Project March Madness

Paul Uhn

2023-12-04

Contents

March Madness

What is March Madness? It is a single-elimination men's college basketball tournament played mostly during the month of March. The tournament consists of 68 teams - 32 division 1 conference champions and 36 "at-large" teams chosen by the NCAA selection committee. The teams are placed in four groups (called regions) and given a seed (1 - 16) within a region. The tournament is seven rounds and is held over 3 weeks. The first week starts with 8 teams competing in the First Four with the winners joining the remaining 60 teams to compete in the first and second rounds. The second week are the regional semifinals and regional finals also know as Sweet Sixteen and Elite Eight, respectively. The final week are the national semifinals and national championships know collectively as the Final Four.

The tournament started in 1939 and has been going on for ~84 years. However in 2020, it was cancelled due to the COVID-19 pandemic.

The tournament has become a part of pop culture through bracket contests that award money and prizes for correctly predicting the outcomes of the most games.

The tournament bracket point system awards points for correctly picking the correct winner of each game.

round	points_per_game	number_of_games	total_points
1 - First Round	1	32	32
2 - Second Round	2	16	32
3 - Sweet Sixteen	4	8	32
4 - Elite Eight	8	4	32
5 - Final Four	16	2	32
6 - National Championship	32	1	32 = 192 overall points

So what's the best strategy for filling out a bracket? Do you need to have in-depth knowledge about each of the 68 teams and have both team and player stats on-hand? The NCAA took a look at their data and compared the average user score vs just picking the higher seeded team. [1] And it turns out the average user score is 67.1444444 compared to the average seed-based bracket score of 87.5555556.

So our goal will be to perform better than both the average bracket and the seed-based bracket. Instead of trying to figure out the upsets and the Cinderellas, we will focus on picking the Elite Eight teams. And if we can pick the right order with the champion followed by second, etc, our base score would be:

$$(1 \cdot 8) + (2 \cdot 8) + (4 \cdot 8) + (8 \cdot 4) + (16 \cdot 2) + (32 \cdot 1) = 152$$

But just getting the Elite Eight right would net us a base score of:

$$(1 \cdot 8) + (2 \cdot 8) + (4 \cdot 8) = 56$$

That still leaves 24 games for 24 points in the First Round and 8 games for 16 points in the Second Round.

But in case you think those scores are not that impressive, the chances of a perfect bracket is just 1 in 9.2 quintillion [2] (or a billion billions).

[1] https://www.ncaa.com/news/basketball-men/bracketiq/2021-02-12/heres-how-your-march-madness-bracket-will-do-if-you-only-pick-better-seeded

[2] https://www.ncaa.com/news/basketball-men/bracketiq/2023-03-16/perfect-ncaa-bracket-absurd-odds-march-madness-dream

Data

The dataset we will be using can be found at:
https://www.kaggle.com/datasets/andrewsundberg/college-basketball-dataset/.

It is data for the division 1 men's college basketball seasons 2013-2023. We will be using the file `cbb.csv` which contains seasons 2013-2019 and seasons 2021-2023. This file was downloaded on Sunday November 12, 2023 and used locally to produce the results found in this report.

```
cbb <- file.path(getwd(), "/cbb.csv")
cbb_csv <- read_csv(cbb)
```

Data Preprocessing

Before we can begin using the data, we first need to do some preprocessing:

1. convert `POSTSEASON` into a `factor`
2. keep only the 68 teams per tournament per season
3. fix invalid column names
4. split the data into training and final holdout set (we will use seasons `2013-2022` to predict season `2023`)

```
postseason_levels <- c("R68", "R64", "R32", "S16", "E8", "F4", "2ND", "Champions")
dat <- cbb_csv |>
  mutate(SEED = as.integer(SEED), # coercion warning
         POSTSEASON = factor(POSTSEASON, levels = postseason_levels)) |>
  filter(!is.na(SEED))
levels(dat$POSTSEASON) <- c("R68", "R64", "R32", "S16", "E8", "F4", "2ND", "1ST")
names(dat) <- make.names(names(dat)) # fix invalid column names
edx <- dat |> filter(YEAR < 2023)
final_holdout_set <- dat |> filter(YEAR == 2023)
```

Data Exploration

We have 18 variables to select our predictors from:

- ADJOE: Adjusted Offensive Efficiency [3]
- ADJDE: Adjusted Defensive Efficiency [4]
- BARTHAG: Power Rating [5]
- EFG_O: Effective Field Goal Percentage Shot
- EFG_D: Effective Field Goal Percentage Allowed
- TOR: Turnover Percentage Allowed [6]
- TORD: Turnover Percentage Committed [7]
- ORB: Offensive Rebound Rate
- DRB: Offensive Rebound Rate Allowed
- FTR : Free Throw Rate [8]
- FTRD: Free Throw Rate Allowed
- X2P_O: Two-Point Shooting Percentage
- X2P_D: Two-Point Shooting Percentage Allowed
- X3P_O: Three-Point Shooting Percentage
- X3P_D: Three-Point Shooting Percentage Allowed
- ADJ_T: Adjusted Tempo [9]
- WAB: Wins Above Bubble [10]
- SEED: Seed in the NCAA March Madness Tournament

To determine which ones to include in our algorithm, let's plot (by season) each variable against each team's postseason level and look for any trends and patterns.

[3] An estimate of the offensive efficiency (points scored per 100 possessions) a team would have against the average Division I defense

[4] An estimate of the defensive efficiency (points allowed per 100 possessions) a team would have against the average Division I offense

[5] Chance of beating an average Division I team

[6] Turnover Rate

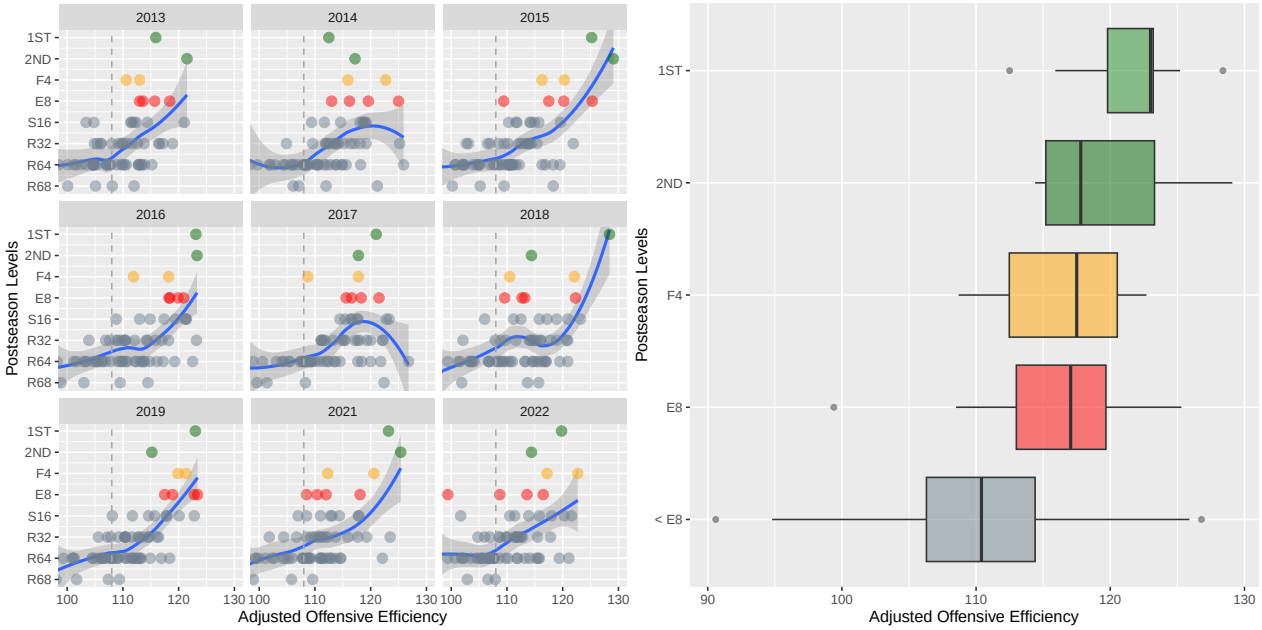
[7] Steal Rate

[8] How often the given team shoots Free Throws

[9] An estimate of the tempo (possessions per 40 minutes) a team would have against the team that wants to play at an average Division I tempo

[10] The bubble refers to the cut off between making the NCAA March Madness Tournament and not making it

ADJOE - Adjusted Offensive Efficiency



The adjusted offensive efficiency (`ADJOE`) is the number of points scored per 100 possessions against the average defense.

For most years, the better the `ADJOE`, the better the postseason level except for years `2014` and `2017`.

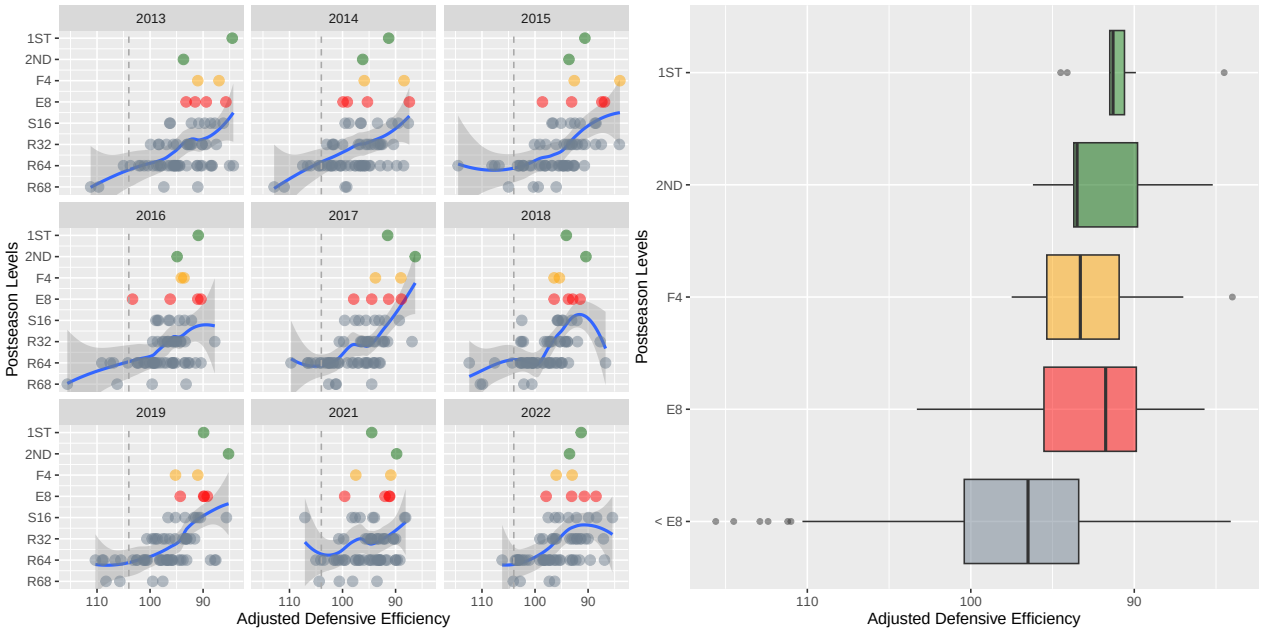
For year `2014`, a few strong `ADJOE` teams lost in the early rounds. And most of the Elite 8 (colored dots) teams only had above average `ADJOE` resulting in a small hump in the blue trend line.

For year `2017`, many strong `ADJOE` teams lost in the early rounds. And many of the Elite 8 teams had similar `ADJOE` resulting in a sharper hump compared to `2014`.

Elite 8 teams had an `ADJOE` of at least `108` (dashed line). There is one outlier below this mark. You can see it in year `2022` on the left plot and left-hand side in `E8` on the right plot.

We will not use `ADJOE` as a predictor. Instead, we will use an `ADJOE` of `108` as a filter.

ADJDE - Adjusted Defensive Efficiency



The adjusted defensive efficiency (`ADJDE`) is the number of points allowed per 100 possessions against the average offense.

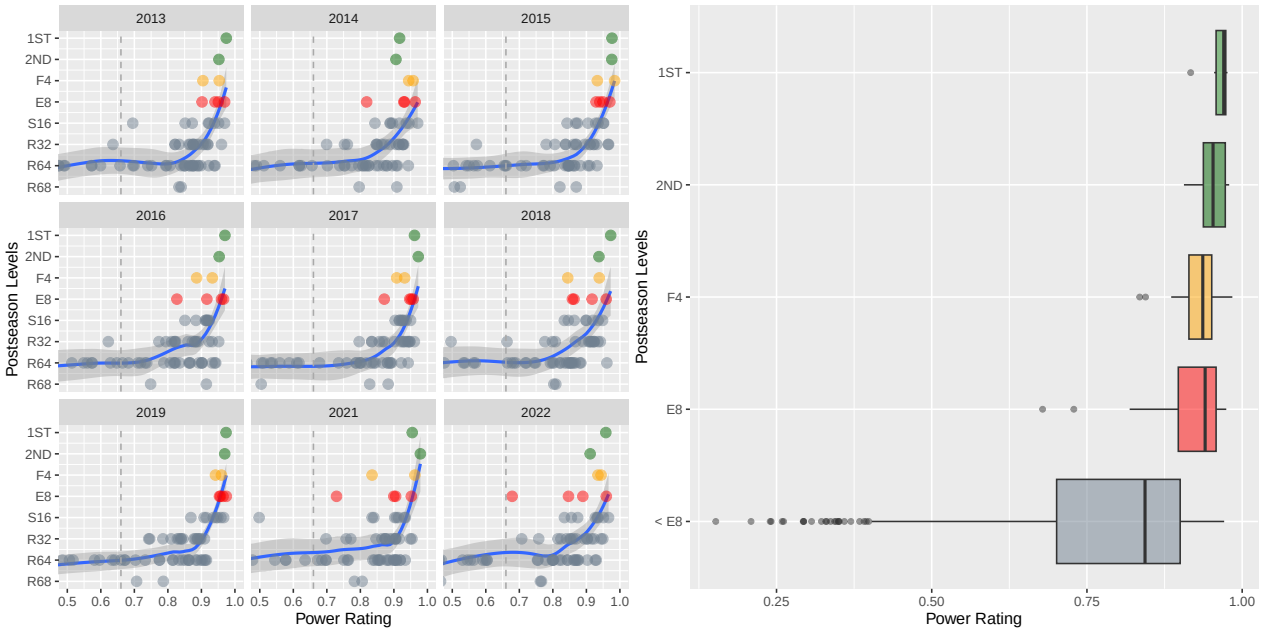
For most years, the better (lower) the `ADJDE`, the better the postseason level except for years `2018` and `2022`.

For year `2018`, a few strong `ADJDE` teams lost in the early rounds. And most of the Elite 8 teams only had above average `ADJDE` resulting in a hump in the blue trend line.

For year `2022`, many strong `ADJDE` teams lost in the early rounds. And most of the Elite 8 teams only had above average `ADJDE` resulting in a softer hump compared to `2018`.

Elite 8 teams had an `ADJDE` of at most `104`. We will not use `ADJDE` as a predictor. Instead, we will use an `ADJDE` of `104` as another filter.

BARTHAG - Power Rating

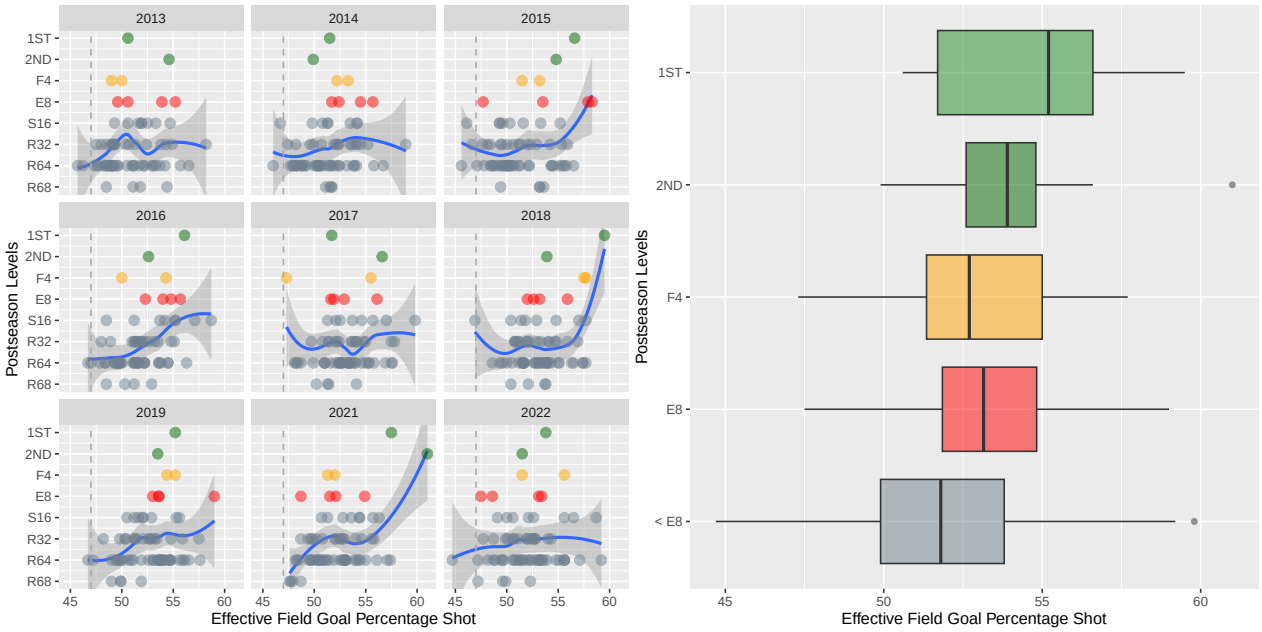


The power rating (`BARTHAG`) is the chance of winning against the average team. It is a stat made popular by Bart Torvik [11] using the Pythagorean expectation formula [12].

As expected, the better the `BARTHAG`, the better the postseason level. This makes it a good predictor.

[11] http://adamcwisports.blogspot.com/p/every-possession-counts.html
[12] https://en.wikipedia.org/wiki/Pythagorean_expectation

EFG_O - Effective Field Goal Percentage Shot

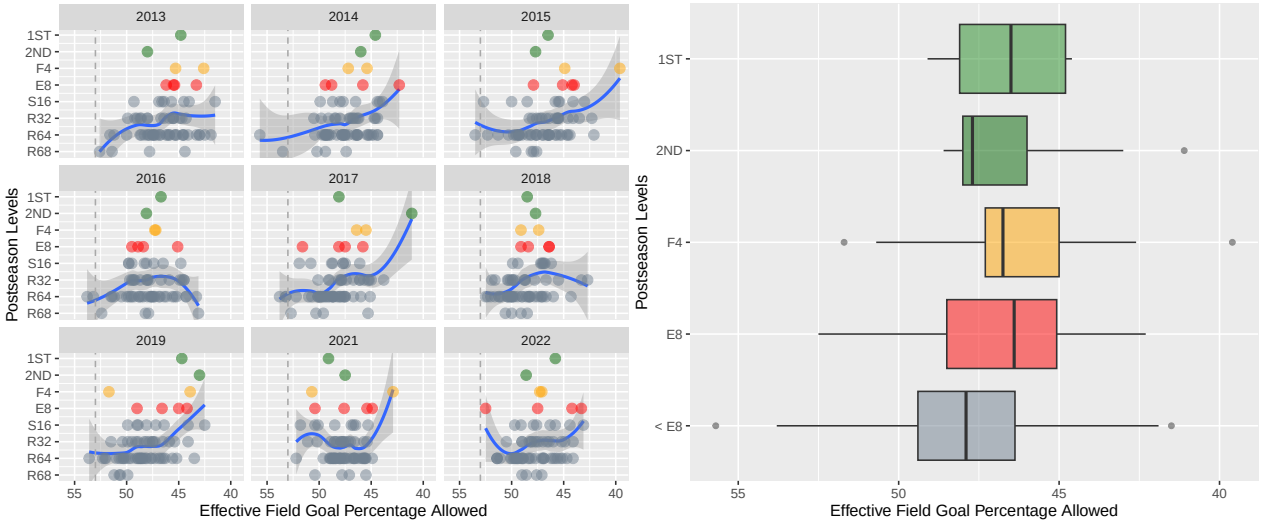


The effective field goal percentage shot (`EFG_O`) is the measurement of how successful a team is from the field. It is calculated by giving the three point shot extra weight:

$$\frac{TwoPointsMade + (1.5 \cdot ThreePointsMade)}{FieldGoalAttempts}$$

We will not use `EFG_O` as a predictor or as a filter.

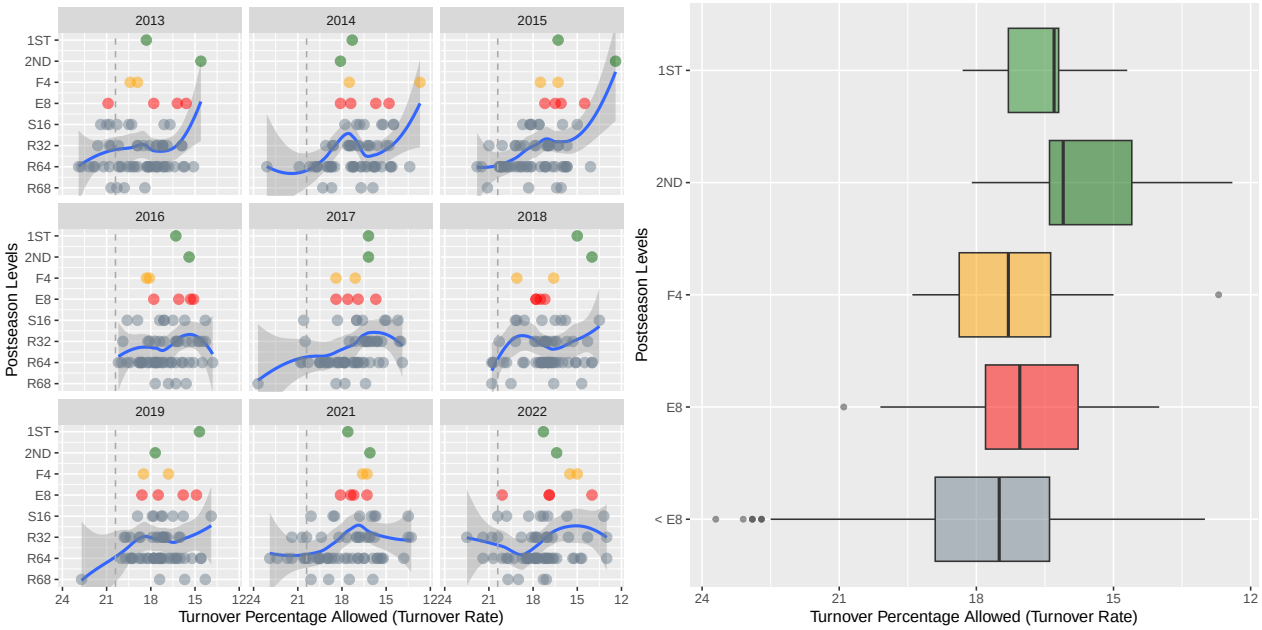
EFG_D - Effective Field Goal Percentage Allowed



The effective field goal percentage allowed (`EFG_D`) is the measurement of how successful a team's opponent is from the field.

We will also not use `EFG_D` as a predictor or as a filter.

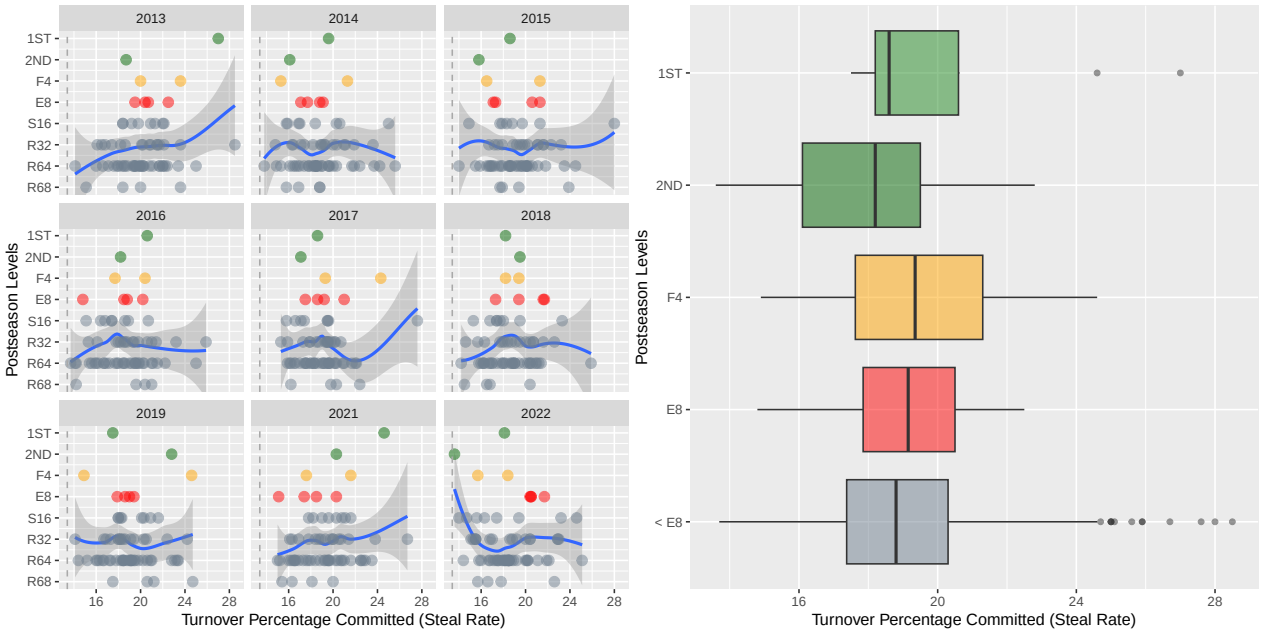
TOR - Turnover Percentage Allowed (Turnover Rate)



The turnover percentage allowed (`TOR`) is the number of turnovers per 100 possessions.

Like the previous two variables, we will not use `TOR` as a predictor or as a filter.

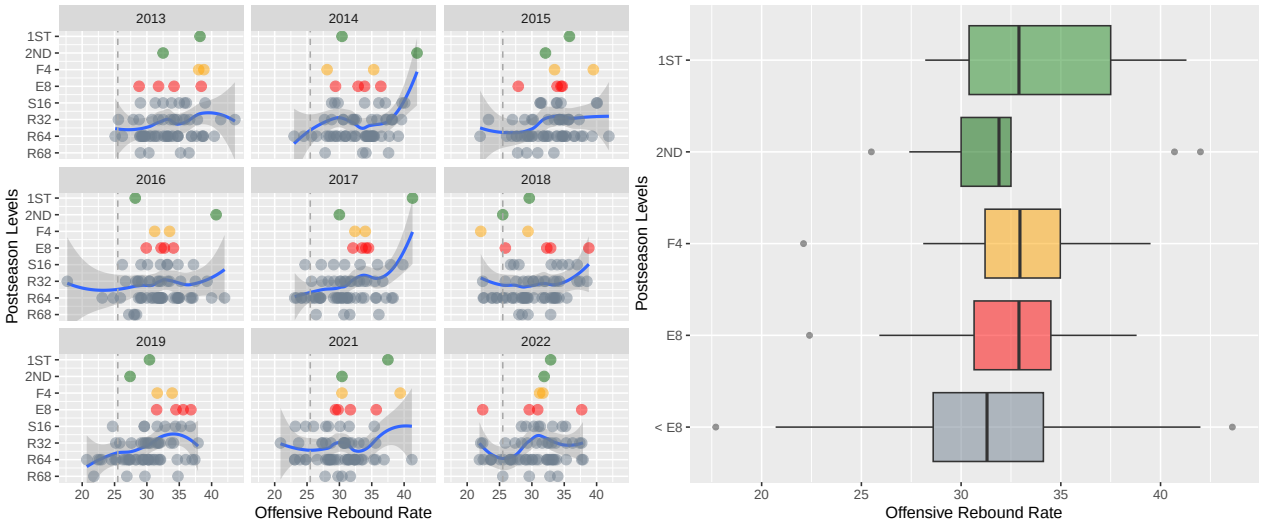
TORD - Turnover Percentage Committed (Steal Rate)



The turnover percentage committed (`TORD`) is the number of turnovers committed by a team's opponent per 100 possessions.

We will also not use `TORD` as a predictor or as a filter.

ORB - Offensive Rebound Rate

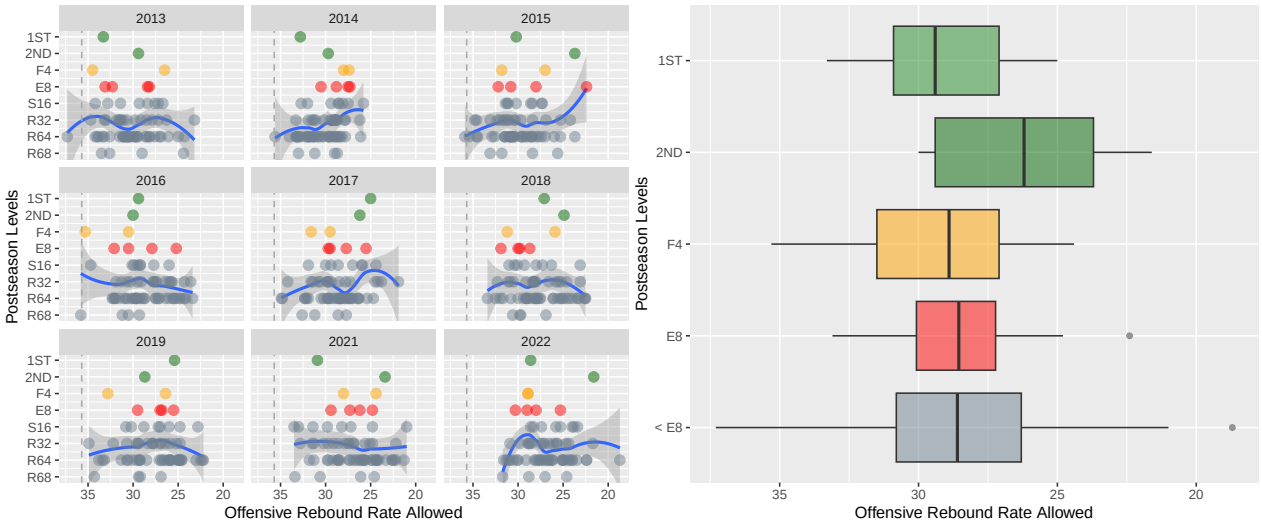


The offensive rebound rate (`ORB`) is the measurement of a team's ability to get offensive rebounds:

$$\frac{Offensive Rebounds}{Offensive Rebounds + Opponent Defensive Rebounds}$$

We will not use `ORB` as a predictor or as a filter.

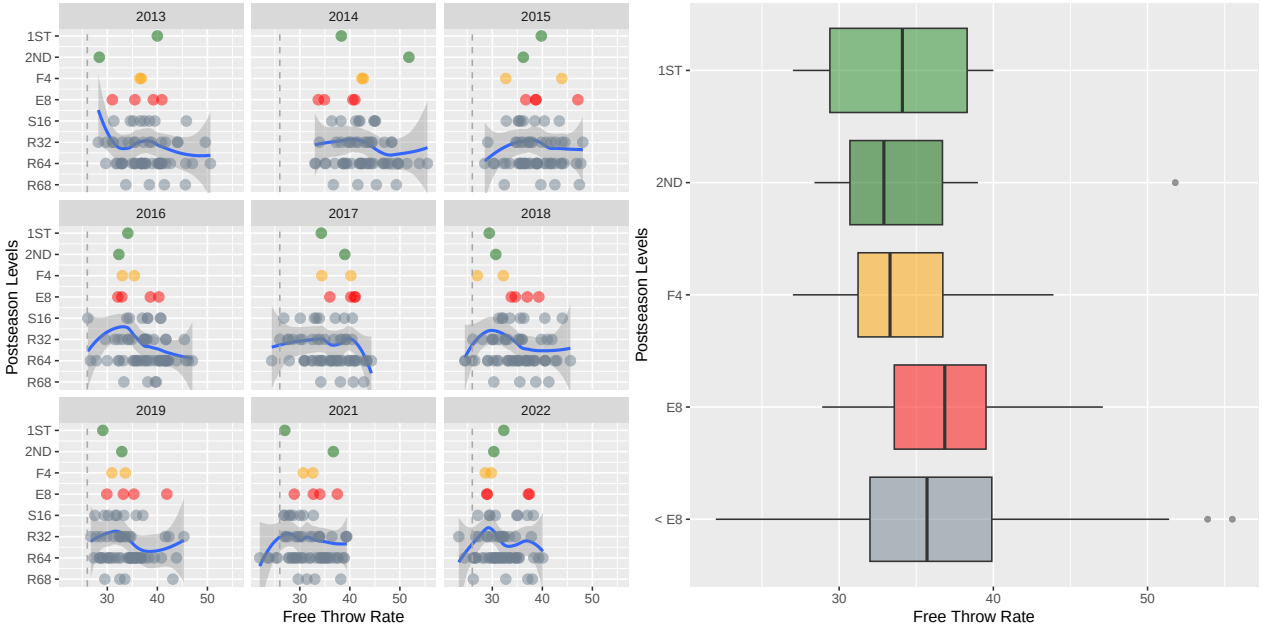
DRB - Offensive Rebound Rate Allowed



The offensive rebound rate allowed (`DRB`) is the measurement of an team's opponent's ability to get offensive rebounds.

We will also not use `DRB` as a predictor or as a filter.

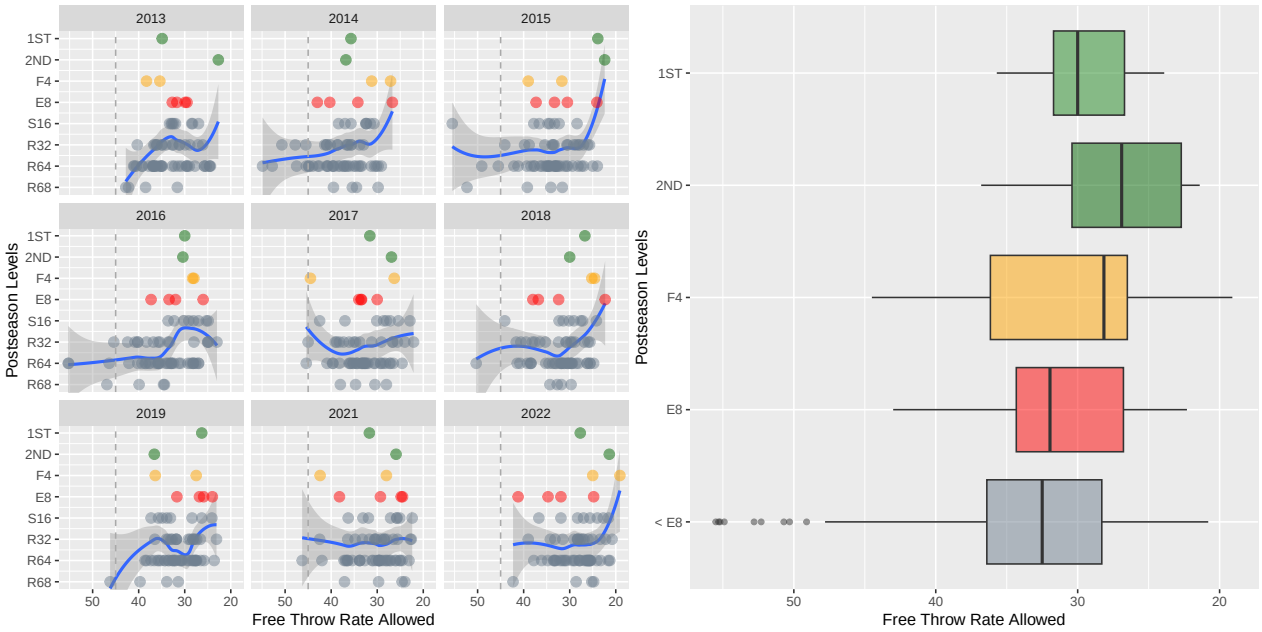
FTR - Free Throw Rate



The free throw rate (`FTR`) is the team's ratio of free throw attempts to field goal attempts.

We will not use `FTR` as a predictor or as a filter.

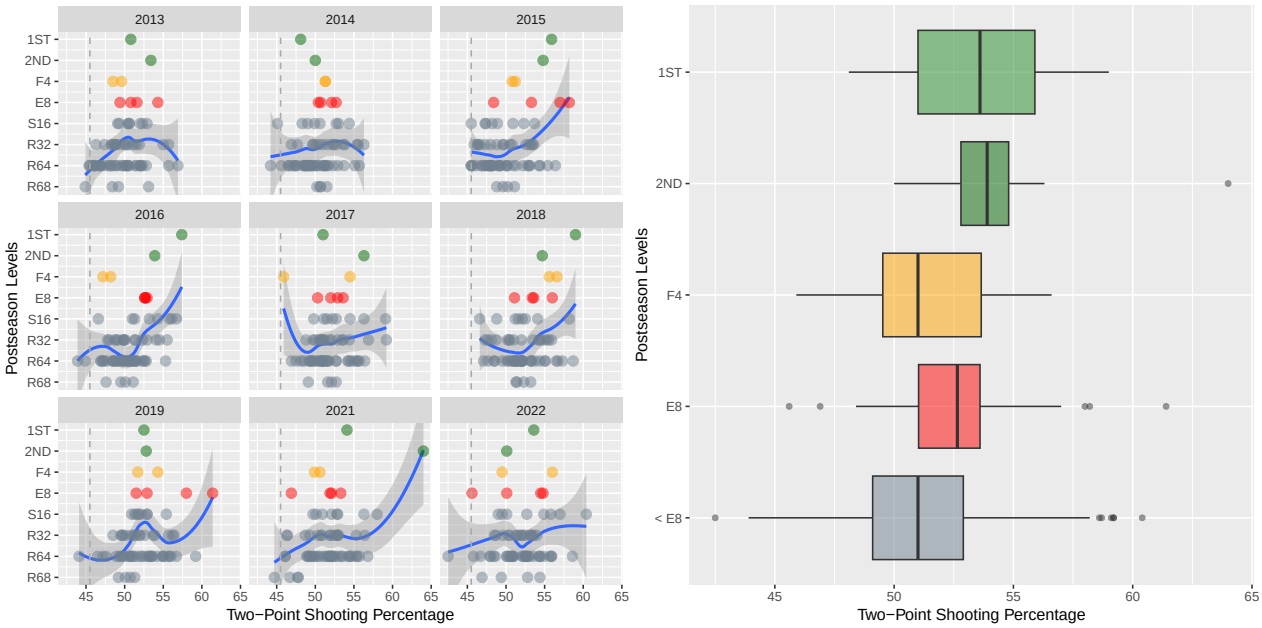
FTRD - Free Throw Rate Allowed



The free throw rate allowed (`FTRD`) is the team's opponent's ratio of free throw attempts to field goal attempts.

We will also not use `FTRD` as a predictor or as a filter.

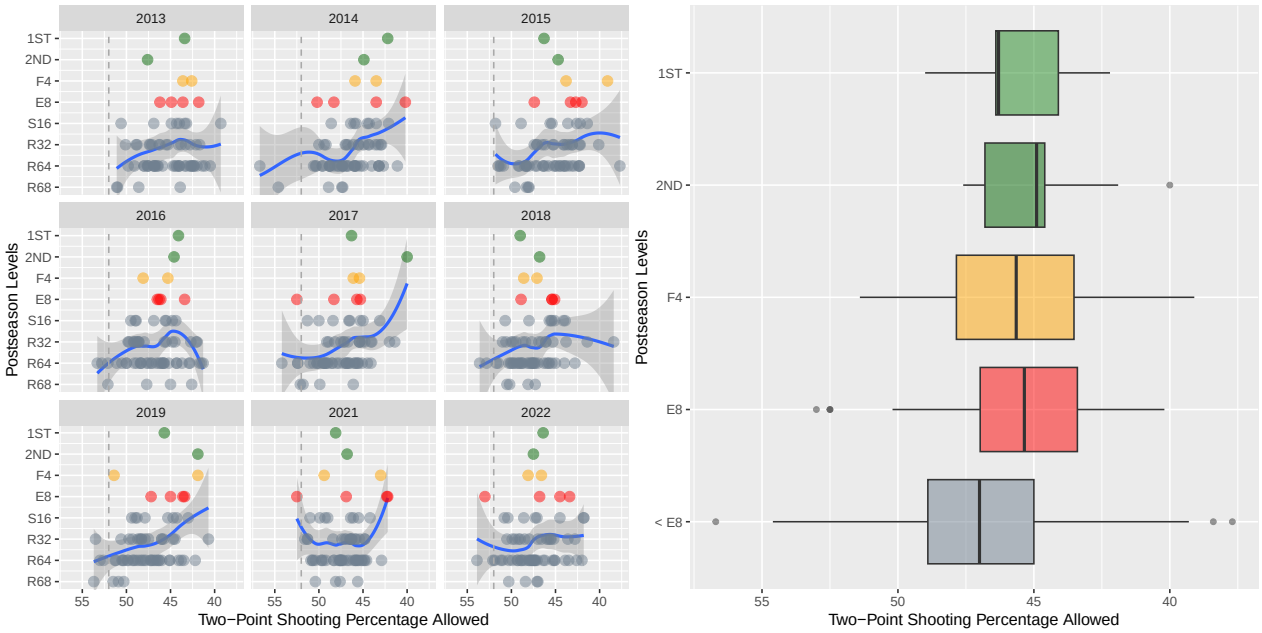
X2P_O - Two-Point Shooting Percentage



The two-point shooting percentage (`X2P_O`) is the team's ratio of two-points made to two-points attempted. It is $\frac{1}{2}$ of the prior composite variable `EFG_O`.

But like `EFG_O`, we will not use `X2P_O` as a predictor or as a filter.

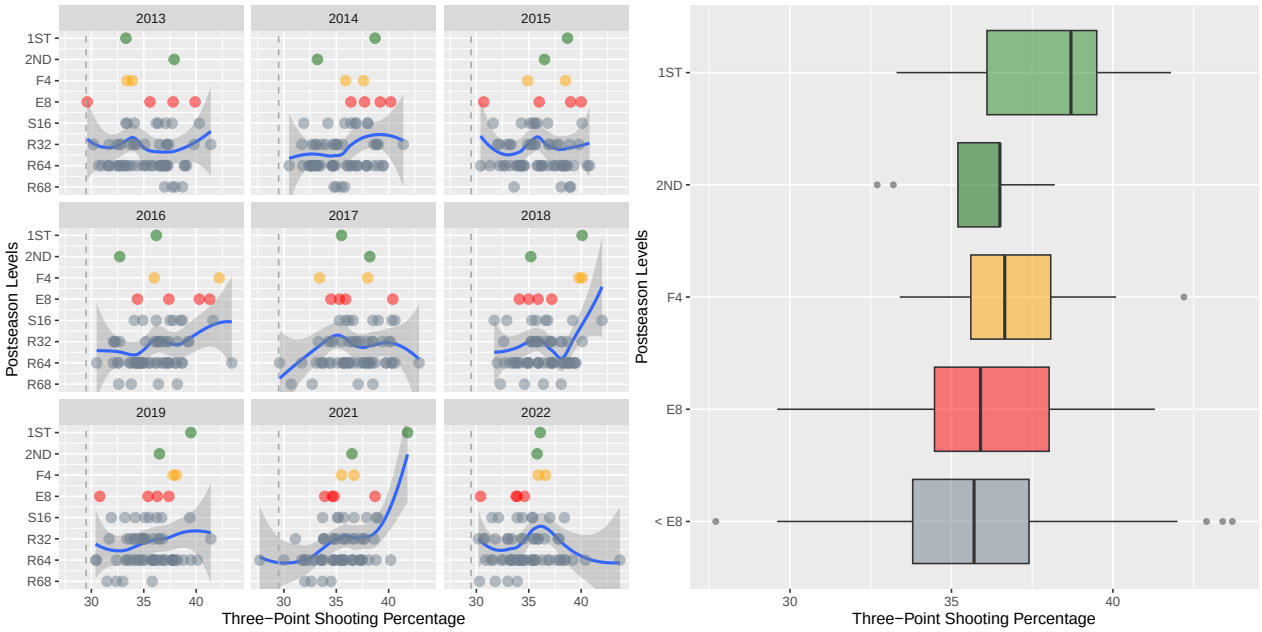
X2P_D - Two-Point Shooting Percentage Allowed



The two-point shooting percentage allowed (`X2P_D`) is the team's opponent's ratio of two-points made to two-points attempted. It is $\frac{1}{2}$ of the prior composite variable `EFG_D`.

But like `EFG_D`, we will not use `X2P_D` as a predictor or as a filter.

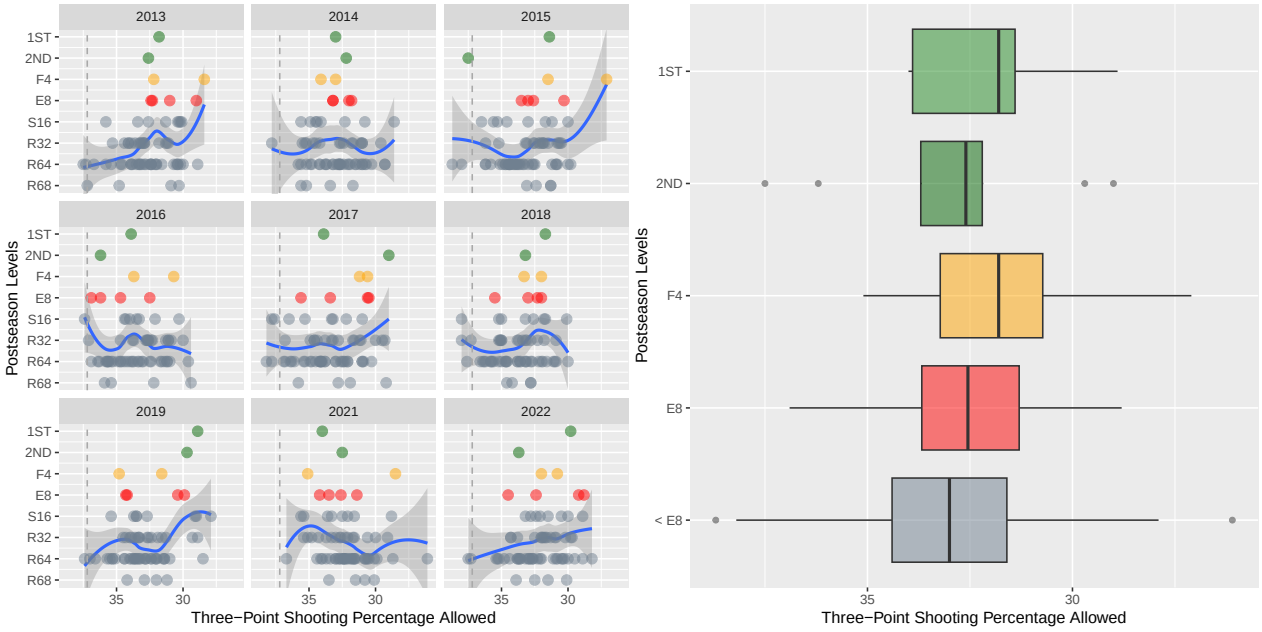
X3P_O - Three-Point Shooting Percentage



The three-point shooting percentage (`X3P_O`) is the team's ratio of three-points made to three-points attempted. It is second-half of the prior composite variable `EFG_O`.

But like `EFG_O`, we will not use `X3P_O` as a predictor or as a filter.

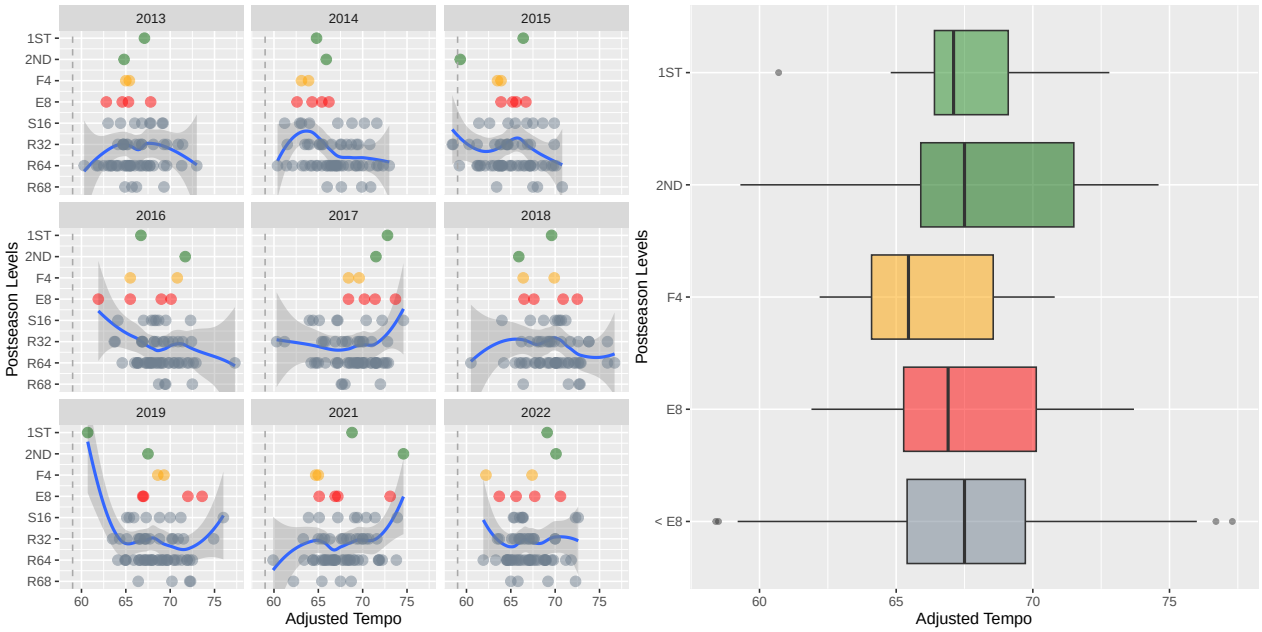
X3P_D - Three-Point Shooting Percentage Allowed



The three-point shooting percentage allowed (`X3P_D`) is the team's opponent's ratio of three-points made to three-points attempted. It is second-half of the prior composite variable `EFG_D`.

But like `EFG_D`, we will not use `X3P_D` as a predictor or as a filter.

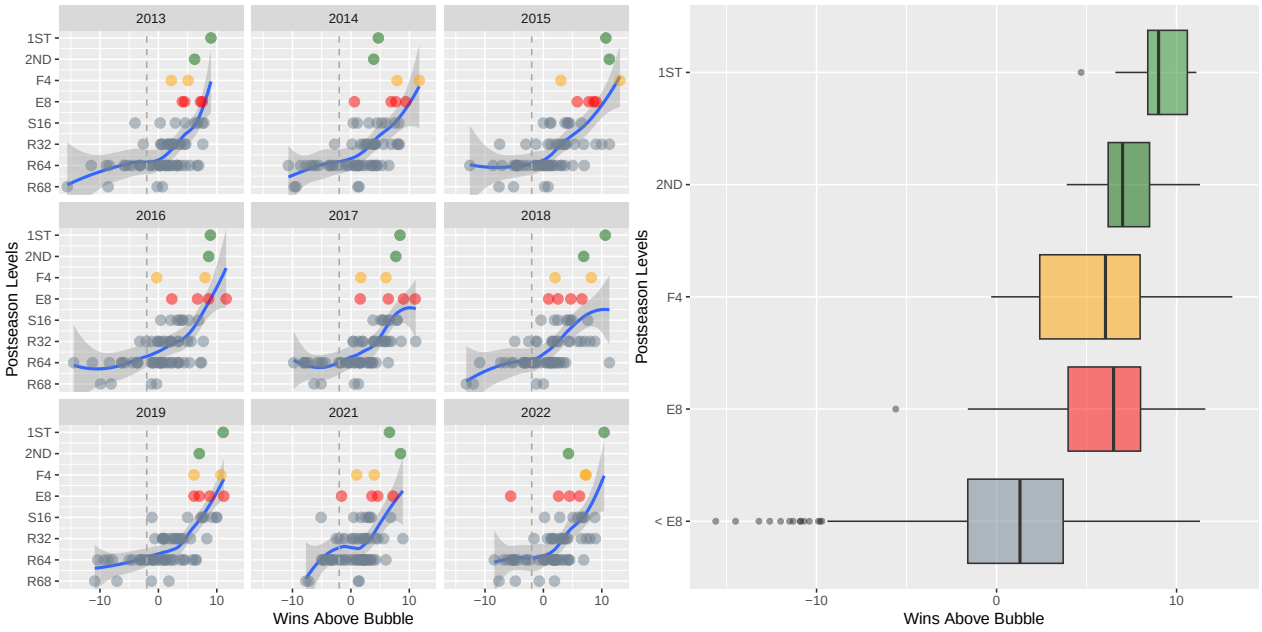
ADJ_T - Adjusted Tempo



The adjusted tempo (`ADJ_T`) is an estimate of a team's tempo (possessions per 40 minutes) against the average defense.

We will not use `ADJ_T` as a predictor or as a filter.

WAB - Wins Above Bubble



The wins above bubble (`WAB`) is the difference in the number of wins a team has compared to the expected number of wins an average bubble team [13] would earn against that team's schedule.

The better the `WAB`, the better the postseason level except the trend line flattens out for years `2017` and `2018`.

For year `2017`, a few strong `WAB` teams lost in the early rounds. And most of the Elite 8 teams only had above average `WAB` resulting in a late flat trend line.

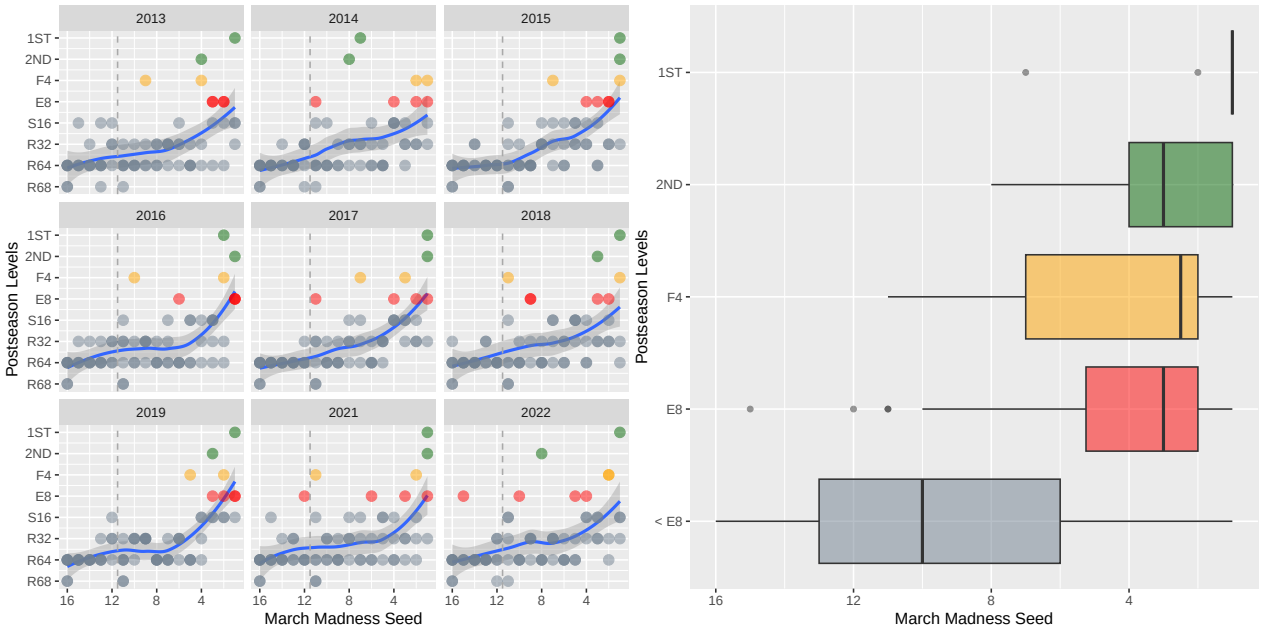
For year `2018`, many strong `WAB` teams lost in the early rounds. And most of the Elite 8 teams had above average `WAB` resulting in an early flat trend line compared to `2017`.

And there is one outlier in year `2022`.

However the pattern is clear that the better the `WAB`, the better the postseason level making it a good predictor.

[13] A bubble team is a team that is on the verge of making the field of 68 for the March Madness, but an invitation isn't guaranteed. At the end of the season, a number of teams are "on the bubble" of getting into the NCAA tournament.

SEED - March Madness Seed



The March Madness seed (`SEED`) is the ranking given to each team as determined by the selection committee.

And as expected, the better the `SEED`, the better the postseason level. This makes it a good predictor.

Final Predictors

After visualizing the data for trends and patterns, we've surprisingly eliminated more than $\frac{2}{3}$ of the eighteen variables we started with.

Here's our 5 finalists:

variable	usage
Adjusted Offensive Efficiency (ADJOE)	filter $>$ 108
Adjusted Defensive Efficiency (ADJDE)	filter $<$ 104
Power Rating (BARTHAG)	predictor
Wins Above Bubble (WAB)	predictor
March Madness Seed (SEED)	predictor

Insights

Looking at the above table, it seems we've removed a lot of variables based on a visual inspection of each variable in isolation. The remaining predictors, on the surface, seem like good predictors. The `BARTHAG` uses a formula that incorporates both `ADJOE` and `ADJDE` along with some "magic" numbers. The `WAB` along with `ADJOE` and `ADJDE` consider the average team in their values. By using these variables, we have, at its heart, the entire NCAA's mean value.

But it's also possible that the "noise" produced from the discarded variables may have just enough effect over that mean. So we will need to consider slowly loosening up our self-imposed constraints to see if the algorithm improves or just gets worse.

Algorithm 1

Our first algorithm will train using the three predictors: `BARTHAG`, `WAB` and `SEED`. We will also remove any teams with "bad" `ADJOE` and `ADJDE` from our test set.

```
# training/test
train_set <- edx |>
  mutate(POSTSEASON = as.numeric(POSTSEASON))
test_set <- final_holdout_set |>
  mutate(POSTSEASON = as.numeric(POSTSEASON)) |>
  filter(ADJOE > 108 & ADJDE < 104) # filter

# train with 3 predictors
fit_glm <- train(POSTSEASON ~ BARTHAG + WAB + SEED, method = "glm", data = train_set)
fit_gam <- train(POSTSEASON ~ BARTHAG + WAB + SEED, method = "gamLoess", data = train_set)
fit_knn <- train(POSTSEASON ~ BARTHAG + WAB + SEED, method = "knn", data = train_set)
fit_rf  <- train(POSTSEASON ~ BARTHAG + WAB + SEED, method = "rf", data = train_set, tuneGrid = data.frame(mtry = 2))

# predict
pred_glm <- predict(fit_glm, test_set)
pred_gam <- predict(fit_gam, test_set)
pred_knn <- predict(fit_knn, test_set)
pred_rf  <- predict(fit_rf,  test_set)

# test
test_set <- test_set |>
  mutate(glm = pred_glm,
         gam = pred_gam,
         knn = pred_knn,
         rf  = pred_rf,
         ensemble = (glm + gam + knn + rf) / 4)
elite_glm <- test_set |> arrange(desc(glm)) |> head(8)
elite_gam <- test_set |> arrange(desc(gam)) |> head(8)
elite_knn <- test_set |> arrange(desc(knn)) |> head(8)
elite_rf  <- test_set |> arrange(desc(rf))  |> head(8)
elite_ensemble <- test_set |> arrange(desc(ensemble)) |> head(8)
```

Score Function

To see how well our algorithm performs, we need to evaluate it against a typical bracket scoring system. We will use the scoring function below:

```
score_for_elite <- function(elite_pred) {
  first_round <- elite_pred |> filter(POSTSEASON >= 3) |> nrow()
  second_round <- elite_pred |> filter(POSTSEASON >= 4) |> nrow()
  sweet_sixteen <- elite_pred |> filter(POSTSEASON >= 5) |> nrow()
  elite_eight <- elite_pred |> head(4) |> filter(POSTSEASON >= 6) |> nrow()
  final_four <- elite_pred |> head(2) |> filter(POSTSEASON >= 7) |> nrow()
  champion <- elite_pred[1,] |> filter(POSTSEASON == 8) |> nrow()
  first_round + second_round * 2 + sweet_sixteen * 4 + elite_eight * 8 + final_four * 16 + champion * 32
}
```

Results

method	score
glm	18
gam	32
knn	11
rf	23
ensemble	24

The results aren't great. Let's try removing the filter to see if we can improve the scores.

Algorithm 2

Our second algorithm will use just the three predictors: `BARTHAG`, `WAB` and `SEED`.

```
# test
test_set <- final_holdout_set |>
  mutate(POSTSEASON = as.numeric(POSTSEASON))

# predict
pred_glm <- predict(fit_glm, test_set)
pred_gam <- predict(fit_gam, test_set)
pred_knn <- predict(fit_knn, test_set)
pred_rf  <- predict(fit_rf,  test_set)

# test
test_set <- test_set |>
  mutate(glm = pred_glm,
         gam = pred_gam,
         knn = pred_knn,
         rf  = pred_rf,
         ensemble = (glm + gam + knn + rf) / 4)
elite_glm <- test_set |> arrange(desc(glm)) |> head(8)
elite_gam <- test_set |> arrange(desc(gam)) |> head(8)
elite_knn <- test_set |> arrange(desc(knn)) |> head(8)
elite_rf  <- test_set |> arrange(desc(rf))  |> head(8)
elite_ensemble <- test_set |> arrange(desc(ensemble)) |> head(8)
```

Results

method	score
glm	18
gam	32
knn	11
rf	23
ensemble	24

How disappointing! The scores are exactly the same as algorithm one. The filter had no effect on the test set even though it had previously removed 18 teams.

Algorithm 3

For our third algorithm, we are going to test the idea that all the variables have some effect on the NCAA mean value and that we shouldn't have excluded some because of the "noise" seen in the plots.

First, let's explore the training set using cross-validation. We will need a function to help split up the data so that the test set is a single season and the training is done on the remaining seasons:

```
# create a test index for training/test sets
test_index_for_year <- function(year) {
  edx |>
    mutate(index = as.numeric(rownames(edx))) |>
    filter(YEAR == year) |>
    select(index) |>
    pull(index)
}
```

Once we have the cross-validation datasets, we can start computing the scores like we did before. To keep things manageable, I used a helper function for handling the training and prediction:

```
# prediction helper function
pred_for_elite_eight <- function(train_set, test_set, method, test_index) {
  train_set <- train_set |>
    select(-TEAM, -CONF, -G, -W, -YEAR) |>
    mutate(POSTSEASON = as.numeric(POSTSEASON))

  fit <- train(POSTSEASON ~ ., method = method, data = train_set)
  y_hat <- predict(fit, test_set)
  edx[test_index,] |>
    mutate(POSTSEASON = as.numeric(POSTSEASON),
           y = y_hat) |>
    arrange(desc(y)) |>
    select(TEAM, POSTSEASON, y) |>
    head(8)
}
```

Then we just supply the necessary starting arguments and let a rip!

```
# compute all scores by year (took ~14 min)
scores <- sapply(unique(edx$YEAR), function(year) {
  test_index <- test_index_for_year(year)
  test_set <- edx[test_index,]
  train_set <- edx[-test_index,]

  # glm
  elite_glm <- pred_for_elite_eight(train_set, test_set, "glm", test_index)
  score_glm <- score_for_elite(elite_glm)
  # gam
  elite_gam <- pred_for_elite_eight(train_set, test_set, "gamLoess", test_index)
  score_gam <- score_for_elite(elite_gam)
  # knn
  elite_knn <- pred_for_elite_eight(train_set, test_set, "knn", test_index)
  score_knn <- score_for_elite(elite_knn)
  # random forest
  elite_rf <- pred_for_elite_eight(train_set, test_set, "rf", test_index)
  score_rf <- score_for_elite(elite_rf)

  c(year, score_glm, score_gam, score_knn, score_rf)
})
```

Cross-Validation Results

season	glm	gam	knn	rf	avg
2013	106	110	59	50	81.25
2014	53	45	44	43	46.25
2015	84	118	72	116	97.50
2016	100	100	101	100	100.25
2017	44	50	49	56	49.75
2018	91	97	51	83	80.50
2019	72	68	68	72	70.00
2021	81	65	41	65	63.00
2022	44	92	61	63	65.00
avg	75	83	61	72	72.75

First thing we notice are the higher numbers compared to the previous algorithms! Being able to train on more predictors really does make a huge difference.

The second thing we notice is on average our base score performs better than the user average score and we sometimes beat the seed-based average score (we could probably consistently beat it if we pick enough first and second round games!)

Let's see how well it does on our final holdout set:

```
train_set <- edx |>
  select(-TEAM, -CONF, -G, -W, -YEAR) |>
  mutate(POSTSEASON = as.numeric(POSTSEASON))
test_set <- final_holdout_set |>
  mutate(POSTSEASON = as.numeric(POSTSEASON))

# train with all 18 predictors (took ~2min)
fit_glm <- train(POSTSEASON ~ ., method = "glm", data = train_set)
fit_gam <- train(POSTSEASON ~ ., method = "gamLoess", data = train_set)
fit_knn <- train(POSTSEASON ~ ., method = "knn", data = train_set)
fit_rf <- train(POSTSEASON ~ ., method = "rf", data = train_set)

# predict
pred_glm <- predict(fit_glm, test_set)
pred_gam <- predict(fit_gam, test_set)
pred_knn <- predict(fit_knn, test_set)
pred_rf <- predict(fit_rf, test_set)

# test
test_set <- test_set |>
  mutate(glm = pred_glm,
         gam = pred_gam,
         knn = pred_knn,
         rf = pred_rf,
         ensemble = (glm + gam + knn + rf) / 4)
elite_glm <- test_set |> arrange(desc(glm)) |> head(8)
elite_gam <- test_set |> arrange(desc(gam)) |> head(8)
elite_knn <- test_set |> arrange(desc(knn)) |> head(8)
elite_rf <- test_set |> arrange(desc(rf)) |> head(8)
elite_ensemble <- test_set |> arrange(desc(ensemble)) |> head(8)
```

Final Results

method	score
glm	87
gam	82
knn	55
rf	32
ensemble	80

Yes! These scores are much better! And picking enough correct first and second round games should net us a higher score than the seed-based average.

Conclusion

One of the most interesting applications of machine learning is to find patterns and trends in a large dataset that a normal human couldn't find. Being able to use it as a tool gives one "superhuman" powers.

When the idea of using data analysis for baseball statistics (aka sabermetrics) took off, it changed how the game was both played and watched. Many people collect baseball cards and even more collect baseball statistics. Ask hardcore fans about batting averages and pitcher ERAs and don't be surprised with all the numbers they come up with.

One of my favorite sporting events is the NCAA March Madness. I've participated in bracket tournaments since as long as I can remember. I've tried picking upsets. I've tried finding my Cinderellas. I've picked my Final Four favorites. I've never won the pool but I've always done a little better than the average. I picked this dataset because I wanted to see if I could take a machine learning algorithmic approach to see how it would perform. (Will I use my new found knowledge this March? :thinking_face: Hmm. . .)

Future Work

Where do we go from here? I think getting an even larger dataset from https://www.kaggle.com/ would be the next step. Finding patterns by games, looking for emerging trends from upsets, compiling the data for all Cinderellas. By adding these to the existing algorithms, we will continue to get better predictive power and watch basketball transform as much as baseball has.

References

The dataset used for this report can be downloaded from: https://www.kaggle.com/datasets/andrewsundberg/college-basketball-dataset/

I obtained a lot of basketball tidbits from both https://en.wikipedia.org/wiki/Main_Page and https://www.ncaa.com/march-madness-live/watch.

I hope this research has inspired you to take on the challenge and fill out your own bracket! See you in March!