Toy Football

JP & Adi Wyner

2024-10-18

The Problem

We are interested in the probability of a team winning a football game given a particular game state. This includes the current score, the number of possessions each team has remaining, and the probability of scoring a touchdown on a given possession.

We make a couple simplifying assumptions:

- Each team has an equal number of possesssions.
- The probability of scoring a touchdown on a particular possession is constant and independent of the current game state.
- The teams are evenly matched, so the teams' probabilities of scoring a touchdown on a particular possession are equal.
- Should the teams tie, the winner is determined by a coin flip.

The Model

Parametrization

We parametrize the model as follows:

- N: a constant representing the total number of possessions each team has.
- n_{rem}: the number of possessions remaining for each team.
- p_{td}: the probability of scoring on a given possession. For simplicity, we will assume that this is 0.5.
- A_{final}, B_{final}: the final score of team A, B.
- $\text{score}_A, \text{score}_B$: the current score of team A, B.
- A_{rem}, B_{rem}: the number of points team A and B score over the rest of the game.

It is clear that the distribution of the final score of each team is binomially distributed with parameters N, p_{td}. Furthermore, the distribution of the number of points team A and B score over the rest of the game is also binomially distributed with parameters n_{rem}, p_{td}. Note that at any point in the game, $A_{final} = \text{score}_A + A_{rem}$, and similarly for B.

Probability of Winning

At each stage in the game, we are interested in

$$\begin{aligned} P(A \text{ wins}) &= P(A_{final} > B_{final}) + \frac{1}{2}P(A_{final} = B_{final}) \\ &= P(\text{score}_A + A_{rem} > \text{score}_B + B_{rem}) + \frac{1}{2}P(\text{score}_A + A_{rem} = \text{score}_B + B_{rem}) \\ &= P(A_{rem} - B_{rem} > \text{score}_B - \text{score}_A) + \frac{1}{2}P(A_{rem} - B_{rem} = \text{score}_B - \text{score}_A) \end{aligned}$$

Consider $Z_{rem} = A_{rem} - B_{rem}$ as a random variable representing the difference in the number of points scored by each team over the rest of the game, and let $z = \text{score}_B - \text{score}_A$. We can then express $P(A \text{ wins}) = P(Z_{rem} > z) + \frac{1}{2}P(Z_{rem} = z)$.

Distribution of Remaining Score Differential Z_{rem}

PMF of Z_{rem} We characterize the PMF of Z_{rem} as a convolution of A_{rem} and B_{rem} Binomial(n_{rem}, p_{td}):

$$\begin{aligned} P(Z_{rem} = z) = P(A_{rem} - B_{rem} = z) &= \sum_{k=0}^{n_{rem}} P(A_{rem} = k)P(B_{rem} = k - z) \\ &= \sum_{k=0}^{n_{rem}} \binom{n_{rem}}{k} p^k (1-p)^{n_{rem}-k} \binom{n_{rem}}{k-z} p^{k-z} (1-p)^{n_{rem}-k+z} \\ &= \sum_{k=0}^{n_{rem}} \binom{n_{rem}}{k} \binom{n_{rem}}{k-z} p^{2k-z} (1-p)^{2n_{rem}-2k+z} \end{aligned}$$

When $p = 1/2$, we can simplify this expression further:

$$P(Z_{rem} = z) = (\frac{1}{2^{2n_{rem}}}) \sum_{k=0}^{n_{rem}} \binom{n_{rem}}{k} \binom{n_{rem}}{k-z}$$

. Then from the Chu-Vandermonde identity, we have that

$$f_{Z_{rem}}(z) = P(Z_{rem} = z) = \frac{\binom{2n_{rem}}{n_{rem}+z}}{2^{2n_{rem}}}$$

.

Validity of PMF We verify this is a valid PMF by verifying non-negativity and that it sums to 1.

- Non-negativity: The binomial coefficients are non-negative, and the denominator is positive, so the PMF is non-negative across all z.
- Normalization: Consider the sum of this PMF across all z:

$$\begin{aligned} \sum_{z=-n_{rem}}^{n_{rem}} \frac{\binom{2n_{rem}}{n_{rem}+z}}{2^{2n_{rem}}} &= \frac{1}{2^{2n_{rem}}} \sum_{z=-n_{rem}}^{n_{rem}} \binom{2n_{rem}}{n_{rem}+z} \\ &= \frac{1}{2^{2n_{rem}}} \sum_{k=0}^{2n_{rem}} \binom{2n_{rem}}{k} \\ &= \frac{1}{2^{2n_{rem}}} \cdot 2^{2n_{rem}} \\ &= 1 \end{aligned}$$

So this is a valid PMF for Z_{rem}.

CDF of Z_{rem} We can get the CDF of Z_{rem} by summing the PMF.

$$
\begin{aligned}
F_{Z_{rem}}(z) = P(Z_{rem} \leq z) &= \sum_{k=-n_{rem}}^{z} P(Z_{rem} = k) \\
&= \sum_{k=-n_{rem}}^{z} \frac{\binom{2n_{rem}}{n_{rem}+k}}{2^{2n_{rem}}} \\
&= \frac{1}{2^{2n_{rem}}} \sum_{k=-n_{rem}}^{z} \binom{2n_{rem}}{n_{rem}+k}
\end{aligned}
$$

Back to Win Probability

Recall that

$$
\begin{aligned}
P(A \text{ wins}) &= P(A_{rem} - B_{rem} > \text{score}_B - \text{score}_A) + \frac{1}{2} P(A_{rem} - B_{rem} = \text{score}_B - \text{score}_A) \\
&= P(Z_{rem} > z) + \frac{1}{2} P(Z_{rem} = z)
\end{aligned}
$$

We can then express the probability that A wins in terms of the CDF and PDF:

$$
\begin{aligned}
P(A \text{ wins}) &= 1 - P(Z_{rem} \leq z) + \frac{1}{2} P(Z_{rem} = z) \\
&= 1 - F_{Z_{rem}}(z) + \frac{1}{2} f_{Z_{rem}}(z) \\
&= 1 - \frac{1}{2^{2n_{rem}}} \left[\sum_{k=-n_{rem}}^{z} \binom{2n_{rem}}{n_{rem}+k} \right] + \frac{1}{2} \cdot \frac{\binom{2n_{rem}}{n_{rem}+z}}{2^{2n_{rem}}}
\end{aligned}
$$

And $P(B \text{ wins})$ is $1 - P(A \text{ wins})$, so

$$
\begin{aligned}
P(B \text{ wins}) &= P(Z_{rem} \leq z) - \frac{1}{2} P(Z_{rem} = z) \\
&= F_{Z_{rem}}(z) - \frac{1}{2} f_{Z_{rem}}(z) \\
&= \frac{1}{2^{2n_{rem}}} \left[\sum_{k=-n_{rem}}^{z} \binom{2n_{rem}}{n_{rem}+k} \right] - \frac{1}{2} \cdot \frac{\binom{2n_{rem}}{n_{rem}+z}}{2^{2n_{rem}}}
\end{aligned}
$$

Functions

```
# mass function
pmf = function(n_rem, z) {
  # pmf formula
  return(choose(2*n_rem, n_rem + z)/(2^(2*n_rem)))
}
```

```
# distribution function
cdf = function(n_rem, z) {
  # computing control: when z < -n_rem, sum = 0
  if(z < -n_rem) {
    return(0)
  }
  # otherwise proceed
  else {
    # cdf formula
    return(sum(sapply(-n_rem:z, function(k) choose(2*n_rem, n_rem + k)))/(2^(2*n_rem)))
  }
}

# win probability
win_prob = function(n_rem, score_a, score_b) {
  # calculate z
  z = score_b - score_a
  # calculate win probability for team a
  wp_a = 1 - cdf(n_rem, z) + 0.5*pmf(n_rem, z)
  # return
  return(wp_a)
}

# compute win probability for all possible states
win_prob_all = function(N) {
  # initialize win probability matrix
  wp_matrix = matrix(nrow = N + 1, ncol = 2 * N + 1)
  rownames(wp_matrix) = paste0('n_rem=', 0:N)
  colnames(wp_matrix) = paste0('z=', -N:N)
  # loop through number of possessions remaining
  for (n_rem in 0:N) {
    # loop through all possible score differentials
    for (z in (n_rem - N):(N - n_rem)) {
      # calculate and input win probability
      wp_matrix[n_rem + 1, z + (N + 1)] = win_prob(n_rem, N, z + N)
    }
  }
  # return
  return(wp_matrix)
}

# simulate a single game
sim_game = function(N, p_td) {
  # simulate drives (incl. game start)
  drives_a = c(0, rbinom(N, 1, p_td))
  drives_b = c(0, rbinom(N, 1, p_td))
  # live scores
  scores_a = cumsum(drives_a)
  scores_b = cumsum(drives_b)
  # win probabilities, time 0
  win_probs_a = sapply(0:N, function(i) win_prob(n_rem = N-i, score_a = scores_a[i + 1], score_b = scor
  win_probs_b = 1 - win_probs_a
  # collect game data
```

```
  game_data = data.frame(drives_a, drives_b, scores_a, scores_b, win_probs_a, win_probs_b)
  rownames(game_data) = paste0('drive ', 0:N)
    # assign winner
  if (scores_a[N + 1] > scores_b[N + 1]) {
    winner = 'a'
  } else if (scores_a[N + 1] < scores_b[N + 1]) {
    winner = 'b'
  } else {
    # if a tie, flip a coin
    winner = sample(c('a', 'b'), 1)
  }
  # return data
  return(list(game_data, winner))
}

# get max wp of losing team
max_wp_loser = function(game) {
  # get game data, winner
  game_data = game[[1]]
  winner = game[[2]]
  # extract win probabilities
  win_probs_a = game_data$win_probs_a
  win_probs_b = game_data$win_probs_b
  # get max win probability of losing team, time it occurs
  if (winner == 'a') {
    max_wp = max(win_probs_b)
    t = which.max(win_probs_b) - 1
  } else {
    max_wp = max(win_probs_a)
    t = which.max(win_probs_a) - 1
  }
  # return max wp
  return(list(max_wp, t))
}
```

Simulation

Win Probabilty for All Game States

```
# set parameters
N = 5
p_td = 0.5
# get win probability matrix
wp_matrix = win_prob_all(N)
# print, rounded to 2 dp
print(round(wp_matrix, 2))
```

```
##         z=-5 z=-4 z=-3 z=-2 z=-1 z=0  z=1  z=2 z=3 z=4 z=5
## n_rem=0    1    1    1 1.00 1.00 0.5 0.00 0.00   0   0   0
## n_rem=1   NA    1    1 1.00 0.88 0.5 0.12 0.00   0   0  NA
## n_rem=2   NA   NA    1 0.97 0.81 0.5 0.19 0.03   0  NA  NA
## n_rem=3   NA   NA   NA 0.94 0.77 0.5 0.23 0.06  NA  NA  NA
## n_rem=4   NA   NA   NA   NA 0.75 0.5 0.25   NA  NA  NA  NA
```

```
## n_rem=5   NA   NA   NA   NA   NA 0.5   NA   NA  NA  NA  NA
```

Single-Game Simulation

```
# set seed
set.seed(2024-10)

# set parameters
N = 10
p_td = 0.5

# sim game
game = sim_game(N, p_td)
# get game data, winner
game_data = game[[1]]
winner = game[[2]]
# print game data, rounded to 2 dp
print(round(game_data, 2))
```

```
##          drives_a drives_b scores_a scores_b win_probs_a win_probs_b
## drive 0         0        0        0        0        0.50        0.50
## drive 1         0        1        0        1        0.32        0.68
## drive 2         0        0        0        1        0.31        0.69
## drive 3         1        1        1        2        0.30        0.70
## drive 4         0        1        1        3        0.13        0.87
## drive 5         1        1        2        4        0.11        0.89
## drive 6         0        0        2        4        0.09        0.91
## drive 7         1        1        3        5        0.06        0.94
## drive 8         1        0        4        5        0.19        0.81
## drive 9         0        1        4        6        0.00        1.00
## drive 10        0        0        4        6        0.00        1.00
```

Max Win Probability of Losing Team

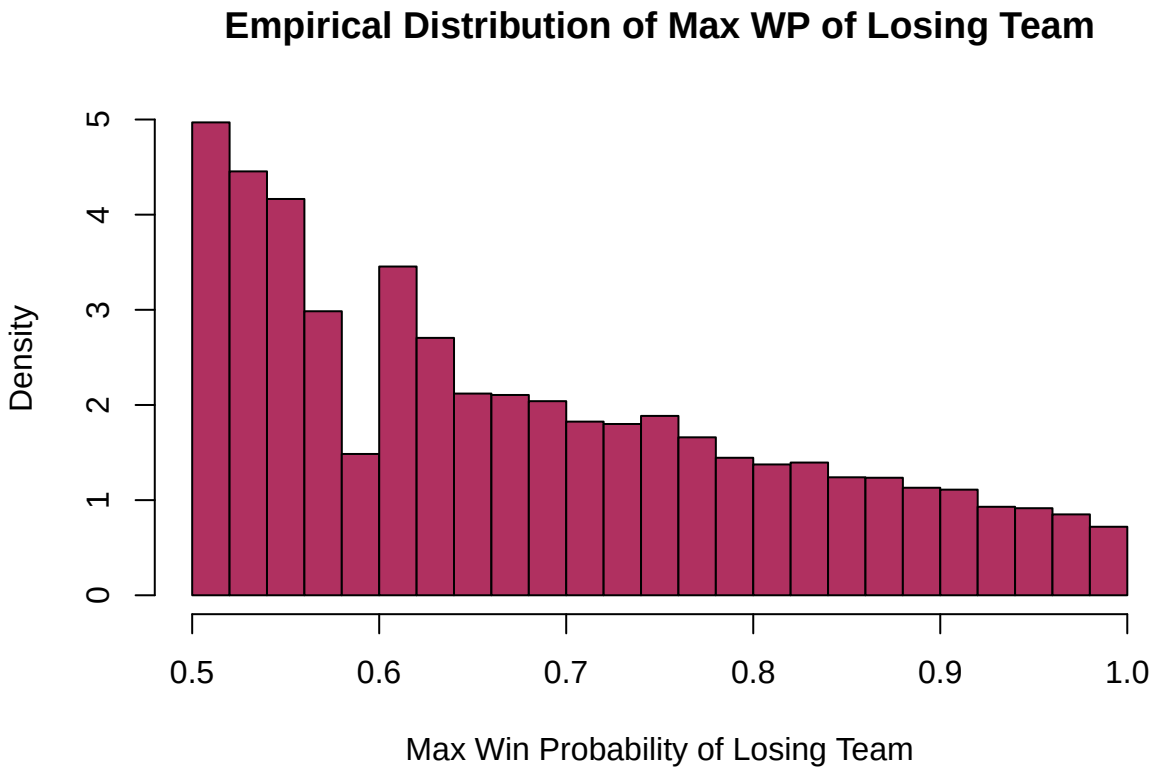
```
# set seed
set.seed(2024-10)

# set parameters
n_sim = 10000
N = 500
p_td = 0.5

# initialize max wp vector
max_wp_losers = numeric()
t_losers = numeric()
# each time
for (i in 1:n_sim) {
  # simulate game
  game = sim_game(N, p_td)
  # get max win probability of losing team, time it occurs
  wp_data = max_wp_loser(game)
  # extract data
  max_wp = wp_data[[1]]
  t = wp_data[[2]]
```

```r
  # store
  max_wp_losers = c(max_wp_losers, max_wp)
  t_losers = c(t_losers, t)
}

# plot empirical distribution of max wp of losing team
hist(max_wp_losers, breaks = 20, freq = F,
     col = 'maroon', border = 'black',
     main = 'Empirical Distribution of Max WP of Losing Team', xlab = 'Max Win Probability of Losing Tea
```

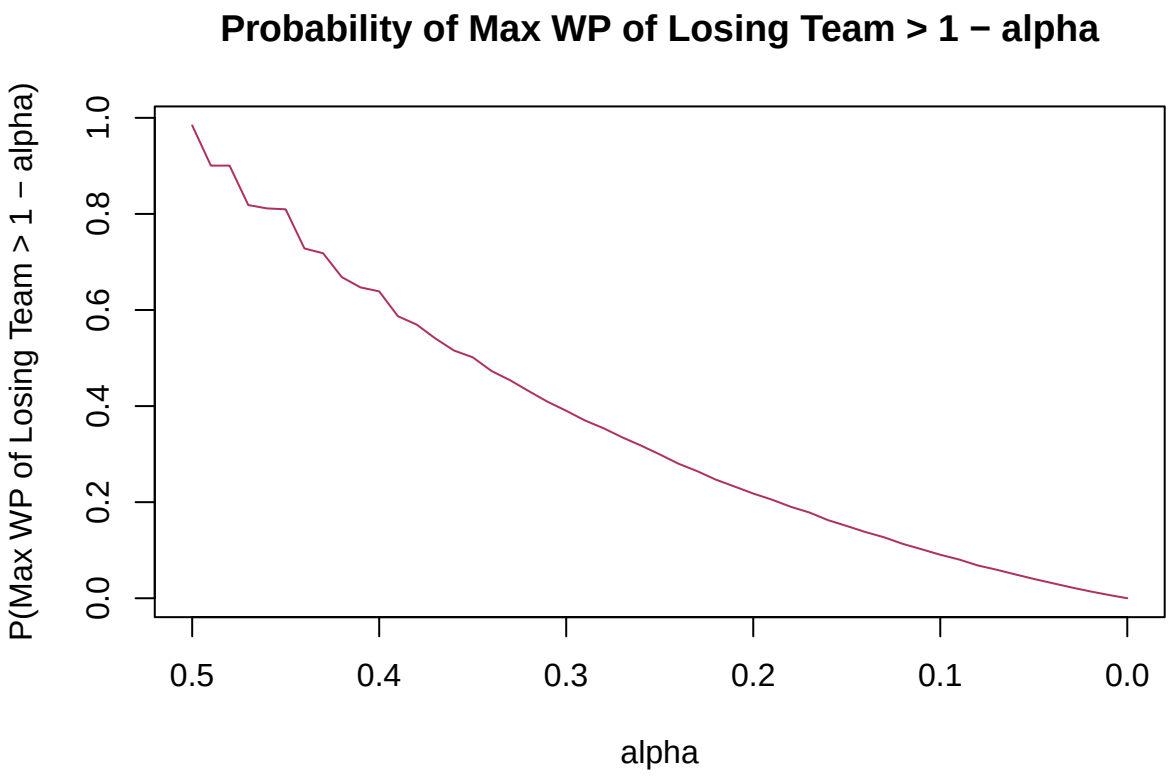


Result: The empirical proportion of games the losing team had a max win probability of 0.5 is 0. The empirical proportion of games the losing team had a max win probability of at least 0.9 is 0.0905.

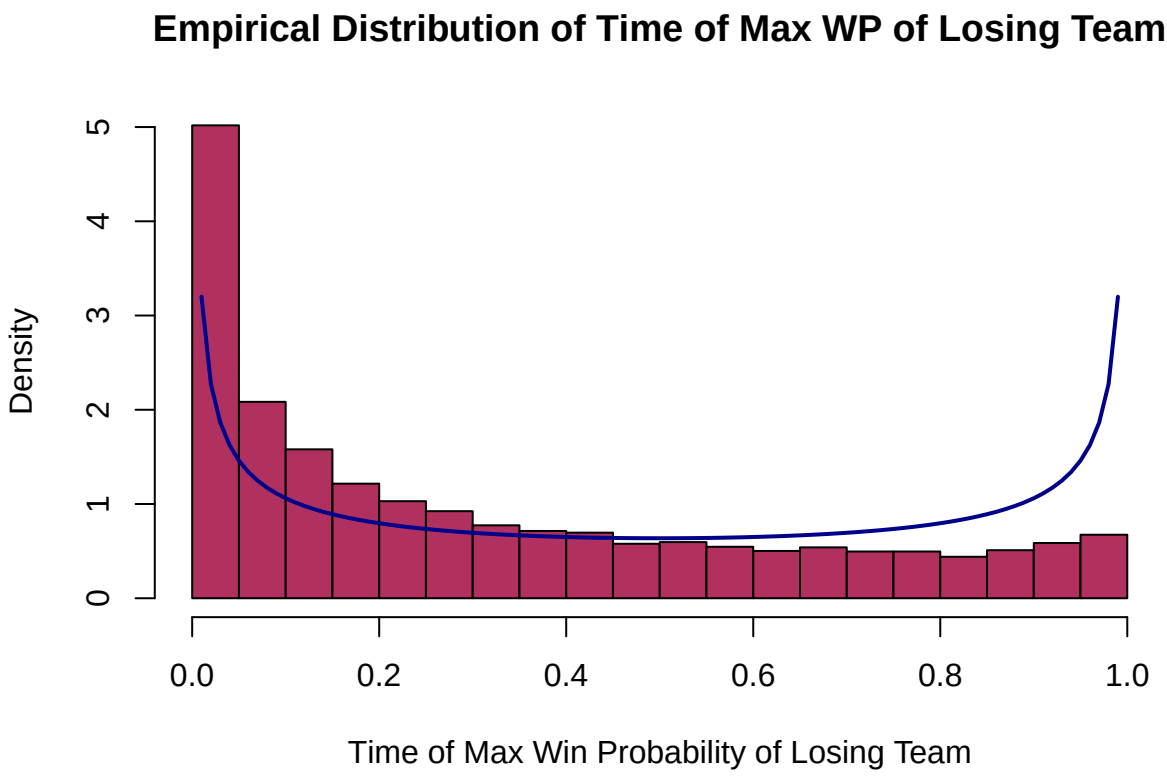
```r
# plot of (Z > 1 - alpha) where Z is win prob of losing team

# set alpha values
alpha_values = seq(0, 0.5, by = 0.01)
# calculate probabilities
prob_values = sapply(alpha_values, function(alpha) mean(max_wp_losers > (1 - alpha)))

# plot
plot(alpha_values, prob_values, type = 'l', col = 'maroon',
  main = 'Probability of Max WP of Losing Team > 1 - alpha',
  xlab = 'alpha', ylab = 'P(Max WP of Losing Team > 1 - alpha)',
  xlim = rev(range(alpha_values)))
```




```r
# plot empirical distribution of t of max wp of losing team (normalized)
hist(t_losers / N, breaks = 20, freq = F,
     col = 'maroon', border = 'black',
     main = 'Empirical Distribution of Time of Max WP of Losing Team', xlab = 'Time of Max Win Probabil
# overlay theoretical arcsin distribution
curve(1 / (pi * sqrt(x * (1 - x))), add = T, col = 'darkblue', lwd = 2)
```



Result: The empirical proportion of games the losing team achieved their max win probability (0.5) at the start of the game is 0.04.

Note: As currently coded, $t_{\max wp}$ measures the *first* time the maximum win probability is achieved. This can be adjusted to measure the *last* time the maximum win probability is achieved if we are interested in investigating comebacks.