Wrangling Project

Thomas Zwiller

2025-05-12

I came to Mendoza because the MSBA program seemed like a perfect fit for me. I was a business major at Holy Cross College, but I also completed a computer science minor so I had some level of coding experience. I have been fascinated by machine learning and building models since the 2016 election when I first learned about Nate Silver.

One of the things that helped grow that interest was modeling sports. I created an NFL model as a freshman at Holy Cross, and then developed a college football model as a junior. One of my goals for the Mendoza program was to learn how to refine my CFB model and make maintaining it more efficient (I update it manually every week in an excel spread sheet).

This is my first step toward completing that goal. But first, a crash course about how my model works.

My Model uses the following equation to make its team rating:

$$\text{Team A Offense Rating} = (PassingCompletions \times 3.7) + (PassingAttempts \times -2.2) + \left(\frac{PassingYards}{4}\right) + (Passing\,Touchdo \tag{1}$$

This equation is used twice, once for the offense rating and once for the defensive rating:

$$\text{Team A Net Rating} = \text{Team A Offense Rating} - \text{Team A Defense Rating} \tag{2}$$

Ideally, you would want your team to be rated as highly as possible on on offense while having as low a defensive rating as possible (negative even!).

A positive net rating represents a better than average team, while a negative rating is below average. The higher the positive number, the better the team, the lower the negative number, the worse the team.

To determine how likely a team is to win a game, the two ratings are compared in the following equation.

$$\text{Team A \% Of Winning} = \frac{1}{1 + 10^{(\text{Team B Rating} - \text{Team A Rating})/400}} \tag{3}$$

My end goal was not to aggregate all my data and then calculate each teams chance of winning a given game. Instead I just wanted to calculate each teams rating from the last ten seasons and then produce a top-25 list based on net rating, then graph those 25 teams.

My first step was to load in the data that I found from Kragle.

```
cfb_2014 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2015 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2016 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2017 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2018 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
```

```
cfb_2019 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2020 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2021 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2022 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
cfb_2023 <- read.csv("/Users/TomTheIntern/Desktop/Mendoza/Mod 1/Wrangling/Wrangling Project/CFB Data/cf
```

My second step was to start building a function:

```
team_rating_func <- function(Pass_Comp, Pass_Att, Pass_Yards, Pass_TD, Rush_Att, Rush_Yards, Rush_TD, Fu
  rating = (as.numeric(Pass_Comp) * 3.7) +
    (as.numeric(Pass_Att) * -2.2) +
    (as.numeric(Pass_Yards)/ 4) +
    (as.numeric(Pass_TD) * 11.3) +
    (as.numeric(Rush_Att) * -1.5) +
    (as.numeric(Rush_Yards) * 0.5) +
    (as.numeric(Rush_TD) * 15.9) +
    (as.numeric(Fumbles) * -14.1) +
    (as.numeric(Ints) * -14.1)

  rating = rating / as.numeric(Games)

  return(rating)
}
```

This function could be applied to offensive and defensive ratings, though the variables would need to be carefully flipped for it to work.

Now, I could actually crack open my data and see what form of wrangling I actually needed to do.

```
str(cfb_2014)
```

```
## 'data.frame':    113 obs. of  146 variables:
##  $ Team                              : chr  "Akron (MAC)" "Alabama (SEC)" "Arizona (Pac-12)" "Arizo
##  $ Games                             : int  12 14 14 13 13 13 12 13 12 13 ...
##  $ Win                               : int  5 12 10 10 7 7 4 8 5 11 ...
##  $ Loss                              : int  7 2 4 3 6 6 8 5 7 2 ...
##  $ Off.Rank                          : int  88 17 25 34 60 20 102 16 92 1 ...
##  $ Off.Plays                         : int  891 1018 1139 975 916 1024 770 939 857 1138 ...
##  $ Off.Yards                         : int  4479 6783 6491 5750 5278 6194 4305 6305 4431 7559 ...
##  $ Off.Yards.Play                    : num  5.03 6.66 5.7 5.9 5.76 6.05 5.59 6.71 5.17 6.64 ...
##  $ Off.TDs                           : int  32 67 55 54 52 60 36 55 35 81 ...
##  $ Total.TDs                         : int  33 68 61 59 56 65 40 58 37 82 ...
##  $ Off.Yards.per.Game                : num  373 484 464 442 406 ...
##  $ Def.Rank                          : int  44 12 103 81 10 83 90 64 86 50 ...
##  $ Def.Plays                         : int  859 945 1115 964 821 975 798 914 877 941 ...
##  $ Yards.Allowed                     : int  4453 4598 6314 5422 4204 5476 5170 5185 5146 4964 ...
##  $ Yards.Play.Allowed                : num  5.18 4.87 5.66 5.62 5.12 5.62 6.48 5.67 5.87 5.28 ...
##  $ Off.TDs.Allowed                   : int  34 24 47 39 30 52 50 44 40 41 ...
##  $ Total.TDs.Allowed                 : int  35 27 49 46 33 54 50 45 42 43 ...
##  $ Yards.Per.Game.Allowed            : num  371 328 451 417 323 ...
##  $ First.Down.Rank                   : int  51 26 122 58 18 85 58 98 68 52 ...
##  $ First.Down.Runs                   : int  100 81 120 100 97 146 117 119 121 85 ...
##  $ First.Down.Passes                 : int  124 141 188 146 115 108 131 131 128 128 ...
```

```
##  $ First.Down.Penalties               : int  31 12 29 15 13 25 13 37 16 43 ...
##  $ First.Downs                        : int  255 234 337 261 225 279 261 287 265 256 ...
##  $ First.Down.Def.Rank                : int  51 26 122 58 18 85 58 98 68 52 ...
##  $ Opp.First.Down.Runs                : int  100 81 120 100 97 146 117 119 121 85 ...
##  $ Opp.First.Down.Passes              : int  124 141 188 146 115 108 131 131 128 128 ...
##  $ Opp.First.Down.Penalties           : int  31 12 29 15 13 25 13 37 16 43 ...
##  $ Opp.First.Downs                    : int  255 234 337 261 225 279 261 287 265 256 ...
##  $ X4th.Down.Rank                     : int  113 5 23 92 20 58 57 95 1 6 ...
##  $ X4th.Attempts                      : int  7 10 16 10 12 18 16 3 11 25 ...
##  $ X4th.Conversions                   : int  21 13 26 23 19 36 31 7 13 34 ...
##  $ X4th.Percent                       : num  0.333 0.769 0.615 0.435 0.632 0.5 0.516 0.429 0.846 0.7
##  $ X4rd.Down.Def.Rank                 : int  29 23 23 51 1 36 117 90 51 4 ...
##  $ Opp.4th.Conversion                 : int  7 7 7 14 4 10 16 15 11 6 ...
##  $ Opp.4th.Attempt                    : int  17 18 18 28 18 23 23 25 22 24 ...
##  $ Opponent.4th.Percent               : num  0.412 0.389 0.389 0.5 0.222 0.435 0.696 0.6 0.5 0.25 ..
##  $ Kickoff.Return.Rank                : int  30 76 27 109 112 4 87 84 107 20 ...
##  $ Kickoffs.Returned                  : int  43 60 40 55 50 42 41 33 41 81 ...
##  $ Kickoff.Return.Yards               : int  828 1291 758 1280 1192 685 892 717 946 1504 ...
##  $ Kickoff.Return.Touchdowns          : int  0 1 0 1 1 0 0 0 1 1 ...
##  $ Avg.Yard.per.Kickoff.Return        : num  19.3 21.5 18.9 23.3 23.8 ...
##  $ Passing.Off.Rank                   : int  49 28 21 32 100 39 125 66 78 4 ...
##  $ Pass.Attempts                      : int  528 451 564 467 359 448 105 332 409 519 ...
##  $ Pass.Completions                   : int  284 290 320 281 199 276 50 208 232 323 ...
##  $ Interceptions.Thrown.x             : int  14 10 10 9 6 8 2 7 7 8 ...
##  $ Pass.Yards                         : int  2995 3890 3945 3556 2444 3381 747 2984 2590 4757 ...
##  $ Pass.Yards.Attempt                 : num  5.67 8.63 6.99 7.61 6.81 7.55 7.11 8.99 6.33 9.17 ...
##  $ Yards.Completion                   : num  10.6 13.4 12.3 12.7 12.3 ...
##  $ Pass.Touchdowns                    : int  14 32 29 34 21 25 3 23 19 38 ...
##  $ Pass.Yards.Per.Game                : num  250 278 282 274 188 ...
##  $ Pass.Def.Rank                      : int  53 58 118 103 37 44 83 68 82 107 ...
##  $ Opp.Completions.Allowed            : int  225 268 349 269 217 238 244 243 250 249 ...
##  $ Opp.Pass.Attempts                  : int  402 493 534 453 392 410 362 421 402 456 ...
##  $ Opp.Pass.Yds.Allowed               : int  2670 3164 3937 3376 2714 2808 2854 2991 2853 3434 ...
##  $ Opp.Pass.TDs.Allowed               : int  20 19 28 22 19 18 31 22 16 24 ...
##  $ Yards.Attempt.Allowed              : num  6.64 6.42 7.37 7.45 6.92 6.85 7.88 7.1 7.1 7.53 ...
##  $ Yards.Completion.Allowed           : num  11.9 11.8 11.3 12.6 12.5 ...
##  $ Pass.Yards.Per.Game.Allowed        : num  222 226 281 260 209 ...
##  $ Penalty.Rank                       : int  116 21 103 4 37 113 10 112 24 124 ...
##  $ Penalties                          : int  101 69 92 58 68 104 49 92 50 127 ...
##  $ Penalty.Yards                      : int  870 562 881 421 581 905 435 896 489 1149 ...
##  $ Penalty.Yards.Per.Game             : num  72.5 40.1 62.9 32.4 44.7 ...
##  $ Punt.Return.Rank                   : int  62 55 82 114 7 9 6 72 78 32 ...
##  $ Punt.Returns                       : int  17 12 32 15 16 10 5 19 14 11 ...
##  $ Net.Punt.Return.Yards              : int  123 83 268 196 49 31 15 148 114 56 ...
##  $ Punt.Return.Touchdowns             : int  0 0 0 2 0 0 0 0 0 0 ...
##  $ Avg.Yards.Per.Punt.Return          : num  7.24 6.92 8.38 13.07 3.06 ...
##  $ Punt.Return.Def.Rank               : int  62 55 82 114 7 9 6 72 78 32 ...
##  $ Opp.Punt.Returns                   : int  17 12 32 15 16 10 5 19 14 11 ...
##  $ Opp.Net.Punt.Return.Yards          : int  123 83 268 196 49 31 15 148 114 56 ...
##  $ Opp.Punt.Return.Touchdowns.Allowed : int  0 0 0 2 0 0 0 0 0 0 ...
##  $ Avg.Yards.Allowed.per.Punt.Return  : num  7.24 6.92 8.38 13.07 3.06 ...
##  $ Redzone.Off.Rank                   : int  122 46 83 12 89 77 102 31 21 50 ...
##  $ Redzone.Attempts                   : int  47 64 60 64 56 58 37 58 46 73 ...
##  $ Redzone.Rush.TD                    : int  15 27 20 17 24 24 24 25 14 40 ...
```

```
##  $ Redzone.Pass.TD                    : int  8 18 13 23 12 13 2 13 8 8 ...
##  $ Redzone.Field.Goals.Made           : int  9 10 15 18 8 10 2 13 19 14 ...
##  $ Redzone.Scores                     : int  32 55 48 58 44 47 28 51 41 62 ...
##  $ Redzone.Points                     : num  0.681 0.859 0.8 0.906 0.786 0.81 0.757 0.879 0.891 0.849
##  $ Redzone.Def.Rank                   : int  22 72 107 23 5 42 111 13 18 98 ...
##  $ Opp.Redzone.Attempts               : int  41 44 53 46 33 44 56 54 55 38 ...
##  $ Opp.Redzone.Rush.TD.Allowed        : int  9 4 13 11 10 21 17 19 22 15 ...
##  $ Opp.Redzone.Pass.Touchdowns.Allowed: int  13 13 19 15 10 8 22 11 10 11 ...
##  $ Opp.Redzone.Field.Goals.Made       : int  9 20 15 9 3 6 11 10 9 7 ...
##  $ Opp.Redzone.Scores                 : int  31 37 47 35 23 35 50 40 41 33 ...
##  $ Redzone.Points.Allowed             : num  0.756 0.841 0.887 0.761 0.697 0.795 0.893 0.741 0.745 0
##  $ Rushing.Off.Rank                   : int  106 35 45 56 24 25 5 13 76 27 ...
##  $ Rush.Attempts                      : int  363 567 575 508 557 576 665 607 448 619 ...
##  $ Rush.Yds                           : int  1484 2893 2546 2194 2834 2813 3558 3321 1841 2802 ...
##  $ Yards.Rush                         : num  4.09 5.1 4.43 4.32 5.09 4.88 5.35 5.47 4.11 4.53 ...
##  $ Rushing.TD                         : int  18 35 26 20 31 35 33 32 16 43 ...
##  $ Rushing.Yards.per.Game             : num  124 207 182 169 218 ...
##  $ Rushing.Def.Rank                   : int  43 4 68 49 12 103 93 67 88 16 ...
##  $ Opp.Rush.Attempts                  : int  457 452 581 511 429 565 436 493 475 485 ...
##  $ Opp.Rush.Yards.Alloweed            : int  1783 1434 2377 2046 1490 2668 2316 2194 2293 1530 ...
##  $ Yds.Rush.Allowed                   : num  3.9 3.17 4.09 4 3.47 4.72 5.31 4.45 4.83 3.15 ...
##  $ Opp.Rush.Touchdowns.Allowed        : int  14 5 19 17 11 34 19 22 24 17 ...
##  $ Rush.Yards.Per.Game.Allowed        : num  149 102 170 157 115 ...
##  $ Sack.Rank                          : int  56 14 105 109 11 85 19 15 12 47 ...
##   [list output truncated]
```

The first thing I realized was that

A. I had a lot of data that I likely didn't really need to use and that B. I was going to need to clean my team names because the format was 'Team (Conference)' C. The data frames that I downloaded all had differing rows (meaning a different number of teams) and differing columns (meaning a different number of variables)

Let's address B first

My first step was to split the strings based on the first parenthesis ('('), and then output the split result into two separate columns. I did so using stringr. I then used gsub to get rid of the last parenthesis and trimws to get rid of any remaining blank spaces.

```
#Import stringr
library(stringr)
cfb_2014[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2014$Team, '\\(', 2)
#https://www.geeksforgeeks.org/how-to-split-column-into-multiple-columns-in-r-dataframe/
cfb_2014$Conference <- gsub("\\)", "", cfb_2014$Conference)
cfb_2014$Team <- trimws(cfb_2014$Team)
#https://study.com/academy/lesson/removing-space-from-string-in-r-programming.html#:~:text=R%20has%20so

cfb_2015[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2015$Team, '\\(', 2)
cfb_2015$Conference <- gsub("\\)", "", cfb_2015$Conference)
cfb_2015$Team <- trimws(cfb_2015$Team)

cfb_2016[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2016$Team, '\\(', 2)
cfb_2016$Conference <- gsub("\\)", "", cfb_2016$Conference)
cfb_2016$Team <- trimws(cfb_2016$Team)

cfb_2017[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2017$Team, '\\(', 2)
```

```
cfb_2017$Conference <- gsub("\\)", "", cfb_2017$Conference)
cfb_2017$Team <- trimws(cfb_2017$Team)

cfb_2018[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2018$Team, '\\(', 2)
cfb_2018$Conference <- gsub("\\)", "", cfb_2018$Conference)
cfb_2018$Team <- trimws(cfb_2018$Team)

cfb_2019[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2019$Team, '\\(', 2)
cfb_2019$Conference <- gsub("\\)", "", cfb_2019$Conference)
cfb_2019$Team <- trimws(cfb_2019$Team)

cfb_2020[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2020$Team, '\\(', 2)
cfb_2020$Conference <- gsub("\\)", "", cfb_2020$Conference)
cfb_2020$Team <- trimws(cfb_2020$Team)

cfb_2021[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2021$Team, '\\(', 2)
cfb_2021$Conference <- gsub("\\)", "", cfb_2021$Conference)
cfb_2021$Team <- trimws(cfb_2021$Team)

cfb_2022[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2022$Team, '\\(', 2)
cfb_2022$Conference <- gsub("\\)", "", cfb_2022$Conference)
cfb_2022$Team <- trimws(cfb_2022$Team)

cfb_2023[ , c('Team', 'Conference')] <- str_split_fixed(cfb_2023$Team, '\\(', 2)
cfb_2023$Conference <- gsub("\\)", "", cfb_2023$Conference)
cfb_2023$Team <- trimws(cfb_2023$Team)
```

Now that I had cleaner names, I decided to design the actual structure of my temporary data frames.

The first way I tried aggregating the data was by using the numerical value of each column and then feeding it into my custom function. However, that process was slow and rather tedious because I would have to check each column for each data frame because the frames all varied in column number and row number.

So for the 2015 data frame, I decided to try and access the columns based on their name, not their number.

That was much smoother operation as the column names were consistent across the various frames. Knowing this method would work, I decided to construct a function to quickly compile the metrics I needed.

```
data_constructor <- function(year){

  #I used paste here to create the variable names so that I could only input the year I needed for the

  #https://www.digitalocean.com/community/tutorials/paste-in-r

  new_frame <- paste("CFB", year, "Results", sep = "_")

  #https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/get
  #I also had to use get so that the function could pull in each existing frame
  old_frame <- get(paste("cfb", year, sep = "_"))

  #and then I constructed the frame template
  new_frame <- data.frame('Primary Key' = character(nrow(old_frame)),
                          'Name' = character(nrow(old_frame)),
                          'Conference' = character(nrow(old_frame)),
                          'Year' = character(nrow(old_frame)),
```

```
                                'O Rating' = double(nrow(old_frame)),
                                'D Rating' = double(nrow(old_frame)),
                                'Net Rating' = double(nrow(old_frame)),
                                 'Wins' = numeric(nrow(old_frame))
                                )
  #made a primary key so I could merge all the tables
  new_frame$Primary.Key <- paste(old_frame$Team, year, sep = "_")
  #loaded in the conferences
  new_frame$Conference <- old_frame$Conference
  #loaded in the team names
  new_frame$Name <- old_frame$Team
  #and put in the years
  new_frame$Year <- rep(year, nrow(old_frame))
  new_frame$Wins <- as.numeric(old_frame$Win)

#and then did the actual calculations for offense by doing a nested function call
new_frame$O.Rating <- apply(old_frame, MARGIN = 1, FUN = function(row) {
  team_rating_func(
    Pass_Comp = as.numeric(row["Pass.Completions"]),
    Pass_Att = as.numeric(row["Pass.Attempts"]),
    Pass_Yards = as.numeric(row["Pass.Yards"]),
    Pass_TD = as.numeric(row["Pass.Touchdowns"]),
    Rush_Att = as.numeric(row["Rush.Attempts"]),
    Rush_Yards = as.numeric(row["Rush.Yds"]),
    Rush_TD = as.numeric(row["Rushing.TD"]),
    Fumbles = as.numeric(row["Fumbles.Lost"]),
    Ints = as.numeric(row["Interceptions.Thrown.x"]),
    Games = as.numeric(row["Games"])
  )
})

#and then the calculations for defense by doing a nested function call
new_frame$D.Rating <- apply(old_frame, MARGIN = 1, FUN = function(row) {
  team_rating_func(
    Pass_Comp = row["Opp.Completions.Allowed"],
    Pass_Att = row["Opp.Pass.Attempts"],
    Pass_Yards = row["Opp.Pass.Yds.Allowed"],
    Pass_TD = row["Opp.Pass.TDs.Allowed"],
    Rush_Att = row["Opp.Rush.Attempts"],
    Rush_Yards = row["Opp.Rush.Yards.Alloweed"], #this is an intentional typo
    Rush_TD = row["Opp.Rush.Touchdowns.Allowed"],
    Fumbles = row["Fumbles.Recovered"],
    Ints = row["Opponents.Intercepted"],
    Games = row["Games"]
  )
})

#finally created the net ratings
new_frame$Net.Rating <- new_frame$O.Rating - new_frame$D.Rating

#and returned the frame
return(new_frame)
```

```
}
```

I used 2014 to make sure my function worked, and then ran 2014-2020 through my function.

```
CFB_2014_Results <- data_constructor(2014)
CFB_2015_Results <- data_constructor(2015)
CFB_2016_Results <- data_constructor(2016)
CFB_2017_Results <-  data_constructor(2017)
CFB_2018_Results <-  data_constructor(2018)
CFB_2019_Results <- data_constructor(2019)
CFB_2020_Results <- data_constructor(2020)
```

This model worked really well until I got to the last three frames, which had a handful of renamed columns. YAY! So, I had to rename them.

In addition, because I wanted to pull wins (something I decided to change at the last minute) I then had to reuse the string split function from earlier.

2021 was a bit of a head scratcher. A lot of columns that you would expect to be numeric (such as pass yards, rush yards, and their defensive counterparts) where character values.

```
#For some reason the 2021 data frame was fairly poor. A lot of columns that were numeric in nature were
cfb_2021$Interceptions.Thrown.x <- cfb_2021[ ,148]
cfb_2021$Pass.Yards <- as.numeric(gsub(",","", cfb_2021$Pass.Yards))
cfb_2021$Rush.Yds <- as.numeric(gsub(",","", cfb_2021$Rush.Yds))
cfb_2021$Opp.Pass.Yds.Allowed <- as.numeric(gsub(",","", cfb_2021$Opp.Pass.Yds.Allowed))
cfb_2021$Opp.Rush.Yards.Alloweed <- as.numeric(gsub(",","", cfb_2021$Opp.Rush.Yards.Alloweed))
cfb_2021[ , c('Win', 'Loss')] <- str_split_fixed(cfb_2021$Win.Loss, '-', 2)
CFB_2021_Results <-  data_constructor(2021)
```

2022 still required some cleaning, but was the easiest of the set.

```
cfb_2022$Interceptions.Thrown.x <- cfb_2022$Interceptions.Thrown_y
cfb_2022[ , c('Win', 'Loss')] <- str_split_fixed(cfb_2022$Win.Loss, '-', 2)
CFB_2022_Results <- data_constructor(2022)
```

I also found that I had an NA by coercion issue for the 2023 data frame. Row 131 seemed to have some sort of weird issue where one of the values was 'reclassifying', likely some sort of error on the part of whoever put the data frame together. So I dropped it, and then cleaned up the pass completion and pass completion allowed values.

```
cfb_2023 <- cfb_2023[ -131, ]
cfb_2023$Interceptions.Thrown.x <- cfb_2023$Interceptions.Thrown_x
cfb_2023$Pass.Completions <- as.numeric(gsub(",","", cfb_2023$Pass.Completions))
cfb_2023$Opp.Completions.Allowed <- as.numeric(gsub(",","", cfb_2023$Opp.Completions.Allowed))
cfb_2023[ , c('Win', 'Loss')] <- str_split_fixed(cfb_2023$Win.Loss, '-', 2)
CFB_2023_Results <-  data_constructor(2023)
```

Now with the data cleaned, I merged all the existing databases into one main table

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#Isn't dplyr a wonderful thing? https://stackoverflow.com/questions/68287677/merging-multiple-dataframe

final_set <- bind_rows(CFB_2014_Results, CFB_2015_Results, CFB_2016_Results, CFB_2017_Results, CFB_2018
                       CFB_2021_Results, CFB_2022_Results, CFB_2023_Results)
```

One thing that did come up when I was trying to graph were team names in my data base not being the same as the ones in the CFBFastR Package. The first one I noticed was USC, which is listed as Southern California in my set. So I ran the graph a few times, saw which teams the computer read in as errors and modified them manually. The names did change a few times, so I had to adjust the same teams multiple times.

```
final_set$Name[final_set$Name == 'Southern California'] <- 'USC'
final_set$Name[final_set$Name == 'Army West Point'] <- 'Army'
final_set$Name[final_set$Name == 'Eastern Mich.'] <- 'Eastern Michigan'
final_set$Name[final_set$Name == 'Fla. Atlantic'] <- 'Florida Atlantic'
final_set$Name[final_set$Name == 'Middle Tenn.'] <- 'Middle Tennessee'
final_set$Name[final_set$Name == 'Southern Miss.'] <- 'Southern Mississippi'
final_set$Name[final_set$Name == 'Western Ky.'] <- 'Western Kentucky'
final_set$Name[final_set$Name == 'Western Mich.'] <- 'Western Michigan'
final_set$Name[final_set$Name == 'La.-Monroe'] <- 'Louisiana Monroe'
final_set$Name[final_set$Name == 'Northern Ill.'] <- 'Northern Illinois'
final_set$Name[final_set$Name == 'South Fla.'] <- 'South Florida'
final_set$Name[final_set$Name == 'Ga. Southern'] <- 'Georgia Southern'
final_set$Name[final_set$Name == 'Central Mich.'] <- 'Central Michigan'
final_set$Name[final_set$Name == 'ULM'] <- 'Louisiana Monroe'
final_set$Name[final_set$Name == 'App State'] <- 'Appalachian State'
final_set$Name[final_set$Name == 'NIU'] <- 'Northern Illinois'
final_set$Name[final_set$Name == 'Coastal Caro.'] <- 'Coastal Carolina'
```

I also realized that Miami Florida and Miami Ohio were named Miami (OH), Miami (FL) so my parenthetical split operation didn't work on them. So I had to make some further alterations to clean up both the name and their column.

```
final_set$Name[final_set$Conference == 'OH (MAC'] <- 'Miami (OH)'
final_set$Name[final_set$Conference == 'FL (ACC'] <- 'Miami'
final_set$Conference[final_set$Name == 'Miami (OH)'] <- 'MAC'
final_set$Conference[final_set$Name== 'Miami'] <- 'ACC'
```

I then needed to do some conference renaming for a few values that were named differently across multiple frames.

```
final_set$Conference[final_set$Name == 'Ole Miss'] <- 'SEC'
final_set$Conference[final_set$Name == 'Pittsburgh'] <- 'ACC'
final_set$Conference[final_set$Name == 'Notre Dame'] <- 'FBS Independent'
final_set$Conference[final_set$Conference == 'MWC'] <- 'Mountain West'
```

Finally, I altered the rating for any team that is traditionally considered to be Group of 5. This lowered their values and kept the values of the 'Power 5' teams normal.

```
final_set$O.Rating <- final_set$O.Rating * ifelse(
    (final_set$Conference == 'MAC' |
    final_set$Conference == 'AAC' |
    final_set$Conference == 'C-USA'|
    final_set$Conference == 'Mountain West' |
    final_set$Conference == 'Sun Belt')
    |
    (final_set$Conference =='FBS Independent' & final_set$Name != 'Notre Dame') , .75, 1)

final_set$D.Rating <- final_set$D.Rating * ifelse(
  (final_set$Conference == 'MAC' |
    final_set$Conference == 'AAC' |
    final_set$Conference == 'C-USA'|
    final_set$Conference == 'Mountain West' |
    final_set$Conference == 'Sun Belt') |
    (final_set$Conference =='FBS Independent' & final_set$Name != 'Notre Dame')
    , 1.25, 1)

final_set$Net.Rating <- final_set$O.Rating - final_set$D.Rating
```

Now, with all 1253 teams defined and in one database, it's time to start answering the question that lead me to do all this: who are the top-25 teams in the CFP era. I started by using dplyr in the frame, filtering to only include Power 5 teams, as well as Notre Dame.

```
library(dplyr)

top_25 <- final_set %>%
  filter(Conference == 'SEC' | Conference == 'ACC' |
           Conference == 'Pac-12' | Conference == 'Big Ten' |
           Conference == 'Big 12'| Name == "Notre Dame") %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

top_25
```

```
##          Primary.Key         Name Conference Year O.Rating D.Rating Net.Rating
## 1       Clemson_2018      Clemson        ACC 2018 207.2833 26.79333   180.4900
## 2      Ohio St._2019     Ohio St.    Big Ten 2019 202.8857 27.65714   175.2286
## 3       Clemson_2019      Clemson        ACC 2019 205.1767 33.82000   171.3567
## 4        Oregon_2023       Oregon     Pac-12 2023 226.2893 69.34286   156.9464
## 5           TCU_2014          TCU     Big 12 2014 176.6462 22.58846   154.0577
## 6           LSU_2019          LSU        SEC 2019 223.0133 70.61667   152.3967
## 7       Alabama_2020      Alabama        SEC 2020 215.8318 64.82273   151.0091
## 8       Clemson_2020      Clemson        ACC 2020 182.9091 34.01364   148.8955
```

```
## 9       Alabama_2019      Alabama        SEC 2019 197.2654 51.25385   146.0115
## 10      Alabama_2017      Alabama        SEC 2017 163.2643 22.69286   140.5714
## 11      Alabama_2016      Alabama        SEC 2016 152.0800 16.15333   135.9267
## 12     Ohio St._2017     Ohio St.    Big Ten 2017 181.8607 46.10000   135.7607
## 13     Ohio St._2016     Ohio St.    Big Ten 2016 162.1346 27.81154   134.3231
## 14      Alabama_2018      Alabama        SEC 2018 193.0600 59.08000   133.9800
## 15      Georgia_2022      Georgia        SEC 2022 184.7100 51.80000   132.9100
## 16      Georgia_2021      Georgia        SEC 2021 158.2000 27.56333   130.6367
## 17      Georgia_2023      Georgia        SEC 2023 187.8571 58.50000   129.3571
## 18     Ohio St._2020     Ohio St.    Big Ten 2020 198.3083 70.23333   128.0750
## 19       Baylor_2014       Baylor     Big 12 2014 203.4500 76.07692   127.3731
## 20     Michigan_2016     Michigan    Big Ten 2016 150.0231 24.90385   125.1192
## 21 Michigan St._2014 Michigan St.    Big Ten 2014 175.0654 50.55000   124.5154
## 22     Ohio St._2021     Ohio St.    Big Ten 2021 215.2923 93.40000   121.8923
## 23     Michigan_2023     Michigan    Big Ten 2023 145.7967 25.05667   120.7400
## 24     Michigan_2022     Michigan    Big Ten 2022 170.7643 50.20357   120.5607
## 25       Baylor_2015       Baylor     Big 12 2015 212.8846 92.65769   120.2269
##    Wins
## 1    15
## 2    13
## 3    14
## 4    12
## 5    12
## 6    15
## 7    11
## 8    10
## 9    11
## 10   13
## 11   14
## 12   12
## 13   11
## 14   14
## 15   15
## 16   14
## 17   13
## 18    6
## 19   11
## 20   10
## 21   11
## 22   11
## 23   15
## 24   13
## 25   10
```

Time to try graphing!

```
if (!require("remotes")) install.packages("remotes")
```

```
## Loading required package: remotes
```

```
## Warning: package 'remotes' was built under R version 4.4.1
```

```
remotes::install_github("Kazink36/cfbplotR")
```

```
## Using GitHub PAT from the git credential store.
```

```
## Skipping install of 'cfbplotR' from a github remote, the SHA1 (b45dddf6) has not changed since last i
##   Use 'force = TRUE' to force installation
```

```
library(cfbfastR)
library(cfbplotR)
library(tidyverse)
```

```
## Warning: package 'purrr' was built under R version 4.4.1
```

```
## Warning: package 'lubridate' was built under R version 4.4.1
```

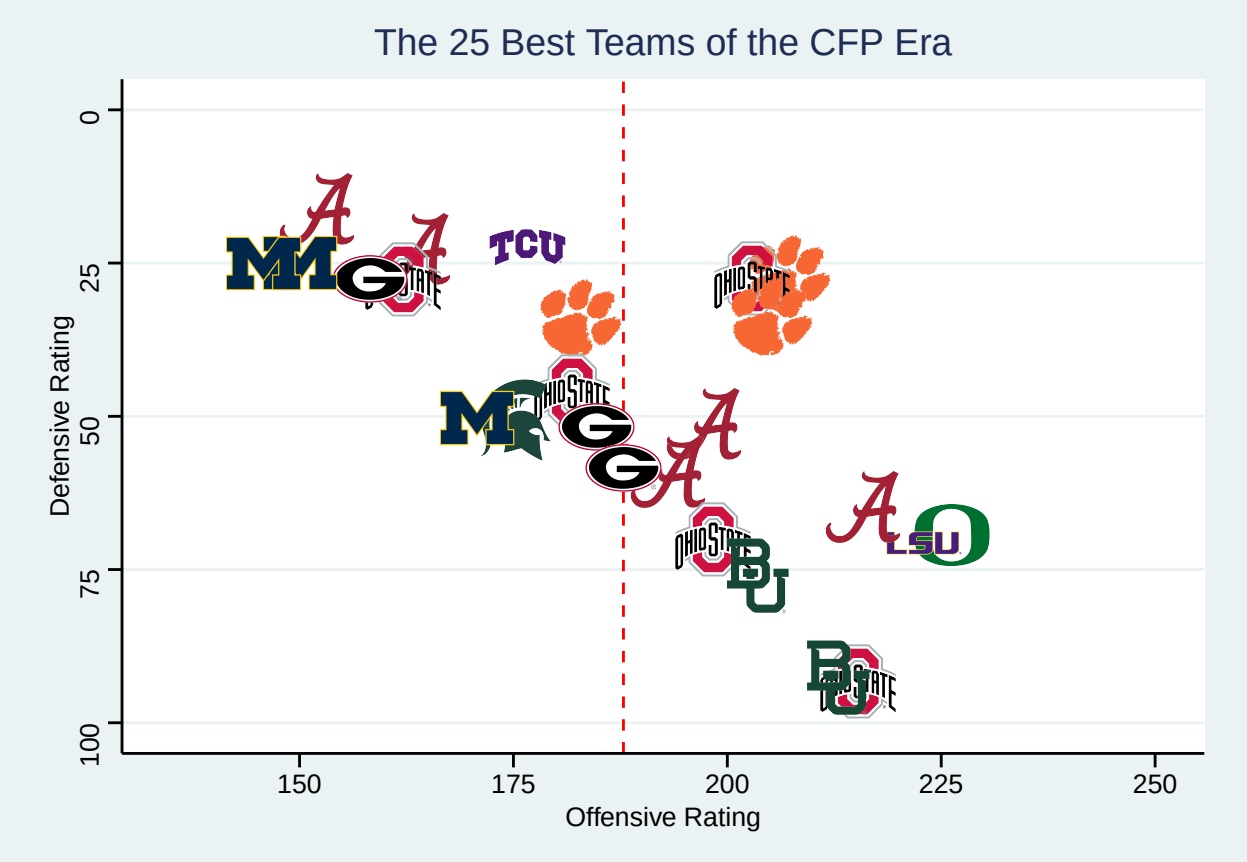
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
```

```
## -- 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(ggthemes)

ggplot(top_25, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "The 25 Best Teams of the CFP Era",  x = "Offensive Rating", y = "Defensive Rating") +
  theme_stata()+
  ylim(100, 0) +
  xlim(135, 250)
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
## Warning: Using the 'size' aesthetic with geom_segment was deprecated in ggplot2 3.4.0.
## i Please use the 'linewidth' aesthetic instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

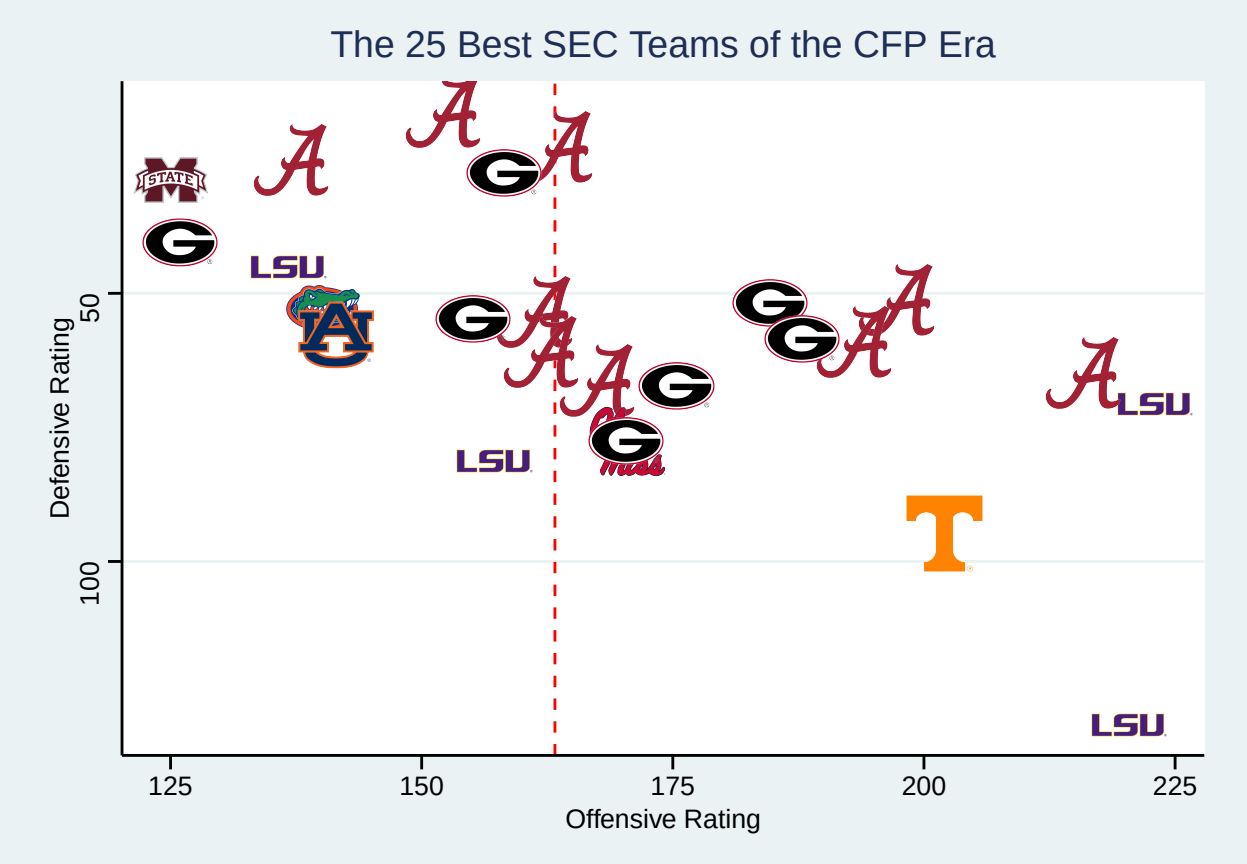



```
#https://kazink36.github.io/cfbplotR/
```

And just like that, I have my answer. The best teams in CFP area include: Alabama, Clemson, Ohio State, LSU, Oregon, Michigan, TCU and Georgia. The singular best team was Clemson 2018, closely followed by Ohio State 2019 and Clemson 2019.

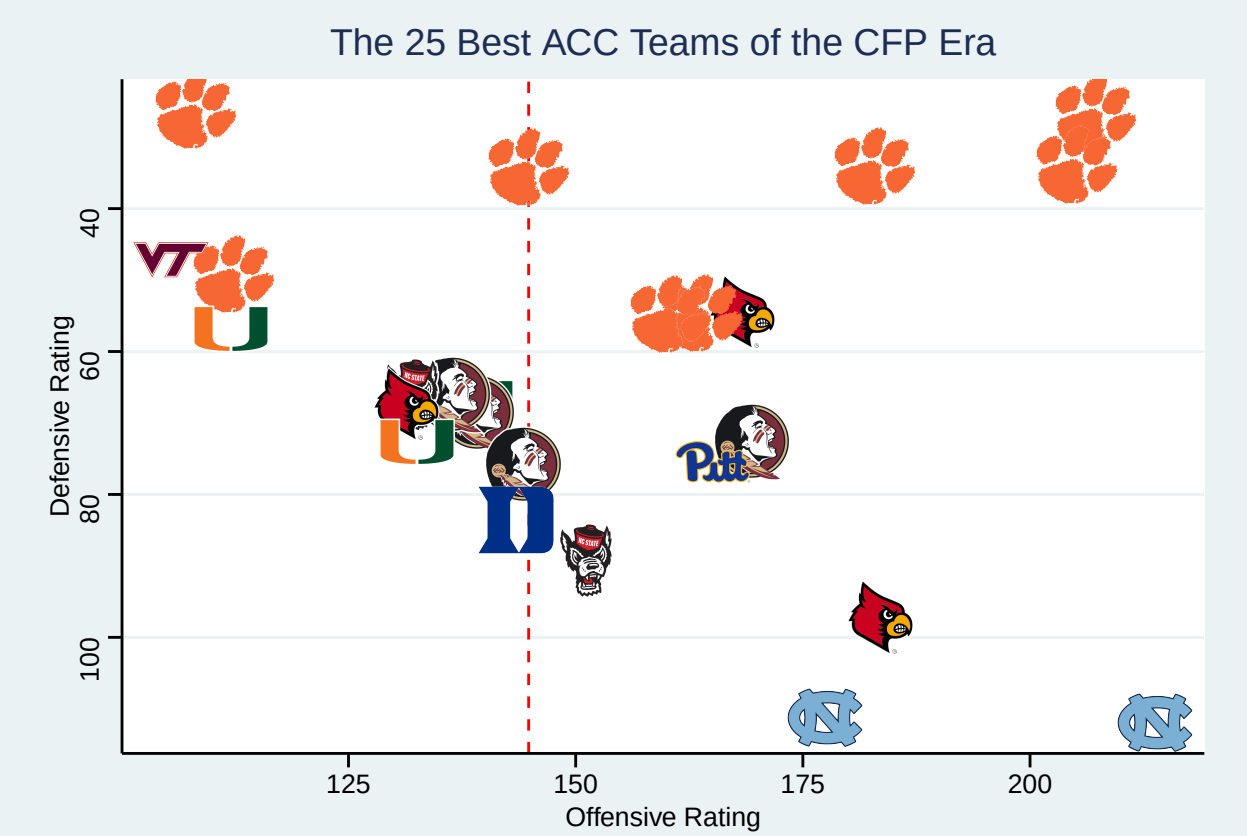
But I couldn't quite stop there. I decided to throw in a top-25 SEC graph.

```
top_25_SEC <- final_set %>%
  filter(Conference == 'SEC') %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

ggplot(top_25_SEC, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "The 25 Best SEC Teams of the CFP Era",  x = "Offensive Rating", y = "Defensive Rating")
  theme_stata()
```



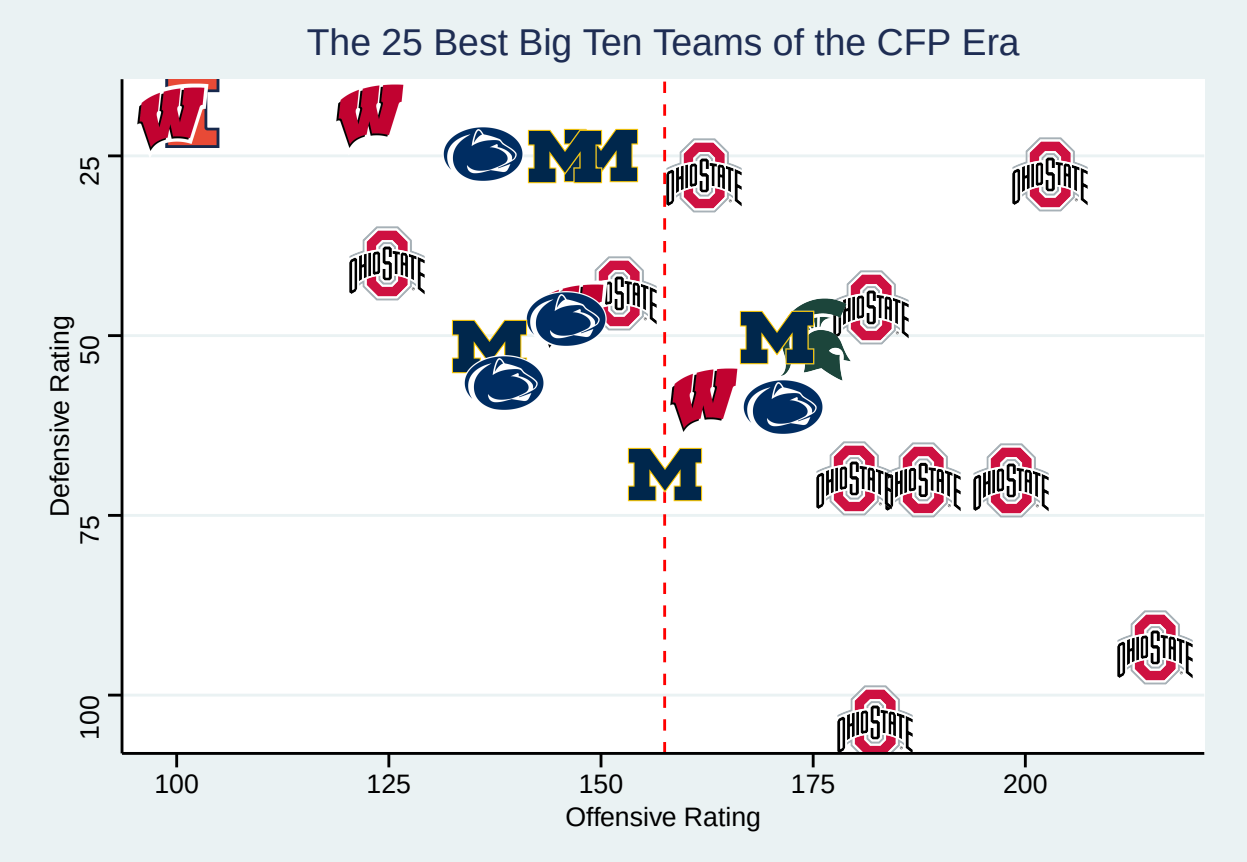
Which led to an ACC graph.

```
top_25_ACC <- final_set %>%
  filter(Conference == 'ACC') %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

ggplot(top_25_ACC, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "The 25 Best ACC Teams of the CFP Era",  x = "Offensive Rating", y = "Defensive Rating") +
  theme_stata()
```



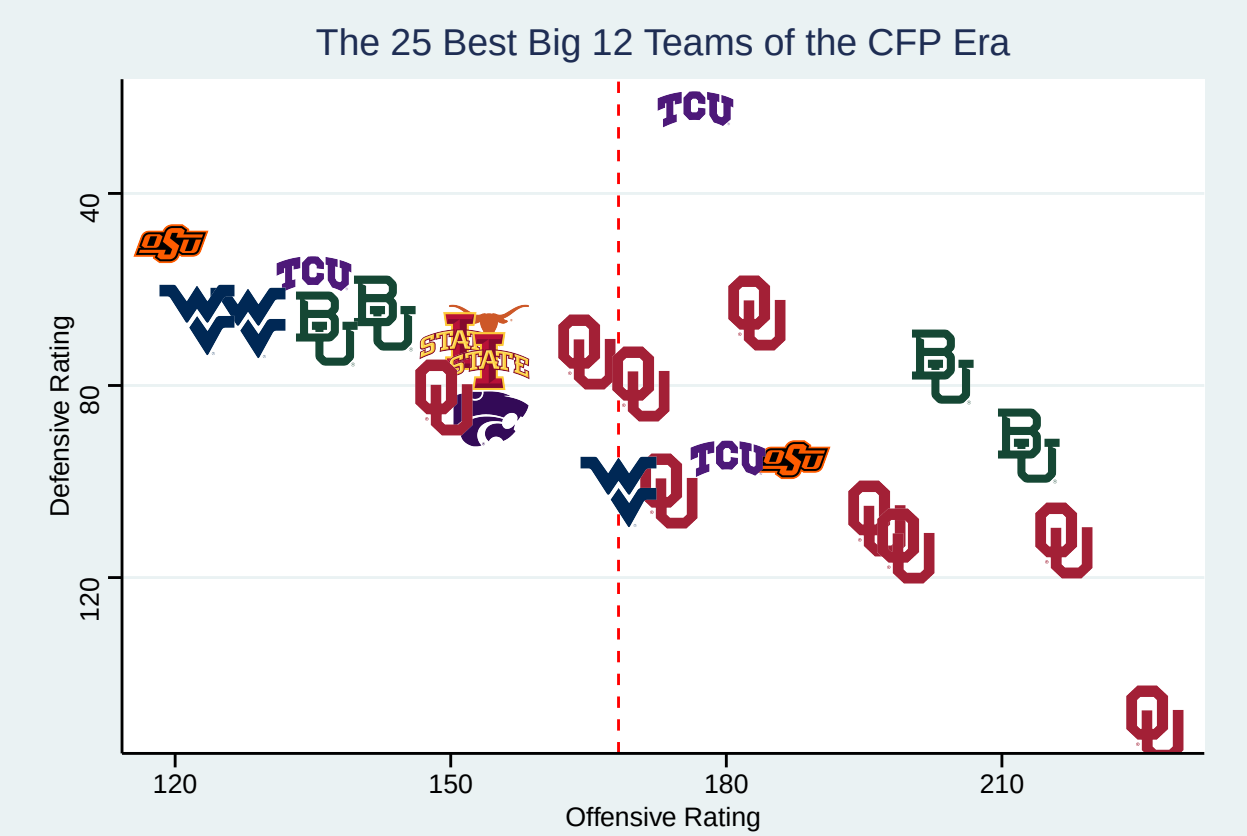
And then I had to do a Big Ten graph.

```
top_25_Big10 <- final_set %>%
  filter(Conference == 'Big Ten') %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

ggplot(top_25_Big10, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "The 25 Best Big Ten Teams of the CFP Era",  x = "Offensive Rating", y = "Defensive Rati
  theme_stata()
```



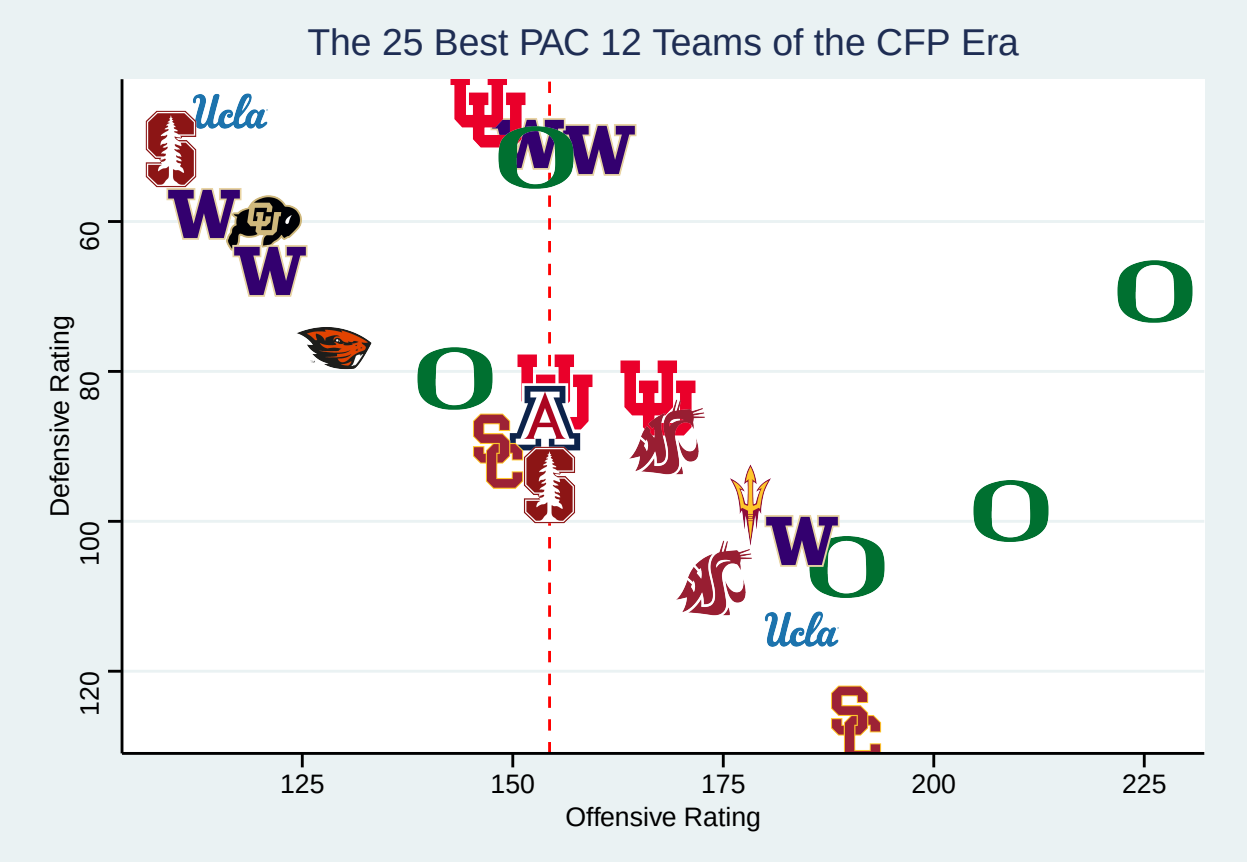
Which led to a Big 12 graphic.

```
top_25_Big12 <- final_set %>%
  filter(Conference == 'Big 12') %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

ggplot(top_25_Big12, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "The 25 Best Big 12 Teams of the CFP Era",  x = "Offensive Rating", y = "Defensive Ratin
  theme_stata()
```



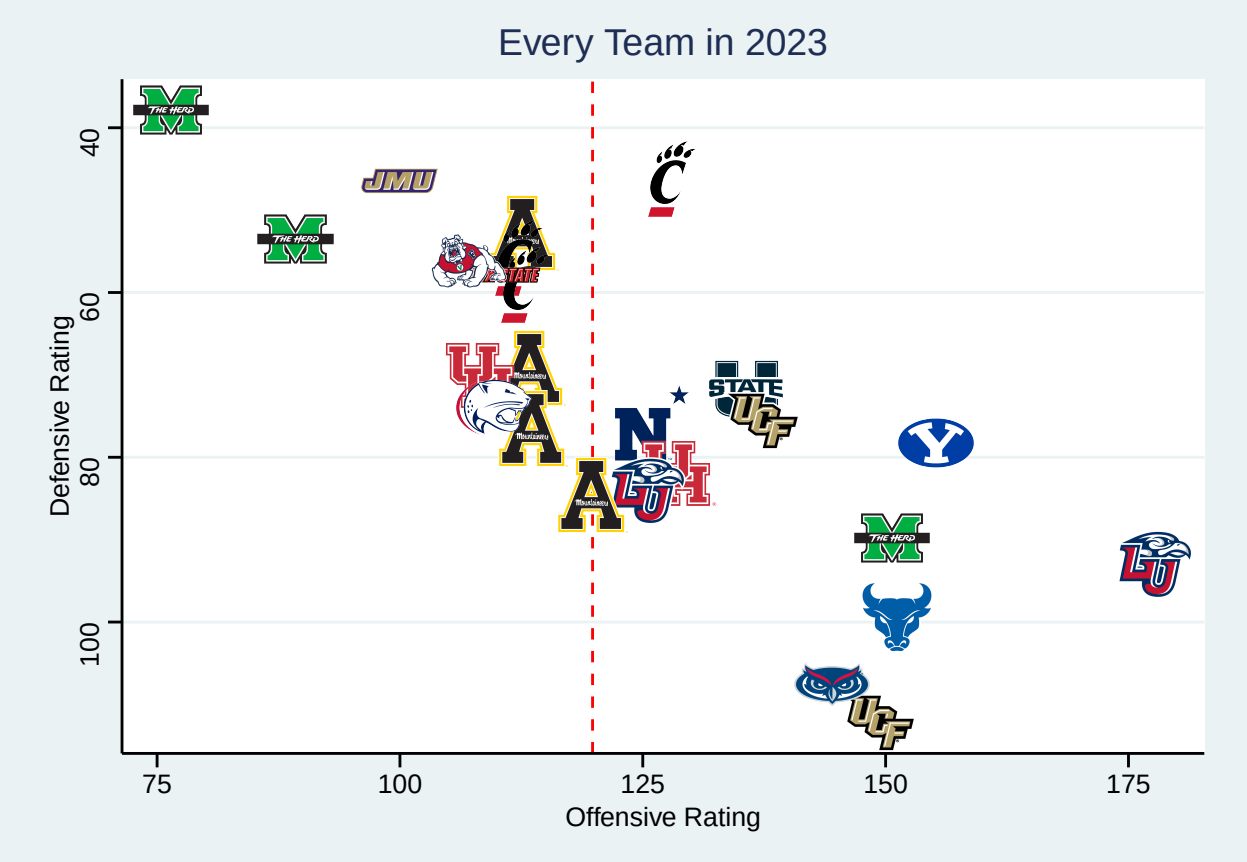
And then I honored the PAC-12 by offering them one last graphic.

```
top_25_PAC12 <- final_set %>%
  filter(Conference == 'Pac-12') %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

ggplot(top_25_PAC12, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "The 25 Best PAC 12 Teams of the CFP Era",  x = "Offensive Rating", y = "Defensive Rating
  theme_stata()
```



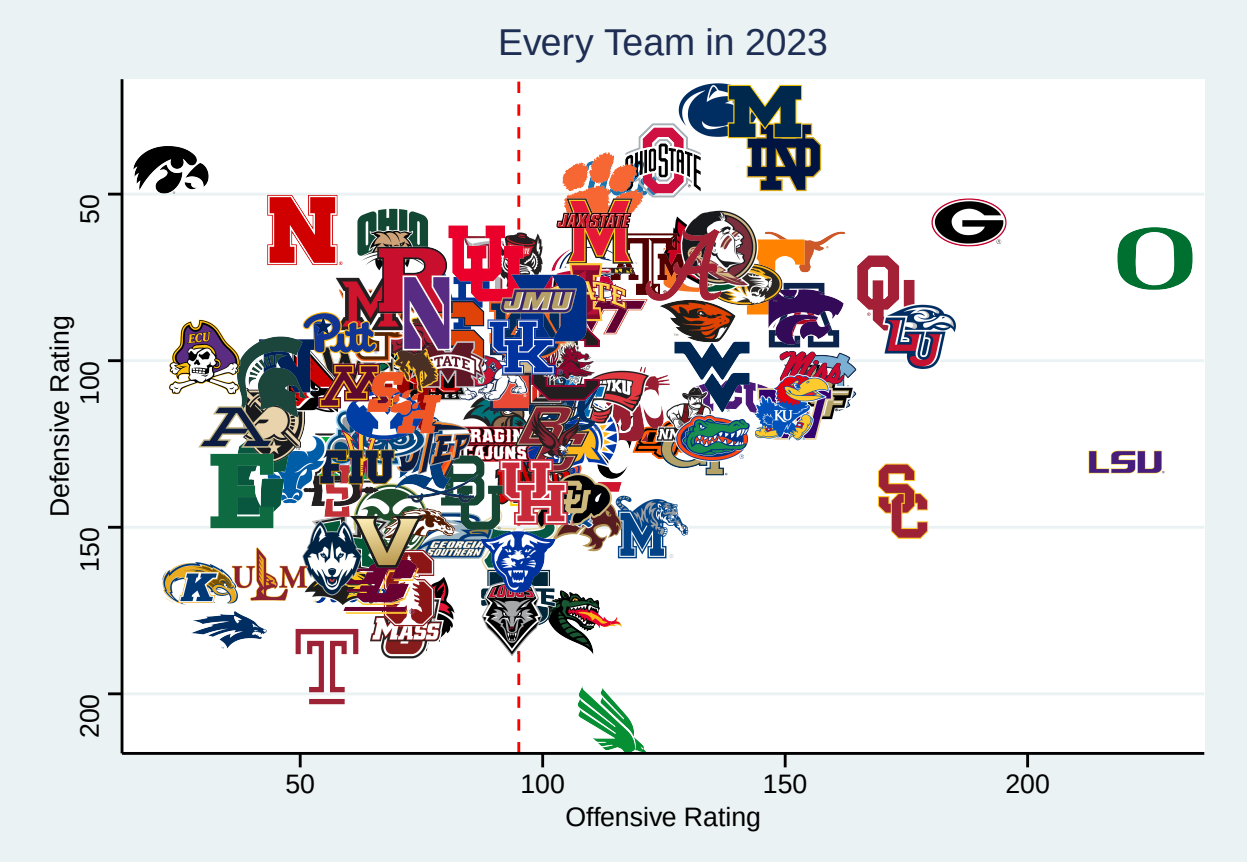
And because I hate ignoring the Group of 5 Conferences, I decided to give them a shoutout too.

```
top_25_G5 <- final_set %>%
  filter(Conference != 'SEC' & Conference != 'ACC' &
           Conference != 'Pac-12' & Conference != 'Big Ten' &
           Conference != 'Big 12' & Name != "Notre Dame") %>%
  arrange(desc(Net.Rating)) %>%
  slice(1:25)

ggplot(top_25_G5, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "Every Team in 2023",  x = "Offensive Rating", y = "Defensive Rating") +
  theme_stata()
```



And more just out of intellectually curiosity, I wanted to see how long it would take to graph every team from 2023. The answer was 30 seconds.

```
season_2023 <- final_set %>%
  filter(Year == 2023)

ggplot(season_2023, aes(x = O.Rating, y = D.Rating)) +
  geom_median_lines(aes(v_var = O.Rating, h_var = D.Rating)) +
  geom_cfb_logos(aes(team = Name), width = 0.075) +
  scale_y_reverse() +
  labs(title = "Every Team in 2023",  x = "Offensive Rating", y = "Defensive Rating") +
  theme_stata()
```



Can you derive a ton of meaning out of it? No, I'll admit as much. But it's also something I have wanted to have the ability to do for a long time and am thrilled to have a chance to do so.

```
(final_25 <- aggregate(
  cbind(Net.Rating, O.Rating, D.Rating)
  ~ Name + Conference,
  data = final_set,
  FUN = mean))
```

```
##                    Name      Conference  Net.Rating  O.Rating  D.Rating
## 1             Charlotte             AAC  -93.271875  41.69687 134.96875
## 2            Cincinnati             AAC   -6.491144  92.92696  99.41811
## 3          East Carolina            AAC  -76.946651  81.86252 158.80917
## 4       Florida Atlantic            AAC  -55.734375  68.00000 123.73438
## 5               Houston             AAC  -23.912257 100.99813 124.91038
## 6               Memphis             AAC  -19.340658 112.91308 132.25374
## 7                  Navy             AAC  -46.931866  80.58029 127.51216
## 8           North Texas             AAC  -94.369792 114.31250 208.68229
## 9                  Rice             AAC  -52.875962  74.79231 127.66827
## 10                  SMU             AAC  -68.811376  93.22261 162.03399
## 11         South Florida            AAC  -71.601434  85.63182 157.23325
## 12               Temple             AAC  -59.391399  61.74539 121.13679
## 13               Tulane             AAC  -48.717911  77.98462 126.70253
## 14                Tulsa             AAC  -74.648013  82.34786 156.99587
## 15                  UAB             AAC  -70.156250 108.45312 178.60938
## 16                  UCF             AAC  -16.034429 100.42364 116.45806
```

```
## 17                UConn            AAC -141.572610  49.82736 191.39997
## 18                 UTSA            AAC  -11.039423  95.07115 106.11058
## 19       Boston College            ACC   -4.147491  87.38207  91.52956
## 20              Clemson            ACC  107.331567 150.45048  43.11892
## 21                 Duke            ACC   -4.659674 100.19528 104.85495
## 22          Florida St.            ACC   29.512863 119.85113  90.33826
## 23         Georgia Tech            ACC   -8.258402 107.86779 116.12619
## 24           Louisville            ACC   33.264465 126.62343  93.35896
## 25                Miami            ACC   35.169910 114.44490  79.27499
## 26             NC State            ACC   26.771559 115.38225  88.61069
## 27       North Carolina            ACC   26.222215 147.59254 121.37033
## 28           Pittsburgh            ACC   21.494534 111.72179  90.22726
## 29             Syracuse            ACC  -20.214776  92.08159 112.29637
## 30             Virginia            ACC   -9.209760  95.52665 104.73641
## 31        Virginia Tech            ACC    9.884548 101.61511  91.73057
## 32          Wake Forest            ACC  -11.507873  98.17638 109.68425
## 33                Baylor         Big 12   28.732089 133.98991 105.25782
## 34                   BYU         Big 12  -49.054167  67.31250 116.36667
## 35            Cincinnati         Big 12  -12.133333 115.73333 127.86667
## 36               Houston         Big 12  -40.808333  98.73333 139.54167
## 37             Iowa St.          Big 12   12.241702 114.17921 101.93751
## 38                Kansas         Big 12  -72.564786  81.85057 154.41535
## 39             Kansas St.         Big 12    9.020996 114.16264 105.14164
## 40               Oklahoma         Big 12   81.284923 183.16562 101.88070
## 41         Oklahoma St.           Big 12   23.904725 132.00885 108.10413
## 42                  TCU           Big 12   45.871205 136.07530  90.20410
## 43                Texas           Big 12   29.762740 129.76992 100.00718
## 44            Texas Tech          Big 12    2.383853 143.82090 141.43704
## 45                  UCF           Big 12   47.096154 156.59231 109.49615
## 46         West Virginia          Big 12   24.427682 123.78500  99.35732
## 47              Illinois        Big Ten  -27.409952  81.07925 108.48920
## 48               Indiana        Big Ten  -11.810343  96.98087 108.79121
## 49                  Iowa        Big Ten   27.992926  82.66448  54.67156
## 50              Maryland        Big Ten  -10.123679 101.61822 111.74190
## 51              Michigan        Big Ten   67.082938 125.82622  58.74329
## 52          Michigan St.        Big Ten    4.333677  90.07501  85.74133
## 53             Minnesota        Big Ten    8.394254  97.26231  88.86806
## 54              Nebraska        Big Ten   -1.164054 105.92556 107.08962
## 55         Northwestern         Big Ten  -12.237624  73.96623  86.20386
## 56              Ohio St.        Big Ten  119.415449 178.75314  59.33769
## 57              Penn St.        Big Ten   57.298984 118.68532  61.38633
## 58                Purdue        Big Ten  -14.386603 101.24138 115.62799
## 59               Rutgers        Big Ten  -62.086688  61.47118 123.55786
## 60             Wisconsin        Big Ten   68.859377 116.80444  47.94506
## 61             Charlotte          C-USA -104.333780  70.17109 174.50486
## 62                   FIU          C-USA  -94.496325  64.63236 159.12869
## 63     Florida Atlantic           C-USA  -54.673385  87.59362 142.26700
## 64       Louisiana Tech           C-USA  -46.725320  87.40614 134.13146
## 65              Marshall          C-USA   -8.370020  89.75012  98.12014
## 66     Middle Tennessee           C-USA  -49.901366  88.54734 138.44871
## 67           North Texas          C-USA  -81.882495  83.80851 165.69100
## 68          Old Dominion          C-USA  -82.981985  69.62812 152.61011
## 69                  Rice          C-USA -105.220634  59.46704 164.68767
## 70  Southern Mississippi          C-USA  -45.377145  72.16559 117.54273
```

```
## 71                  UAB           C-USA  -12.522186  82.91664  95.43882
## 72                 UTEP           C-USA -101.144600  54.64842 155.79302
## 73                 UTSA           C-USA  -51.766741  70.81821 122.58495
## 74      Western Kentucky           C-USA  -20.548351 105.02633 125.57469
## 75                  FIU            CUSA  -70.175000  61.72500 131.90000
## 76      Jacksonville St.           CUSA   52.650000 110.55769  57.90769
## 77               Liberty           CUSA   84.732143 177.74286  93.01071
## 78         Louisiana Tech           CUSA  -38.212500 105.28750 143.50000
## 79      Middle Tennessee            CUSA   -4.583333 105.90833 110.49167
## 80        New Mexico St.            CUSA   14.886667 130.90667 116.02000
## 81           Sam Houston            CUSA  -41.291667  71.45000 112.74167
## 82                 UTEP            CUSA  -48.625000  77.38333 126.00833
## 83      Western Kentucky           CUSA    3.865385 114.76538 110.90000
## 84                 Army FBS Independent  -32.667483  83.39813 116.06561
## 85                  BYU FBS Independent  -14.048210  97.64757 111.69578
## 86               Liberty FBS Independent   -1.371923 102.21894 103.59087
## 87         Massachusetts FBS Independent -150.644792  51.24063 201.88542
## 88                 Navy FBS Independent  -42.592308 105.10962 147.70192
## 89         New Mexico St. FBS Independent -120.423237  62.71088 183.13411
## 90             Notre Dame FBS Independent   61.214324 138.29095  77.07663
## 91                 UConn FBS Independent -109.720994  47.10833 156.82933
## 92         New Mexico St.     Independent   17.496154 117.51538 100.01923
## 93                  Akron             MAC -108.398194  45.43709 153.83529
## 94               Ball St.             MAC  -82.885048  76.08576 158.97081
## 95          Bowling Green             MAC -108.992940  66.12452 175.11746
## 96                Buffalo             MAC  -43.110983  85.36221 128.47319
## 97      Central Michigan            MAC  -47.449543  73.20846 120.65800
## 98      Eastern Michigan            MAC  -89.918029  71.45204 161.37007
## 99               Kent St.            MAC  -89.503654  71.14014 160.64379
## 100        Massachusetts             MAC  -90.743229  72.54844 163.29167
## 101           Miami (OH)             MAC  -60.758024  62.96909 123.72712
## 102    Northern Illinois             MAC  -53.011268  78.00140 131.01267
## 103                 Ohio             MAC  -29.687470  90.58776 120.27523
## 104               Toledo             MAC  -17.419574 107.80668 125.22625
## 105     Western Michigan            MAC  -31.175928 100.45686 131.63279
## 106            Air Force   Mountain West   -7.697848  92.44907 100.14692
## 107            Boise St.   Mountain West  -10.157889  91.46871 101.62659
## 108         Colorado St.   Mountain West  -70.480540  70.33740 140.81794
## 109           Fresno St.   Mountain West   -5.604299  96.82208 102.42638
## 110               Hawaii   Mountain West  -89.219646  77.64449 166.86414
## 111               Nevada   Mountain West  -68.923905  75.09405 144.01795
## 112           New Mexico   Mountain West -103.072364  53.63374 156.70610
## 113        San Diego St.   Mountain West  -20.284189  64.72107  85.00526
## 114         San Jose St.   Mountain West  -67.345640  71.37793 138.72357
## 115                 UNLV   Mountain West  -93.540200  74.72341 168.26361
## 116             Utah St.   Mountain West  -58.860907  80.67723 139.53813
## 117              Wyoming   Mountain West  -30.877312  64.94663  95.82395
## 118              Arizona          Pac-12  -16.312940 126.71117 143.02411
## 119           Arizona St.         Pac-12    3.407276 119.26728 115.86000
## 120            California         Pac-12   -5.493622 110.39856 115.89218
## 121              Colorado         Pac-12  -29.348232  96.68401 126.03224
## 122                Oregon         Pac-12   63.162443 168.88080 105.71835
## 123            Oregon St.         Pac-12  -23.197839 113.11620 136.31404
## 124              Stanford         Pac-12  -11.364876 107.31106 118.67594
```

```
## 125                 UCLA          Pac-12   17.865929 128.37172 110.50579
## 126                  USC          Pac-12   34.185852 143.64652 109.46066
## 127                 Utah          Pac-12   46.209989 116.62786  70.41787
## 128           Washington          Pac-12   52.462524 132.36542  79.90290
## 129       Washington St.          Pac-12   23.316699 136.71253 113.39583
## 130              Alabama             SEC  118.694530 167.80980  49.11527
## 131             Arkansas             SEC   -8.813385 110.20813 119.02152
## 132               Auburn             SEC   30.418959 118.95592  88.53696
## 133              Florida             SEC   39.379742 118.11748  78.73774
## 134              Georgia             SEC   90.764878 148.01970  57.25482
## 135             Kentucky             SEC    4.076231 102.21315  98.13692
## 136                  LSU             SEC   58.594549 143.45795  84.86340
## 137      Mississippi St.             SEC   34.315436 123.18510  88.86967
## 138             Missouri             SEC   17.808775 114.92150  97.11272
## 139             Ole Miss             SEC   30.634904 149.74358 119.10868
## 140       South Carolina             SEC  -13.274179  96.96258 110.23676
## 141            Tennessee             SEC   17.444090 116.36983  98.92574
## 142            Texas A&M             SEC   32.929829 123.72395  90.79412
## 143           Vanderbilt             SEC  -62.159647  72.43831 134.59796
## 144      Appalachian St.       Sun Belt   37.177802 112.37918  75.20137
## 145    Appalachian State       Sun Belt    3.412612 106.67098 103.25837
## 146         Arkansas St.       Sun Belt  -60.448895  87.81756 148.26645
## 147     Coastal Carolina       Sun Belt  -37.286978 102.23998 139.52696
## 148     Georgia Southern       Sun Belt  -50.744569  81.30195 132.04652
## 149          Georgia St.       Sun Belt  -83.917139  77.36211 161.27925
## 150                Idaho       Sun Belt -100.612378  72.50165 173.11403
## 151        James Madison       Sun Belt   35.241914  99.53955  64.29764
## 152            Louisiana       Sun Belt  -37.096193  94.42111 131.51730
## 153     Louisiana Monroe       Sun Belt -118.140510  64.35459 182.49510
## 154             Marshall       Sun Belt  -15.511058  67.99615  83.50721
## 155       New Mexico St.       Sun Belt -135.495486  75.70417 211.19965
## 156         Old Dominion       Sun Belt  -74.153966  58.11346 132.26743
## 157        South Alabama       Sun Belt  -66.968910  66.26028 133.22918
## 158 Southern Mississippi       Sun Belt  -80.630288  56.02596 136.65625
## 159            Texas St.       Sun Belt -100.720321  68.59719 169.31751
## 160                 Troy       Sun Belt  -25.681644  82.96344 108.64509
```

```
(final_25 <- final_25 %>%
               filter(Conference == 'SEC' | Conference == 'ACC' |
                        Conference == 'Pac-12' | Conference == 'Big Ten' |
                        Conference == 'Big 12' | Name == 'Notre Dame') %>%
               arrange(desc(Net.Rating)) %>%
               slice(1:25))
```

```
##               Name      Conference Net.Rating O.Rating  D.Rating
## 1         Ohio St.         Big Ten  119.41545 178.7531  59.33769
## 2          Alabama             SEC  118.69453 167.8098  49.11527
## 3          Clemson             ACC  107.33157 150.4505  43.11892
## 4          Georgia             SEC   90.76488 148.0197  57.25482
## 5         Oklahoma          Big 12   81.28492 183.1656 101.88070
## 6        Wisconsin         Big Ten   68.85938 116.8044  47.94506
## 7         Michigan         Big Ten   67.08294 125.8262  58.74329
## 8           Oregon          Pac-12   63.16244 168.8808 105.71835
## 9       Notre Dame FBS Independent   61.21432 138.2910  77.07663
```

```
## 10             LSU            SEC   58.59455 143.4579  84.86340
## 11        Penn St.        Big Ten   57.29898 118.6853  61.38633
## 12      Washington         Pac-12   52.46252 132.3654  79.90290
## 13             UCF         Big 12   47.09615 156.5923 109.49615
## 14            Utah         Pac-12   46.20999 116.6279  70.41787
## 15             TCU         Big 12   45.87121 136.0753  90.20410
## 16         Florida            SEC   39.37974 118.1175  78.73774
## 17           Miami            ACC   35.16991 114.4449  79.27499
## 18 Mississippi St.            SEC   34.31544 123.1851  88.86967
## 19             USC         Pac-12   34.18585 143.6465 109.46066
## 20      Louisville            ACC   33.26446 126.6234  93.35896
## 21       Texas A&M            SEC   32.92983 123.7240  90.79412
## 22        Ole Miss            SEC   30.63490 149.7436 119.10868
## 23          Auburn            SEC   30.41896 118.9559  88.53696
## 24           Texas         Big 12   29.76274 129.7699 100.00718
## 25     Florida St.            ACC   29.51286 119.8511  90.33826
```

```
top_25_Years <- merge(final_25,
                      final_set,
                      by.x = 'Name',
                      by.y = 'Name')
```