Simulating NFL Playoff Chances

Noah Sturgeon, Chase Bishop, Carter Nault

Project Introduction

Each team in the National Football League plays 17 games, and teams that either win their division or win enough games to be placed in the wild card are sent to the postseason. Teams that win early in the season are more likely to reach the postseason, and so our goal is to estimate playoff probabilities of each team through an incomplete season of the NFL, specifically through week 13 of the current 2025 season. We accomplished this by simulating results of each unplayed game in the season with a calculated win probability for the home team in each game. This project is a recreation of the simulations done by Football Sensei, with the major difference being that our calculated win probability uses Elo and a variable home field advantage, while their calculated win probability uses Elo and a constant home field advantage.

For the sake of this paper, Elo is a numerical system that attempts to quantify a contestant's skill level in competitive games with 1v1 matches. The Elo system accomplishes this in two steps. First, each contestant is given a starting number. Then, played game data is used to iteratively update the Elo system. Each team will gain an amount of points for winning each game, and lose an amount of points for losing each game. The exact formula used is up to the implementer, and will be discussed further on in this paper.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- 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
```

```
library(nflreadr)
set.seed(400)
```

To get the schedule data needed for whatever week we are simulating, we take NFL schedule data from the nflreadr package and turn it into a dataframe that has the week's schedule for each team with the home team for the scheduled game and if the team won, lost, tied, or did not play.

```
schedule_by_week <- function(week_num) {
  schedule <- load_schedules(2025)

  away_teams <- schedule |>
    select(week, away_team, home_team, result) |>
    filter(week == week_num) |>
    rename(team = away_team) |>
    mutate(win = case_when(result < 0 ~ 1,
                           result == 0 ~ 0.5,
                           .default = 0)) |>
```

```
    select(team, home_team, win)

  home_teams <- schedule |>
    select(week, away_team, home_team, result) |>
    filter(week == week_num) |>
    mutate(team = home_team) |>
    mutate(win = case_when(result > 0 ~ 1,
                           result == 0 ~ 0.5,
                           .default = 0))  |>
    select(team, home_team, win)

  week_schedule <- bind_rows(away_teams, home_teams)
  return(week_schedule)
}
```

To calculate the win probabilities for each home team, we used the following formula:

$P_{home} = \frac{1}{1+10^{-\frac{E_{home}+HFA+E_{away}}{S}}}$

Where E_home is the Elo of the home team and E_{away} is the Elo of the away team. HFA is the home field advantage for the home team. In Football Sensei's paper, he has a hard-coded value of 55 Elo for the home field advantage, but we changed the formula to be a variable that is different for each team. S is a scaling parameter. It can be set to whatever we want in theory, but we kept the Football Sensei's value of 400.

The away team's win probability is $1 - P_{home}$.

Instead of reusing the factors used in nfeloapp for Home Field Advantage, we incorporated elevation, crowd, and home record as those are stronger factors in home-field advantages. Home record in the form of a win percentage was weighted by 30% for historical performance and 70% for the past two seasons since recent performance is more accurate to a teams current home performance.

For crowd impact the factors that played into that was $\frac{Average\ Attendance - League\ Average\ Attendance}{\sigma}$ where finding the difference between average attendance to the league average tells us how much larger or smaller the crowd is compared to the league average and dividing by the standard deviation to scale the raw attendance to crowd point advantage.

For elevation we used the formula $\frac{Stadium\ Elevation - NFL\ Average\ Elevation}{1000 * 0.7}$, dividing by 1000 to scale down to a point advantage and then multiplying by 0.7 to a elevation point advantage.

Some factors provided by nfeloapp aren't useful, like "home bye" and "away bye" which doesn't happen every week, "division scaling" since division games are useful indicators but don't apply to every game, and "turf type" since every team has a lot of experience with both types and what type of turf is used didn't have much of an impact as in the past. Elevation and Crowd Impact, on the other hand, has a direct contribution to performance of the visiting team giving the home team a better advantage. Instead of multiplying the base by the percentage adjustments we directly solved for point advantage factor and added them to the HFA base of 2 for the "2 point home field advantage rule".

```
get_opposite_index_of_value <- function(input_vector, target_index) {
  result <- which(input_vector == input_vector[target_index])
  return(result[result != target_index])
}

calculate_win_probabilities <- function(team_names, home_teams, elo, hfa, S) {
  win_probabilities <- rep(0, length(team_names))
  for(team in 1:length(team_names)) {
    if (is.na(home_teams[team])) {
      win_probabilities[team] <- NA
```

```
    }
    else if(team_names[team] == home_teams[team]) {
      away_team_index <- get_opposite_index_of_value(home_teams, team)
      exponent <- (-1*(elo[team] + hfa[team] - elo[away_team_index]))/S
      win_probabilities[team] <- 1/(1+10^exponent)
      win_probabilities[away_team_index] <- 1-win_probabilities[team]
    }
  }
  return(win_probabilities)
}
```

Once we have the results from the simulated games, we update the Elo of each team according to the results. The formula for the change is as follows:

$Elo' = Elo + K * (Results - P_{team})$.

Where P_{team} is the win probability for that team. K is the sensitivity scalar. Football Sensei does not include what he uses for his sensitivity scalar on his website, so we had free range to choose whatever scale we wanted. We ended up choosing K=32.

```
elo_updater <- function(win_probabilities, actual_results, elos, sensitivity) {
  K <- sensitivity
  for (team in 1:length(elos)) {
    if (!is.na(win_probabilities[team])) {
      elos[team] <- elos[team] +
        K * (actual_results[team] - win_probabilities[team])
    }
  }
  return(elos)
}
```

```
standings_odds <- read_csv("2024_standings_2025_odds(in).csv")
```

```
## Rows: 32 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): TCode, Tm
## dbl (4): W, L, T, ProjW
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
hfa_data <- read_csv("NFL-HomeFieldAdvantage-Data.csv")
```

```
## New names:
## Rows: 32 Columns: 8
## -- Column specification
## ------------------------------------------------------- Delimiter: "," chr
## (2): Team Code, Team Name dbl (5): Record, Elevation, Crowd Impact, Normalizing
## Constant, Point Advantage lgl (1): ...8
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...8`
```

```
weights <- c(0.4, 0.6)
mean_elo <- 1000
scale <- 32
```

```
team_data <- standings_odds |>
  arrange(Tm) |>
  rename(Team_Name = Tm) |>
  mutate(Weighted_ProjW = weights[1] * W + weights[2] * ProjW) |>
  mutate(Elo = (Weighted_ProjW - 8.5) * scale + mean_elo) |>
  mutate(HFA = hfa_data$'Point Advantage') |>
  select(TCode, Team_Name, Elo, HFA)
```

Each team has a starting Elo which is a weighted average of last year's standings and the Vegas over-under line for wins from the preseason. Slightly more weight was placed on the over-under line to account for the changes in coaching and roster that an NFL team undergoes over the offseason, which would be considered in the preseason betting lines.

```
last_week <- 13

for (week in 1:last_week) {
  current_week_schedule <- schedule_by_week(week)
  current_week_schedule <- rename(current_week_schedule, TCode = team)
  sched_w_elo <- left_join(team_data, current_week_schedule)
  sched_w_elo$WP <- calculate_win_probabilities(sched_w_elo$TCode, sched_w_elo$home_team, sched_w_elo$E
  sched_w_elo$Elo <- elo_updater(sched_w_elo$WP, sched_w_elo$win, sched_w_elo$Elo, scale)
  team_data$Elo <- sched_w_elo$Elo
}
```

```
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
## Joining with `by = join_by(TCode)`
```

Using the win probability and Elo updater functions, the Elo of each team is updated for each week that has already been played in the season, which was up until week 13.

```
season_sim <- function() {
  current_week <- 14
  last_week <- 18

  sim_wins = data.frame(
    TCode = team_data$TCode,
    wins = rep(0, 32)
  )

  for (week in current_week:last_week) {
    schedule <- schedule_by_week(week) |>
      rename(TCode = team)
    schedule <- left_join(schedule, team_data, by = join_by(TCode))
    schedule$WP <- calculate_win_probabilities(schedule$TCode, schedule$home_team, schedule$Elo, schedul
```

```
    for (i in 1:length(unique(schedule$home_team))) { #Have to simulate for each game which is half the
      u <- runif(1)
      if (u < schedule$WP[i]) {
        #TCode (away team) wins
        sim_wins[sim_wins$TCode == schedule$TCode[i], "wins"] <- sim_wins[sim_wins$TCode == schedule$TC
      }
      else {
        #home team wins
        sim_wins[sim_wins$TCode == schedule$home_team[i], "wins"] <- sim_wins[sim_wins$TCode == schedul
      }
    }
  }

  return(sim_wins)
}
```

Each unplayed week is iterated over and simulated, and each unplayed game is simulated using the inverse CDF method of a Bernoulli trial, where p is the calculated win probability of the home team. If a randomly generated number is less than this value, a loss for the home team is simulated. Otherwise, a win is simulated.

```
last_week <- 13
actual_wins <- sapply(team_data$TCode, function(team) {
  sum(sapply(1:last_week, function(wk) {
    sched <- schedule_by_week(wk)
    row <- sched[sched$team == team, ]
    if(nrow(row) == 0) return(0)
    sum(row$win)
  }))
})
names(actual_wins) <- team_data$TCode

sim_wins <- season_sim()

sim_wins$total_wins <- sim_wins$wins + actual_wins[sim_wins$TCode]
```

```
AFC_North <- c("BAL","PIT", "CIN", "CLE")
AFC_East  <- c("NE", "BUF", "MIA", "NYJ")
AFC_South <- c("JAX", "IND", "HOU", "TEN")
AFC_West  <- c("KC", "LAC", "DEN", "LV")

NFC_North <- c("CHI", "GB", "DET", "MIN")
NFC_East  <- c("PHI", "DAL", "WAS", "NYG")
NFC_South <- c("TB", "CAR", "ATL", "NO")
NFC_West  <- c("LA", "SEA", "SF", "ARI")

divisions <- list(
  AFC_North = AFC_North, AFC_East = AFC_East, AFC_South = AFC_South, AFC_West = AFC_West,
  NFC_North = NFC_North, NFC_East = NFC_East, NFC_South = NFC_South, NFC_West = NFC_West
)

conference <- list(
  AFC = unlist(list(AFC_North, AFC_East, AFC_South, AFC_West)),
  NFC = unlist(list(NFC_North, NFC_East, NFC_South, NFC_West))
)
```

```r
nsims <- 100000
division_standings <- function(division_name, team_names, sim_wins_df) {
  df <- sim_wins_df[sim_wins_df$TCode %in% team_names, ]
  df$division <- division_name
  df <- df[order(-df$total_wins), ]
  rownames(df) <- NULL
  return(df)
}

simulate_once <- function() {
  sim_wins <- season_sim()

  sim_wins$total_wins <- sim_wins$wins + actual_wins[sim_wins$TCode]

  all_standings <- lapply(names(divisions), function(div) {
    division_standings(div, divisions[[div]], sim_wins)
  })
  names(all_standings) <- names(divisions)

  division_winners <- sapply(all_standings, function(df) df$TCode[1])

  AFC_divwinners <- division_winners[grepl("AFC", names(division_winners))]
  NFC_divwinners <- division_winners[grepl("NFC", names(division_winners))]

  AFC_nondiv <- setdiff(conference$AFC, AFC_divwinners)
  NFC_nondiv <- setdiff(conference$NFC, NFC_divwinners)

  AFC_wildcard_df <- sim_wins[sim_wins$TCode %in% AFC_nondiv, ]
  AFC_wildcard_df <- AFC_wildcard_df[order(-AFC_wildcard_df$total_wins), ][1:3, ]

  NFC_wildcard_df <- sim_wins[sim_wins$TCode %in% NFC_nondiv, ]
  NFC_wildcard_df <- NFC_wildcard_df[order(-NFC_wildcard_df$total_wins), ][1:3, ]

  seed_placement <- function(winner_teams, wildcard_df, sim_wins_df) {
    all_teams <- c(winner_teams, wildcard_df$TCode)
    out <- sim_wins_df[sim_wins_df$TCode %in% all_teams, ]
    out <- out[order(-out$total_wins), ]
    rownames(out) <- NULL
    return(out)
  }

  AFC_seeds <- seed_placement(AFC_divwinners, AFC_wildcard_df, sim_wins)
  NFC_seeds <- seed_placement(NFC_divwinners, NFC_wildcard_df, sim_wins)

  return(
    data.frame(
      conference = c(rep("AFC", 7), rep("NFC", 7)),
      seed = rep(1:7, 2),
      team = c(AFC_seeds$TCode, NFC_seeds$TCode)
    ) |>
    dplyr::left_join(
      sim_wins |> dplyr::select(TCode, total_wins),
      by = c("team" = "TCode")
```

```
    )
  )
}
```

```
results <- do.call(rbind, replicate(nsims, simulate_once(), simplify = FALSE))

### ---- SAVE AS CSV ----
write.csv(results, "playoff_sims_100k.csv", row.names = FALSE)
```

```
results <- read.csv("playoff_sims_100k.csv")
playoff_counts <- setNames(rep(0, length(team_data$TCode)), team_data$TCode)

for (i in 1:nsims) {
  playoffteams <- results$team[(1 + (i - 1) * 14):(i * 14)]
  playoff_counts[playoffteams] <- playoff_counts[playoffteams] + 1
}

playoff_probabilities <- playoff_counts / nsims
playoff_probabilities
```

```
##     ARI     ATL     BAL     BUF     CAR     CHI     CIN     CLE     DAL     DEN
## 0.00000 0.00291 0.65532 0.98021 0.29076 0.91910 0.01801 0.00010 0.15706 0.99969
##     DET      GB     HOU     IND     JAX      KC      LV     LAC      LA     MIA
## 0.43557 0.82108 0.70265 0.72146 0.80249 0.22929 0.00000 0.55644 0.95922 0.00182
##     MIN      NE      NO     NYG     NYJ     PHI     PIT      SF     SEA      TB
## 0.00019 0.99950 0.00000 0.00000 0.00000 0.95672 0.33302 0.82155 0.90929 0.72655
##     TEN     WAS
## 0.00000 0.00000
```

To get the playoff seeds, we take the wins from the simulated seasons, and use them to get the division leaders and wildcard spots for each season. Then, we take the count of times each team made the playoffs during all of the simulated seasons, and divide by the number of simulations, 100,000, to get the playoff probability.

```
playoff_df <- data.frame(
  team = names(playoff_probabilities),
  playoff_prob = playoff_probabilities,
  stringsAsFactors = FALSE
) %>%
  mutate(conference = ifelse(team %in% conference$AFC, "AFC", "NFC")) %>%
  arrange(conference, desc(playoff_prob))

AFC_probs <- playoff_df %>%
  filter(conference == "AFC") %>%
  select(AFC_team = team, AFC_prob = playoff_prob)

NFC_probs <- playoff_df %>%
  filter(conference == "NFC") %>%
  select(NFC_team = team, NFC_prob = playoff_prob)

PlayoffProb <- cbind(AFC_probs, NFC_probs)

rownames(PlayoffProb) <- NULL


PlayoffProb
```

```
##    AFC_team AFC_prob NFC_team NFC_prob
## 1       DEN  0.99969       LA  0.95922
## 2        NE  0.99950      PHI  0.95672
## 3       BUF  0.98021      CHI  0.91910
## 4       JAX  0.80249      SEA  0.90929
## 5       IND  0.72146       SF  0.82155
## 6       HOU  0.70265       GB  0.82108
## 7       BAL  0.65532       TB  0.72655
## 8       LAC  0.55644      DET  0.43557
## 9       PIT  0.33302      CAR  0.29076
## 10       KC  0.22929      DAL  0.15706
## 11      CIN  0.01801      ATL  0.00291
## 12      MIA  0.00182      MIN  0.00019
## 13      CLE  0.00010      ARI  0.00000
## 14       LV  0.00000       NO  0.00000
## 15      NYJ  0.00000      NYG  0.00000
## 16      TEN  0.00000      WAS  0.00000
```

This table shows the playoff probability, sorted into AFC and NFC, and sorts the probability from highest to lowest.

```r
constant_hfa <- 2
team_data$HFA <- rep(constant_hfa, 32)
nsims <- 100

results <- do.call(rbind, replicate(nsims, simulate_once(), simplify = FALSE))

playoff_counts <- setNames(rep(0, length(team_data$TCode)), team_data$TCode)

for (i in 1:nsims) {
  playoffteams <- results$team[(1 + (i - 1) * 14):(i * 14)]
  playoff_counts[playoffteams] <- playoff_counts[playoffteams] + 1
}

playoff_probabilities_const_hfa <- playoff_counts / nsims
knitr::kable(playoff_probabilities_const_hfa)
```

	x
ARI	0.00
ATL	0.00
BAL	0.73
BUF	0.98
CAR	0.27
CHI	0.91
CIN	0.00
CLE	0.00
DAL	0.18
DEN	1.00
DET	0.42
GB	0.77
HOU	0.68
IND	0.76
JAX	0.83
KC	0.16

	x
LV	0.00
LAC	0.59
LA	0.98
MIA	0.00
MIN	0.00
NE	1.00
NO	0.00
NYG	0.00
NYJ	0.00
PHI	0.96
PIT	0.27
SF	0.82
SEA	0.95
TB	0.74
TEN	0.00
WAS	0.00

The above table simulates a smaller number of games with home field advantage set to a constant 2. Results are very similar to the simulation with variable home field advantage, because of how far into the season the simulations take place. However, the variable HFA is better able to pick up on the nuances in the NFL schedule. For example, the Chiefs probability is slightly increased in the variable HFA, because of how much higher their HFA is over the constant and because they play several home games against high leverage opponents during the simulation (Texans, Chargers, and Broncos). We believe that variable HFA is a useful addition because of this.

References

Ho T, Carl S (2025). nflreadr: Download 'nflverse' Data. R package version 1.5.0.9000, https://nflreadr.nflverse.com.

Nfelo. (2021, December 20). A further exploration of home field advantage in the NFL. nfelo. https://www.nfeloapp.com/analysis/a-further-exploration-of-home-field-advantage-in-the-nfl/

Methodology: How football sensei calculates NFL odds. Methodology | How Football Sensei Calculates NFL Odds. (2025). https://footballsensei.com/methodology

Pro Football Stats, history, scores, standings, playoffs, Schedule & Records. Pro. (n.d.). https://www.pro-football-reference.com/