the model

```
# takes arouns 15 minutes to run
fit = stan(
  model_code = file_code,
  data = stan_data,
  iter = 5000,
  warmup = 500,
  chains = 4,
  seed = 50,
  control = list(adapt_delta = 0.95)
)
```

This code chunk evaluates evaluates the MCMC

```
traceplot(fit)
```

```
## 'pars' not specified. Showing first 10 parameters by default.
```




```
summary(fit)
```

The code chunk below uses fitted values to extract the latent state of each player at the time of their last kick.

```
# summary(fit)
posterior = extract(fit)
T_k = stan_data$T
alpha = posterior$alpha
last_pos = cumsum(T_k)

last_alpha = alpha[,last_pos]
la = alpha[,last_pos-10]
```

The code chunk below takes the average simulated latent state for all simulations for all kickers.

```
pred = colMeans(last_alpha)
predictions4 = data.frame(pred, currentPlayers)
```

The code chunk below assembles the leaderboard.csv file. Each player's rating is the mean of all MCMC samples for the hidden state at the time of their last kick.

```
# get player name and id columns in a data frame
leaderboard = distinct(data.frame(player_id = fullData$player_id, player_name = fullData$player_name))

# filter out unused players
leaderboard = leaderboard %>% filter(player_name %in% currentPlayers)

# join player rankings to the data frame
colnames(predictions4) = c("rating", "player_name")
leaderboard = left_join(leaderboard, predictions4, by = "player_name")

# add rank to the data frame
rank = 1:nrow(leaderboard)
leaderboard = leaderboard %>% arrange(-rating)
leaderboard = cbind(leaderboard,rank)

write.csv(leaderboard, "leaderboard.csv")
```