Simulating NFL Playoff Chances

Noah Sturgeon, Chase Bishop, Carter Nault

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.6
## ✓ forcats   1.0.1     ✓ stringr   1.6.0
## ✓ ggplot2   4.0.1     ✓ tibble    3.3.0
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.1
## ✓ purrr     1.2.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(nflreadr)
library(tinytex)
library(dplyr)
```

```
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. In football sensei's paper, he has a hard-coded value of 55 elo, but we changed the formula to be a variable that is different for each team. S is a scaling parameter. It can be whatever we want in theory, but we kept the football sensei's value of 400.

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

```
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)
}
```

After getting the win probabilities of all teams, we simulate a week's worth of games (aka 1 game per team that does not have a bye). We generate a Unif(0,1) random variable to simulate the outcome of a game. If the result is less that P_{home}, the home team wins. Otherwise, the away team wins.

```
week_simulator <- function(team_names, home_teams, win_probabilities) {
  game_results <- rep(-1, length(team_names))

  for(team in seq_along(team_names)) {
    if(!is.na(home_teams[team]) && team_names[team] == home_teams[team]) {
      u <- runif(1, min = 0, max = 1)
      away_team_index <- get_opposite_index_of_value(home_teams, team)

      if(u < win_probabilities[team]) {
        game_results[team] <- 1
        game_results[away_team_index] <- 0
      } else {
        game_results[team] <- 0
        game_results[away_team_index] <- 1
      }
    }
  }

  return(game_results)
}
```

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)
}
```

The schedule by week function takes NFL schedule data from the nflreadr package and takes the needed week number as an input, and outputs 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.

```
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$home_team, sched_
w_elo$Elo, sched_w_elo$HFA, 400)
  sched_w_elo$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 week schedule, 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.

```
simulate_season <- function(team_data, start_week = 14, end_week = 18,
                            S = 400, K = 32, sims = 1000) {

  teams <- team_data$TCode
  n_teams <- length(teams)

  sim_wins_mat <- matrix(0, nrow = n_teams, ncol = sims,
                     dimnames = list(teams, NULL))

  for (sim in 1:sims) {
    elos <- team_data$Elo
    hfa  <- team_data$HFA
    wins <- rep(0, n_teams)
    names(wins) <- teams

    for (week in start_week:end_week) {
      week_sched <- schedule_by_week(week) |> rename(TCode = team)
      sched <- left_join(team_data |> select(TCode), week_sched)
      sched$Elo <- elos[match(sched$TCode, teams)]
      sched$HFA <- hfa[match(sched$TCode, teams)]
      sched$WP <- calculate_win_probabilities(sched$TCode, sched$home_team, sched$Elo, sched$HF
A, S)
      results <- week_simulator(sched$TCode, sched$home_team, sched$WP)
      wins <- wins + ifelse(results == 1, 1, 0)
      elos <- elo_updater(sched$WP, results, elos, K)
    }

    sim_wins_mat[, sim] <- wins
  }

  sim_wins <- rowMeans(sim_wins_mat)
  names(sim_wins) <- teams   # ensure names match TCode
  return(sim_wins)
}

sim_wins <- simulate_season(team_data, start_week = 14, end_week = 18, sims = 10)
```

```
## 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)`
## 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)`
## 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)`
## 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)`
```

```
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("LAR", "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))
)
```

```
division_standings <- function(division_name, team_names, sim_wins){
  valid_teams <- team_names[team_names %in% names(sim_wins)]
  wins <- sim_wins[valid_teams]

  df <- data.frame(
    division = division_name,
    team = valid_teams,
    wins = wins,
    stringsAsFactors = FALSE
  )

  df <- df[order(-df$wins), ]
  rownames(df) <- NULL
  return(df)
}

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$team[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 <- sort(sim_wins[AFC_nondiv], decreasing = TRUE)[1:3]
NFC_wildcard <- sort(sim_wins[NFC_nondiv], decreasing = TRUE)[1:3]

seed_placement <- function(winners, wildcards, sim_wins){
  teams <- c(winners, names(wildcards))
  sort(sim_wins[teams], decreasing = TRUE)
}

AFC_seeds <- seed_placement(AFC_divwinners, AFC_wildcard, sim_wins)
NFC_seeds <- seed_placement(NFC_divwinners, NFC_wildcard, sim_wins)

AFC_divwinners
```

```
## AFC_North  AFC_East AFC_South  AFC_West
##     "BAL"      "NE"     "JAX"     "DEN"
```

```
NFC_divwinners
```

```
## NFC_North  NFC_East NFC_South  NFC_West
##      "GB"     "PHI"      "TB"     "SEA"
```

```
AFC_wildcard
```

```
## HOU  KC LAC
## 3.0 2.9 2.9
```

```
NFC_wildcard
```

```
## DET  SF DAL
## 3.1 2.6 2.5
```

```
AFC_seeds
```

```
## DEN JAX  NE HOU BAL  KC LAC
## 3.4 3.3 3.2 3.0 2.9 2.9 2.9
```

```
NFC_seeds
```

```
##  GB PHI DET  TB SEA  SF DAL
## 3.1 3.1 3.1 3.0 2.7 2.6 2.5
```

The playoffseed takes the wins from the simulated season and grabs the division leaders and wildcard spots and sorts them by wins and tiebreakers. After that these positions will be run through the weeksimulator again to get to the simulated super bowl and finally simulate the superbowl.

References

HFA Data 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/ (https://www.nfeloapp.com/analysis/a-further-exploration-of-home-field-advantage-in-the-nfl/)

Instead of reusing the factors used in nfeloapp, we incorporated elevation, crowd, and home record as those are stronger factors in home-field advantages. Home record was weighted by 30% for historical performance and 70% for the past two seasons since recent performance is more accurate to a teams performance. We found these factors to be stronger indicators for HFA since the some factors provided by nfeloapp like “home bye” and “away bye” doesn’t happen every week and elevation and crowd 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”.