Why aren't there any 6-win or 6-loss teams in our league

Elliot Riesman-Tremonte

11/22/2020

Introduction

Through 10 weeks of Fantasy Football play, our league's playoff picture is still cloudy. As usual, many teams are still competitive for a playoff berth while some players have failed to live up to expectations. What is unusual, however, is the absence of a 4-6 team or a 6-4 team. This phenomenon seems strange, and in this report I will examine the likelihood of this occurrence from a variety of statistical perspectives.

Data

At first, I will examine the lack of 4-6 or 6-4 teams from a purely scheduling perspective. In other words, assuming that our teams and managerial decisions are fixed (but not the East/West divisions and schedules), what is the likelihood of a random schedule producing this phenomenon? I will obtain this data by directly copying from our league's scoreboard. Unfortunately, ESPN does not provide a CSV or TXT file to directly load into RStudio. Then using the data, I will try to generalize to hypothetical leagues whose weekly scoring distributions mimic our own league. This data will be obtained from specific distribution functions in R packages (rnorm, etc.).

Research Question

How ridiculous is the current absence of 4-6 or 6-4 teams?

Our League

Lots of Possible Schedules

There are several criteria that make a schedule in 10-team ESPN fantasy leagues valid. No team can play the same team twice in consecutive weeks. Each team must play each division rival twice and each interdivisional opponent once. Combinatorially, there is a massive number of possible league schedules. Take Week 1 for example. Suppose you designate each team A, B, C, . . . J:

- A can play any team X in the 9-element set $[B, J]$
- Some team Y (that is not X or A) can play any of the remaining 7 teams
- Repeat this process to find there are $9 * 7 * 5 * 3 * 1 = 945$ possible schedules in any given week.

This result can also be obtained this way:

- There are 10! ways to order a list of 10 unique teams
- Remember that a particular schedule AB, CD, EF, GH, IJ is equivalent to a schedule BA, CD, EF, GH, IJ
  - So, there are 2^5 ways to express any particular order of 5 matchups
- Remember that ordering of the matchups also does not matter. There are 5! ways of ordering a set of matchups, all of which should only be counted once.
- $\frac{10!}{2^5 5!} = 945$

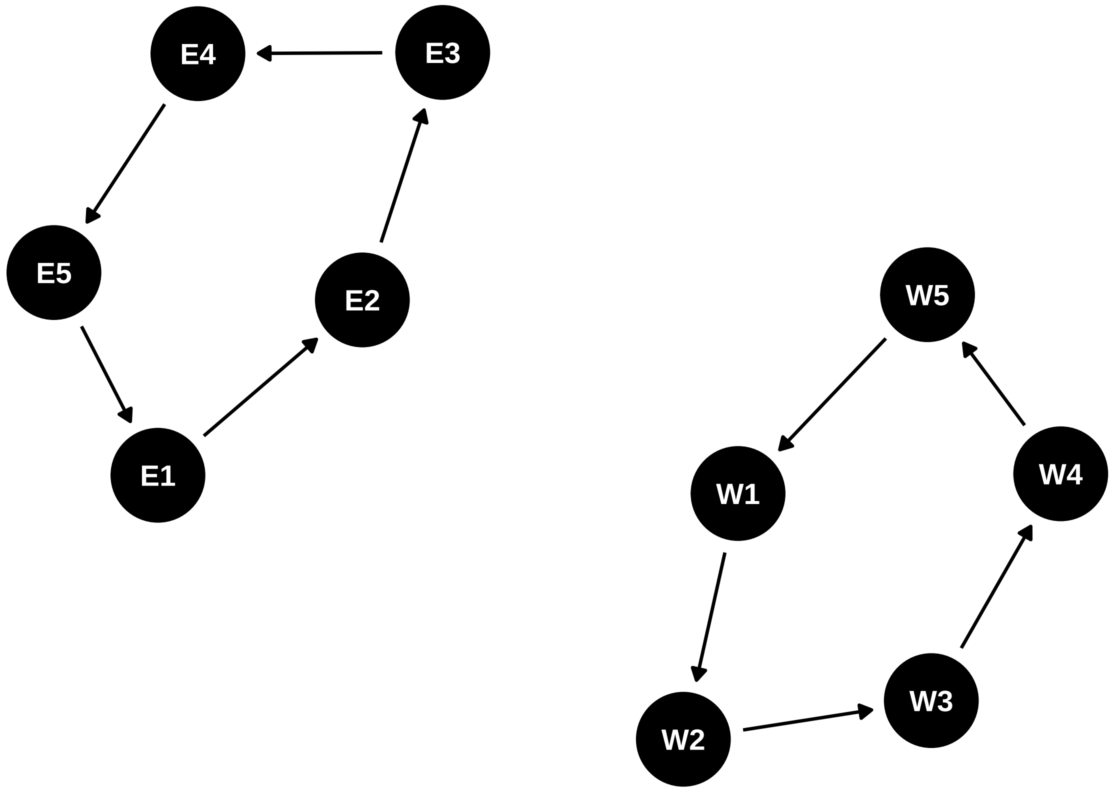
Yes, in any given week, divisional and counting constraints will significantly reduce this number, but there are still a lot of possible schedules.

We only really use 1 schedule

Thankfully, ESPN does not schedule by creating all possible valid schedules and then selecting one at random. Instead, ESPN 2-division, 10-team schedules are super easy to understand.

- General Setup
  - Each team will play all 4 divisional opponents, as well as 1 interdivisional opponent, in the first 5 weeks.
  - Each team will play all 4 divisional opponents, as well as 1 interdivisional opponent, int the last 5 weeks.
  - In Weeks 6-8, all teams play interdivisional opponents.
- Visualizing the Schedule
  - Look at the diagram presented below
  - The East and West teams are designated E1, ... E5 and W1, ... W5
  - In weeks 1 through 5:
    - E1 plays W1 in Week 1, E2 plays W2 in Week 2, and so forth.
    - When an East team, say EX, is playing its interdivisional matchup in Week X, the teams adjacent to EX on the diagram will play each other. The teams non-adjacent to EX on the diagram will play each other. The same goes for the West division.
  - In Weeks 6 through 8, W1 plays E3, E4 and E5, in that order. W2 plays E4, E5, E1. W3 plays E5, E1, E2, etc.
  - Before Week 9, E1 becomes E2, E2 becomes E3, E3 become E4, E4 becomes E5 and E5 becomes E1.
  - After such rotation, Weeks 9 through 13 are played the exact same way as weeks 1 through 5. This rotation enables the schedule to work for all criteria.
  - To help understand this process, E1 and W1 are Alex and Aaron, respectively, in our league.

This leaves us with only 10! = 3268800 total schedules, because each team in our league can occupy any one of these schedules.



Creating a Schedule matrix

In the matrix below, rows represent Weeks. In any particular row, the 1st and 2nd element play each other, etc.

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    6    2    5    3    4    8    9    7    10
##  [2,]    2    7    1    3    4    5    6    8    9    10
##  [3,]    3    8    2    4    7    9    1    5    6    10
##  [4,]    4    9    3    5    8   10    1    2    6     7
##  [5,]    5   10    1    4    6    9    2    3    7     8
##  [6,]    6    2    7    3    8    4    9    5   10     1
##  [7,]    6    3    7    4    8    5    9    1   10     2
##  [8,]    6    4    7    5    8    1    9    2   10     3
##  [9,]    6    5    2    3    1    4    8    9    7    10
## [10,]    1    7    3    4    2    5    6   10    7     8
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Creating a function

The function ‘scoring’, seen below takes a schedule matrix and a vector of ‘teams’ that match the numbers 1 through 10 to specific teams

```r
#Here is the function scoring

scoring <- function (mat, teams) {
  win_vec <- c(rep(0, 10))
  for (i in 1:10) {
    for (j in seq(from = 1, to = 9, by = 2))
      if (teams[i, mat[i, j]] >= teams[i, mat[i, j + 1]]) {
        win_vec[mat[i, j]] <- win_vec[mat[i, j]] + 1
      } else {
        win_vec[mat[i, j + 1]] <- win_vec[mat[i, j + 1]] + 1
      }
  }
  win_vec
}
#Current schedule loads the current schedule we have used this season

current_schedule <- cbind(ALX, EJO, TOM, EBE, EMM, AAF, CAL, SEB, ELL, JUS)

current_schedule
```

```
##          ALX    EJO    TOM    EBE    EMM    AAF    CAL    SEB    ELL    JUS
##  [1,]  95.10 114.70 139.92  87.84 109.98  81.78 139.30  83.90 106.56 106.58
##  [2,]  98.58 115.66 126.80 112.28 115.30 114.42 133.74 136.24  77.48 120.50
##  [3,] 101.14  84.08 114.76 116.40 139.34 151.80 103.30 126.20 103.82 112.72
##  [4,] 118.30 104.62 122.50  71.62 118.12 108.80 104.12  94.10 115.90 114.98
##  [5,] 113.94 104.10 135.44  90.50 118.84 103.38  83.20 110.82 107.54  85.86
##  [6,]  98.82 129.64 158.10  87.70  83.98  82.64  71.32  74.32  83.14 115.48
##  [7,] 122.02  85.72  77.76  85.66 106.78 110.22 100.80 127.86 118.10 100.62
##  [8,] 111.12  69.72 126.14  87.24 100.46  64.44  37.30 119.12 132.66  89.48
##  [9,] 111.46  98.50 110.94  81.28 116.70  94.80  78.82  98.44  85.96  57.04
## [10,]  84.04  87.26 103.72  45.10  89.26  52.62  87.00 111.28  74.98 135.00
```

```r
names <- c("Alex", "Eli J", "Tom", "Eli B", "Emmett", "Aaron", "Mike", "Seb", "Elliot", "Justin")

cbind(names, scoring(schedule, current_schedule))
```

```
##       names
##  [1,] "Alex"   "5"
##  [2,] "Eli J"  "2"
##  [3,] "Tom"    "8"
##  [4,] "Eli B"  "3"
##  [5,] "Emmett" "7"
##  [6,] "Aaron"  "3"
##  [7,] "Mike"   "5"
##  [8,] "Seb"    "7"
##  [9,] "Elliot" "5"
## [10,] "Justin" "5"
```

As you can see, running the scoring function on our existing schedule properly apportions the amount of wins each team should have.

Generating Random Orders of Teams

In this section, 100,000 random team schedules are selected from the over 3 million permutations of 10 elements in a list. Then the 'scoring' function is called to see what the league wins outcome will be. The histogram provides a visualization of how many teams are 4-6 and 6-4 in different situations. The final number provides the percentage of leagues that have none of these teams.

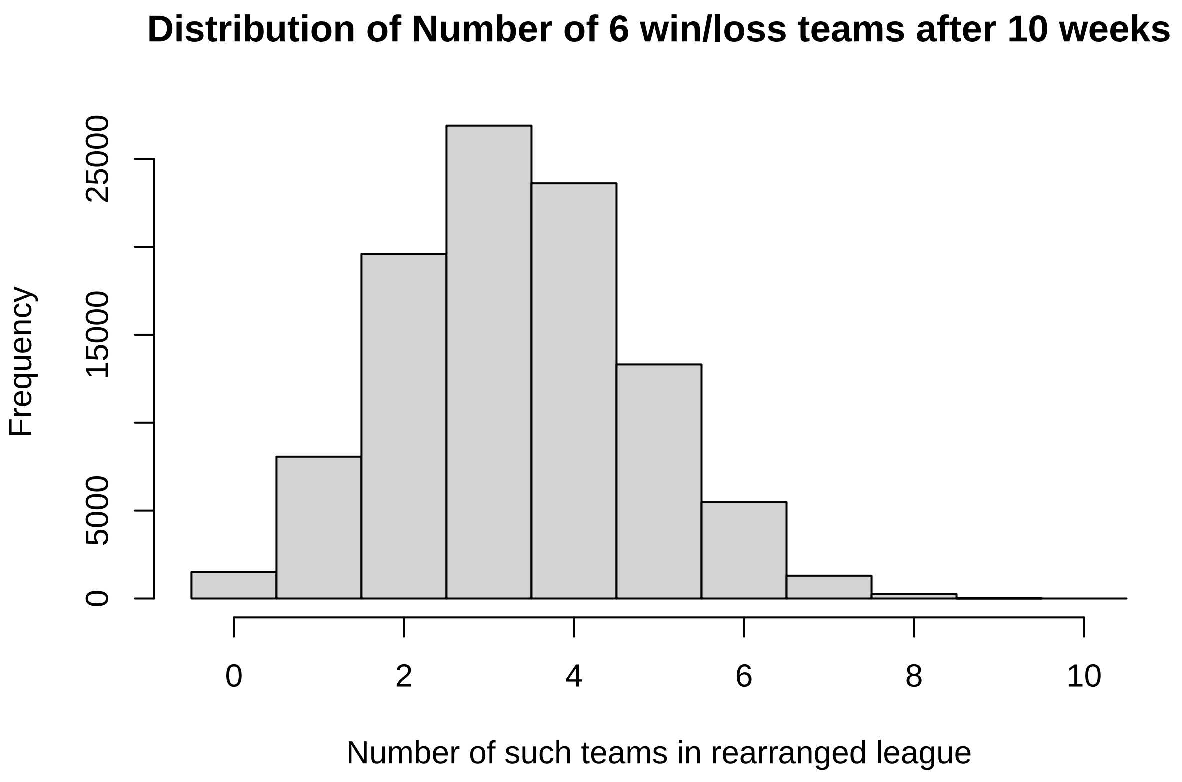
```r
k1 <- permn(1:10)
k2 <- sample(k1, 100000)
data_final <- rep(0, 100000)
for (l in 1:100000) {
  team_order <- as.matrix(current_schedule[, k2[[l]][1]])
  for (m in 2:10) {
    team_order <- cbind(team_order, current_schedule[, k2[[l]][m]])
  }
  wins <- scoring(schedule, team_order)
  data_final[l] <- length(wins[wins == 4 | wins == 6])
}
#Mean number of teams with either 6 wins or 6 losses
mean(data_final)
```

```
## [1] 3.32855
```

```r
#Number of 0-situations
length(data_final[data_final == 0])
```

```
## [1] 1500
```

```r
hist(data_final, main = "Distribution of Number of 6 win/loss teams after 10 weeks", breaks = c(-0.5, 0
```



```
#Portion of Simulations that satisfy the criteria
length(data_final[data_final == 0]) / length(data_final)
```

```
## [1] 0.015

## [1] 4.66257

## [1] 0
```

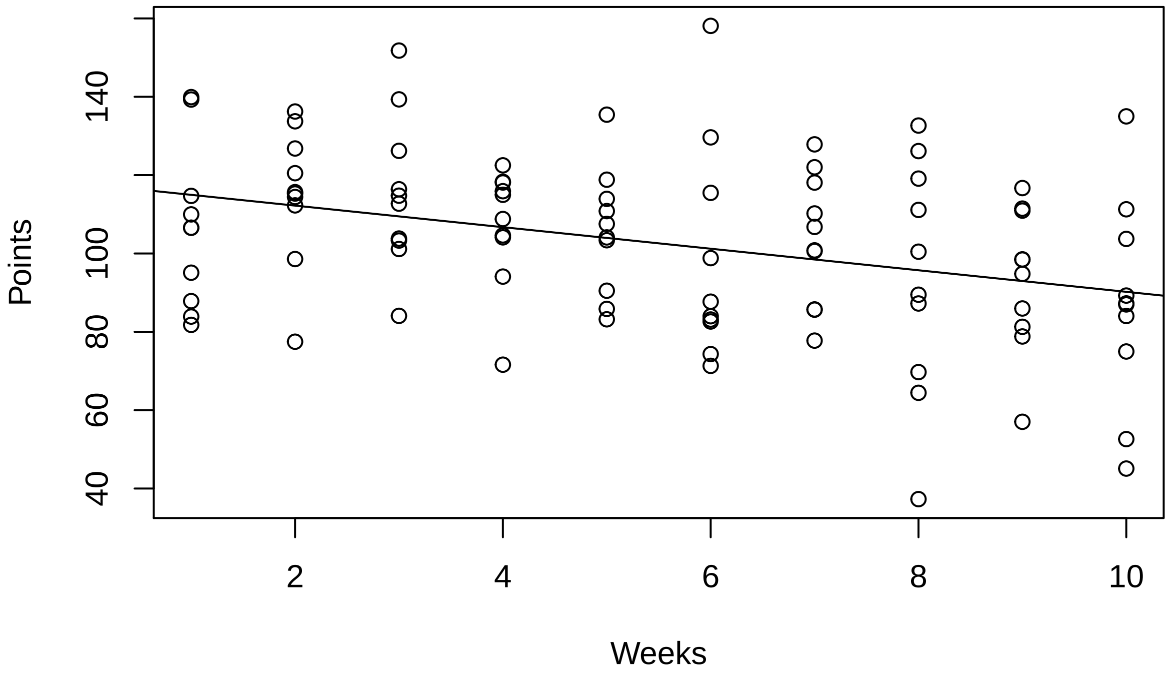
Between 1 and 2 percent of league scenarios resulted in 0 6W/6L teams.

Randomizing leagues similar to ours

Looking at weekly team/league scores

```
points <- as.matrix(as.vector(current_schedule))
weeks <- c(rep(1:10, 10))
players <- c(rep("Alex", 10), rep("Eli J", 10), rep("Tom", 10),
             rep("Eli B", 10), rep("Emmett", 10), rep("Aaron", 10),
             rep("Mike", 10), rep("Seb", 10), rep("Elliot", 10),
             rep("Justin", 10))
df1 <- data.frame(points, weeks, players)


wksmodel <- summary(lm(data = df1, points ~ weeks))
plot(x = df1$weeks, y = df1$points, xlab = "Weeks", ylab = "Points") +
  abline(a = wksmodel$coefficients[1], b = wksmodel$coefficients[2])
```



```
## integer(0)
```

P-values are relatively low here. Now I am adding, as a regressor, the mean of a team's output.

Simulating team scores based on a linear model with 2 regressors

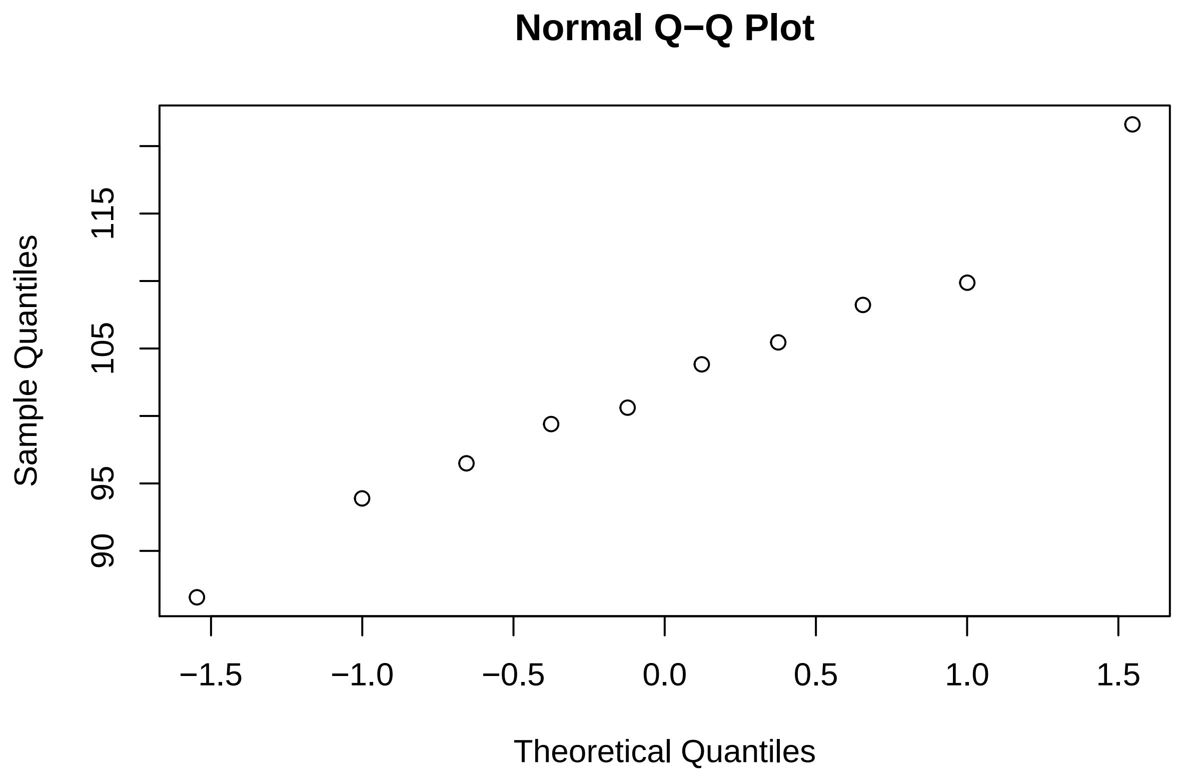
```r
player_means <- function(player) {
  mean(df1$points[df1$players == player])
}
df1$means <- sapply(df1$players, player_means)

#Creating a linear model using the R lm function

model_points <- summary(lm(data = df1, points ~ means + weeks))

fit_value <- function(week, team) {
  model_points$coefficients[1] + team*model_points$coefficients[2] + week*model_points$coefficients[3]
}
sterror <- model_points$sigma
```

This model ‘model_points’ will serve as an estimator of a random team’s points over the first 10 weeks of a season, given the Week # and a team’s average point total. Of course, these point totals vary and we must check to see if team’s mean points per week follow a normal distribution. A Normal QQ-Plot can help to support or contradict a normality assumption.



Even though 10 is an extremely small dataset, for the purposes of this exercise, mean points per week by team can follow a normal distribution, especially because the Normal QQ plot generally follows a straight line.

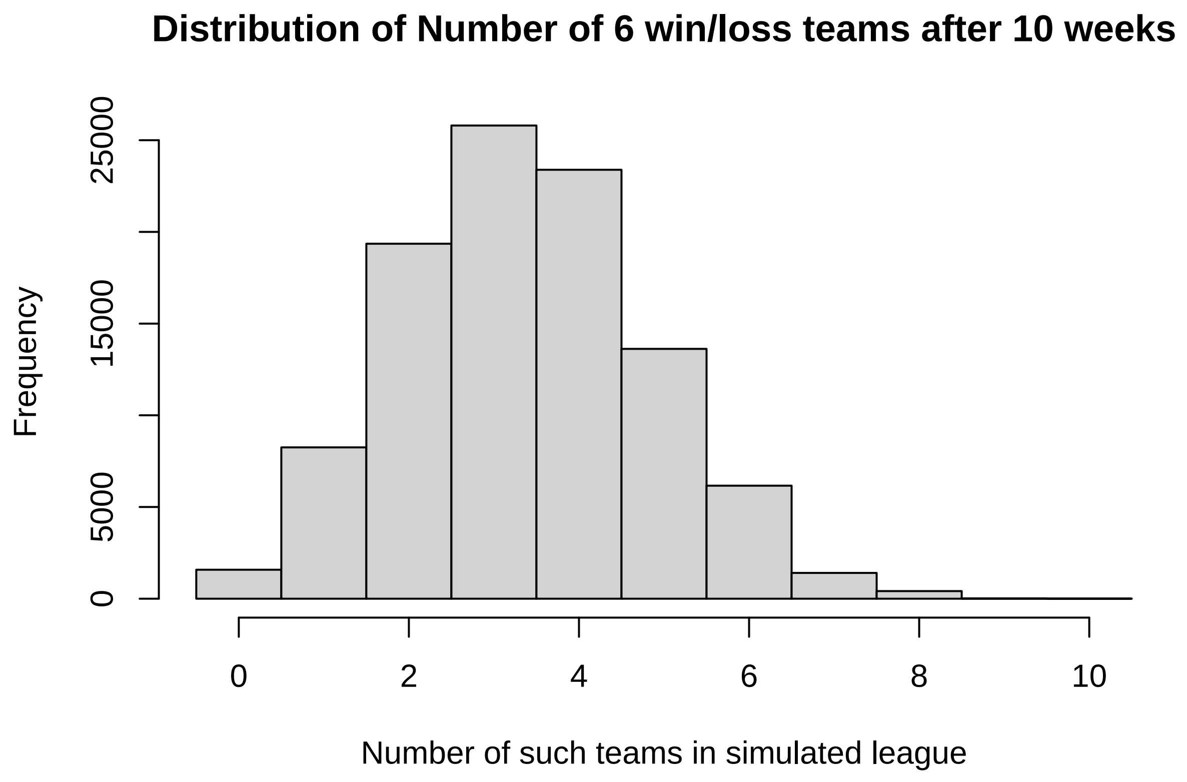
Here, we compute a random league matrix for 10 teams in 10 weeks. Each column has a specified team mean.

```
sd_means <- sd(team_means)

ran_matrix <- matrix(c(rep(0, 100)), ncol = 10)
ran_data <- c(rep(0, 100000))

#Running 100,000 simulations of random leagues
for (c in 1:100000) {
    ran_means <- rnorm(10, mean = mean(team_means), sd = sd_means)
    for (a in 1:10) {
      for (b in 1:10)
        ran_matrix[b, a] <- fit_value(b, ran_means[a]) + rnorm(1, mean = 0, sd = sterror)
    }
    ran_wins <- scoring(schedule, ran_matrix)
    ran_data[c] <- length(ran_wins[ran_wins == 4 | ran_wins == 6])
}

hist(ran_data, main = "Distribution of Number of 6 win/loss teams after 10 weeks", breaks = c(-0.5, 0.5
```



```
#Mean number of 6W/6L teams
mean(ran_data)
```

```
## [1] 3.36321
```

```
#Portion of 0-situations among randomly determined leagues
length(ran_data[ran_data == 0]) / length(ran_data)
```

```
## [1] 0.0158
```

Conclusion

It’s pretty weird that there aren’t any 6-win or 6-loss teams. There appears to be a roughly 1.5% chance of this happening in a given league given ESPN standard 2-division, 10-team scoring. In our league, assuming managerial and draft decisions constant, about 1.5% of schedules produce no 6-win or 6-loss teams. In other words, there doesn’t seem to be something about our league’s scoring to date that increases the chances of this phenomenon happening. It’s weird, but not that weird.