A Noble Attempt at Using Sabermetrics for D3 College Baseball

Ben Steves

6/29/2021

## Premise

There is nothing more niche than going about a data analysis using Division 3 baseball stats. This is a hill I am happy to die on. Division 3 baseball is not popular - not even in the slighest, at least on the global scale. Major league baseball scouts primarily look at D1 and D2 athletes when trying to evaluate talent. Not that D3 is bad - but the skill level is lesser than D1. MLB teams draft maybe 2 or 3 D3 players a year, sometimes more, sometimes none. Considering every team has 20 picks \* 30 teams + 20 possible compensatory picks gives you about 620 total picks, give or take a few. Three out of 620 is 0.4%. Scouts don’t really scout D3 baseball players unless you hit 20 homers or throw 96, which is D1 level play, usually.

I aimed to try and bring sabermetrics used by Major League Baseball (MLB) teams to the college level, specifically to the teams in the MIAA. Sabermetrics are advanced statistics that give information on how a player is performing. These stats, however, are more detailed and technical than typical stats like batting average, and require heftier calculations. For this project, I calculate a statistic called OFF that uses sabermetrics to calculate the total offensive production of certain players in the MIAA, including Calvin’s team.

## Functions

I’ve documented the functions I made in the actual code. The first three are functions I made myself so that I could perform them on multiple datasets at once, and they are mainly for data cleaning. The rest are all sabermetrics. All formulas come from <https://www.fangraphs.com/> which is the go-to website for sabermetrics data and info.

# remove NA values  
removeNA <-function(df) {  
 df <- df %>%  
 mutate(across(everything(), .fns = ~replace\_na(.,0)))   
}  
  
# choose any columns you want and make integer (for multiple columns at once)  
makeStatsIntegers <- function(df, startCol, endCol) {  
 df <- df %>%  
 mutate\_at(c(startCol:endCol), ~ as.numeric(.)); df  
}   
  
# cleans up variable names into format R can use  
cleanWithJanitor <- function(df) {  
 df <- df %>%  
 janitor::clean\_names(); df  
}  
  
# Calculates Weighted On Base Average (wOBA), which is like Batting Average except it weights certain hits differently  
wOBA <- function(bb, hbp, x1b, x2b, x3b, hr, ab, sf) {  
 numerator = (0.69\*bb) + (0.719\*hbp) + (0.87\*x1b) + (1.217\*x2b) + (1.529\*x3b) + (1.94\*hr)  
 denominator = ab + hbp + bb + sf  
 numerator/denominator  
}  
  
# Batting Average on Balls in Play (BABIP) - batting average on all non-strikeouts and home runs  
BABIP <- function(h, hr, ab, k, sf) {  
 numerator = (h - hr)  
 denominator = (ab - k - hr + sf)  
 numerator/denominator  
}  
  
# wRAA - How many runs a player contributes to a team above average  
wRAA <- function(wOBA, lgWOBA, pa) {  
 ((wOBA - lgWOBA)/1.1) \* pa  
}  
  
# Calculates if certain parks are more hitter friendly or pitcher friendly.   
ParkFactor <- function(g\_home, g\_away, rf\_home, rf\_away, ra\_home, ra\_away) {  
 home = (rf\_home + ra\_home) / g\_home  
 away = (rf\_away + ra\_away) / g\_away  
 result = home / away  
 result  
}  
  
# Weighted Runs Created - very similar to wRAA  
wRC <- function(wOBA, lgwOBA, pa, lgR, lgPA) {  
 (((wOBA - lgwOBA)/1.1) + (lgR/lgPA)) \* pa  
}  
  
# Part of OFF formula, which is batting\_runs + base\_running\_runs  
BattingRuns <- function(wRAA, lgR, lgPA, pf, pa, lgwRC, lgPAh) {  
 wRAA + (lgR/lgPA - (pf\*lgR/lgPA))\*pa + (lgR/lgPA - (lgwRC/lgPAh))\*pa  
}  
  
# Used in wSB, just the average runs in a year / average outs in a year  
RunsPerOut <- function(lgR, lgO) {  
 lgR / lgO  
}  
  
# Value used in wSB  
runCS <- function(rpo) {  
 -1 \* (2 \* rpo + 0.075)  
}  
  
# League average Weighted Stolen Base Runs (wSB), which contains   
# league averages of stolen base attempts, times caught stealing, etc  
lgwSB <- function(lgSB, lgCS, runCS, lg1b, lgBB, lgHBP) {  
 ((lgSB \* 0.2) + (lgCS \* runCS)) / (lg1b + lgBB + lgHBP)  
}  
  
# Part of off formula. Other elements of base running runs were not available as data,   
# so I instead calculated my own version of OFF which takes out those values.   
wSB <- function(sb, cs, runCS, lgwSB, x1b, bb, hbp) {  
 (sb \* 0.2) + (cs \* runCS) - (lgwSB \* (x1b + bb + hbp))  
}

## Cleaning, Transforming, Calculating

### Offensive stats

Scraping web data to get data for 8 MIAA teams 2021 statistics. Schools include Calvin University, Alma College, Albion College, Adrian College, Hope College, Trine University, Kalamazoo College, and Olivet College.

url1 <- 'https://calvinknights.com/sports/bsb/2020-21/teams/calvin?view=lineup&r=0&pos='  
content1 <- read\_html(url1)  
calvinDF <- html\_table(content1, header = TRUE)[[4]] %>% as.data.frame() %>%   
 mutate(team = "calvin")  
  
url2 <- 'https://athletics.hope.edu/sports/bsb/2020-21/teams/hope?view=lineup&r=0&pos='  
content2 <- read\_html(url2)  
hopeDF <- html\_table(content2, header = TRUE)[[4]] %>% as.data.frame() %>%  
 mutate(team = "hope")  
  
url3 <- 'https://www.miaa.org/sports/bsb/2020-21/teams/adrian?view=lineup&r=0&pos='  
content3 <- read\_html(url3)  
adrianDF <- html\_table(content3, header = TRUE)[[4]] %>% as.data.frame() %>%  
 mutate(team = "adrian")  
  
url4 <- 'http://almascots.com/sports/bsb/2020-21/teams/alma?view=lineup&r=0&pos='  
content4 <- read\_html(url4)  
almaDF <- html\_table(content4, header = TRUE)[[4]] %>% as.data.frame() %>%  
 mutate(team = "alma")  
  
url5 <- 'https://www.trinethunder.com/sports/bsb/2020-21/teams/trine?view=lineup&r=0&pos='  
content5 <- read\_html(url5)  
trineDF <- html\_table(content5, header = TRUE)[[5]] %>% as.data.frame() %>%  
 mutate(team = "trine")  
  
url6 <- 'https://www.miaa.org/sports/bsb/2020-21/teams/albion?view=lineup&r=0&pos='  
content6 <- read\_html(url6)  
albionDF <- html\_table(content6, header = TRUE)[[4]] %>% as.data.frame() %>%  
 mutate(team = "albion")  
  
url7 <- 'https://hornets.kzoo.edu/sports/bsb/2020-21/teams/kalamazoo?view=profile&r=0&pos='  
content7 <- read\_html(url7)  
kalamazooDF <- html\_table(content7, header = TRUE)[[16]] %>% as.data.frame() %>%  
 mutate(team = "kalamazoo")  
  
url8 <- 'https://www.olivetcomets.com/sports/bsb/2020-21/teams/olivet?view=lineup&r=0&pos='  
content8 <- read\_html(url8)  
olivetDF <- html\_table(content8, header = TRUE)[[4]] %>% as.data.frame() %>%  
 mutate(team = "olivet")

teams <- list(adrianDF=adrianDF, trineDF=trineDF,  
 calvinDF=calvinDF, almaDF=almaDF,  
 hopeDF=hopeDF, albionDF=albionDF,  
 olivetDF=olivetDF, kalamazooDF=kalamazooDF)

# data cleaning  
system.time(teams <- lapply(teams, cleanWithJanitor))

## user system elapsed   
## 0.170 0.014 0.220

system.time(teams <- lapply(teams, makeStatsIntegers, 5, 19))

## user system elapsed   
## 0.145 0.002 0.445

system.time(teams <- lapply(teams, removeNA))

## user system elapsed   
## 0.052 0.000 0.054

Parallel processing try, to compare for fun

numCores = detectCores(); numCores

## [1] 4

system.time(teams <- mclapply(teams, cleanWithJanitor, mc.cores = numCores))

## user system elapsed   
## 0.181 0.067 0.163

system.time(teams <- mclapply(teams, makeStatsIntegers, 5, 19, mc.cores = numCores))

## user system elapsed   
## 0.096 0.082 0.064

system.time(teams <- mclapply(teams, removeNA, mc.cores = numCores))

## user system elapsed   
## 0.095 0.088 0.063

Would have to think using parallelism is more beneficial with more data - not that many records to deal with here.

# data cleaning  
all\_teams <- do.call(bind\_rows, teams) %>%  
 filter(ab < 200) %>%   
 mutate(ops = obp + slg,  
 year\_num = case\_when(  
 yr == "Fr." ~ 1, yr == "Fr" ~ 1, yr == "Freshman" ~ 1,  
 yr == "So." ~ 2, yr == "So" ~ 2, yr == "Sophomore" ~ 2,  
 yr == "Jr." ~ 3, yr == "Jr" ~ 3, yr == "Junior" ~ 3,  
 yr == "Sr." ~ 4, yr == "Sr" ~ 4, yr == "Senior" ~ 4,  
 yr == "Sr.-5th" ~ 5, yr == "GS" ~ 5  
 ) %>% as.factor())

### Extended offensive stats

url9 <- 'https://calvinknights.com/sports/bsb/2020-21/teams/calvin?view=lineup&r=0&pos='  
content9 <- read\_html(url9)  
calvinDFb <- html\_table(content9, header = TRUE)[[5]] %>% as.data.frame() %>%   
 mutate(team = "calvin")  
  
url10 <- 'https://athletics.hope.edu/sports/bsb/2020-21/teams/hope?view=lineup&r=0&pos='  
content10 <- read\_html(url10)  
hopeDFb <- html\_table(content10, header = TRUE)[[5]] %>% as.data.frame() %>%  
 mutate(team = "hope")  
  
url11 <- 'https://www.miaa.org/sports/bsb/2020-21/teams/adrian?view=lineup&r=0&pos='  
content11 <- read\_html(url11)  
adrianDFb <- html\_table(content11, header = TRUE)[[5]] %>% as.data.frame() %>%  
 mutate(team = "adrian")  
  
url12 <- 'http://almascots.com/sports/bsb/2020-21/teams/alma?view=lineup&r=0&pos='  
content12 <- read\_html(url12)  
almaDFb <- html\_table(content12, header = TRUE)[[5]] %>% as.data.frame() %>%  
 mutate(team = "alma")  
  
url13 <- 'https://www.trinethunder.com/sports/bsb/2020-21/teams/trine?view=lineup&r=0&pos='  
content13 <- read\_html(url13)  
trineDFb <- html\_table(content13, header = TRUE)[[6]] %>% as.data.frame() %>%  
 mutate(team = "trine")  
  
url14 <- 'https://www.miaa.org/sports/bsb/2020-21/teams/albion?view=lineup&r=0&pos='  
content14 <- read\_html(url14)  
albionDFb <- html\_table(content14, header = TRUE)[[5]] %>% as.data.frame() %>%  
 mutate(team = "albion")  
  
url15 <- 'https://hornets.kzoo.edu/sports/bsb/2020-21/teams/kalamazoo?view=profile&r=0&pos='  
content15 <- read\_html(url15)  
kalamazooDFb <- html\_table(content15, header = TRUE)[[18]] %>% as.data.frame() %>%  
 mutate(team = "kalamazoo")  
  
url16 <- 'https://www.olivetcomets.com/sports/bsb/2020-21/teams/olivet?view=lineup&r=0&pos='  
content16 <- read\_html(url16)  
olivetDFb <- html\_table(content16, header = TRUE)[[5]] %>% as.data.frame() %>%  
 mutate(team = "olivet")

teams\_exH <- list(adrianDFb=adrianDFb, trineDFb=trineDFb,  
 calvinDFb=calvinDFb, almaDFb=almaDFb,  
 hopeDFb=hopeDFb, albionDFb=albionDFb,  
 olivetDFb=olivetDFb, kalamazooDFb=kalamazooDFb)

#data cleaning  
teams\_exH <- lapply(teams\_exH, cleanWithJanitor)  
teams\_exH <- lapply(teams\_exH, makeStatsIntegers, 5, 15)  
teams\_exH <- lapply(teams\_exH, removeNA)

#data cleaning  
all\_teams\_exH <- do.call(bind\_rows, teams\_exH) %>%  
 filter(tb < 120) %>%  
 select(1:2, 6:15)  
  
all\_teams\_hitting <- all\_teams %>%  
 left\_join(all\_teams\_exH, by = "name", copy = TRUE)

### Park Factor data

Park Factor is weird but it is a part of the Batting Runs parameter of OFF. Certain ballparks are more hitter friendly than others (maybe they have smaller fences, higher altitudes, etc). I gathered this data from each individual website on the “Split Stats” tab of their statistics pages. For example, Calvin’s is here: <https://calvinknights.com/sports/bsb/2020-21/teams/calvin?view=splits&pos=h> . Park Factors over 1 indicate a more hitter-friendly park.

school <- c("albion", "kalamazoo" ,"calvin", "adrian", "trine", "hope", "alma", "olivet")  
park <- c("Frank Joranko Field at Champions Stadium", "Woodworth Field" ,   
 "Baseball Field (Calvin)", "Nicolay Field", "Jannen Field",   
 "Boeve Stadium", "Klenk Park", "Baseball Field (Olivet)")  
  
fields <- data.frame(school, park)

teams\_ss <- read.csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vSP2PP-tu0Tqv\_eieBRlNRc7w4FVuusEd6Aw9EaGbQAr\_E4n9lqjdPsiip-COwA8TpGwuZ4tnJ9c9b5/pub?output=csv")  
  
fields <- fields %>%  
 left\_join(teams\_ss, by = "school", copy = TRUE)   
  
fields <- fields %>%  
 mutate(pf = ParkFactor(g\_home, g\_away, rf\_home, rf\_away, ra\_home, ra\_away),  
 team = school) %>%  
 select(team, pf)   
  
fields %>%   
 arrange((desc(pf))) %>%  
 pander::pander()

|  |  |
| --- | --- |
| team | pf |
| alma | 1.228 |
| calvin | 1.186 |
| trine | 1.069 |
| olivet | 0.9843 |
| hope | 0.9826 |
| kalamazoo | 0.982 |
| adrian | 0.9086 |
| albion | 0.8541 |

### Combine into master table

Combining Park Factor data and hitting data

all\_teams\_hitting <- all\_teams\_hitting %>%  
 left\_join(fields, by = "team", copy = TRUE)   
  
# all\_teams\_hitting\_158 <- all\_teams\_hitting  
  
all\_teams\_hitting <- all\_teams\_hitting %>%  
 filter(ab>40)

### Calculating Batting Runs

I calculate a lot of different stats in this section, and some are relevant to the Batting Runs formula while others are not. The important stats to note are wOBA, wRAA, wRC, and Batting Runs (see “Functions” section for explanations on these metrics).

all\_teams\_hitting <- all\_teams\_hitting %>%  
 mutate(x1b = h - x2b - x3b - hr,  
 wOBA = wOBA(bb, hbp, x1b, x2b, x3b, hr, ab, sf),  
 bbRate = bb / pa,  
 kRate = k / pa,  
 babip = BABIP(h, hr, ab, k, sf))  
  
lgR\_sum <- sum(all\_teams\_hitting$r)  
lgPA\_sum <- sum(all\_teams\_hitting$pa)  
lgPA\_sumh <- lgPA\_sum / 2  
lgwOBA\_avg <- mean(all\_teams\_hitting$wOBA)  
  
all\_teams\_hitting <- all\_teams\_hitting %>%  
 mutate(lgwOBA = lgwOBA\_avg,  
 lgR = lgR\_sum,  
 lgPA = lgPA\_sum,  
 lgPAh = lgPA\_sumh,   
 wRAA = wRAA(wOBA, lgwOBA, pa),  
 wRC = wRC(wOBA, lgwOBA, pa, lgR, lgPA),  
 lgwRC = sum(wRC) / 2,  
 batting\_runs = BattingRuns(wRAA, lgR, lgPA, pf, pa, lgwRC, lgPAh))

### Calculating base running runs and OFF

Here I calculate the elements needed for an alternative version of the Base Running Runs metric. The typical formula is BsR = wSB + UBR + wGDP, but only data for wSB is available, so it is only that element that is added into the OFF formula (which is labeled “d3 off” in the code).

For OFF, I could have standardized variables so that batting and base running both are the same weight. For example, Gunner Rainey of Adrian College had a Batting Runs at about 21, but a Base Running Runs of about 1.4. Of course, this doesn’t add to his OFF score that much, but I figured that it is okay to weight batting runs more importantly than base running runs since hitting is objectively more important anyway. In the future, I may try and standardize these variables and add them, just to see the results it produces.

all\_teams\_hitting <- all\_teams\_hitting %>%  
 mutate(outs = ab - h,  
 lgO = sum(outs),  
 rpo = RunsPerOut(lgR, lgO),  
 runCS = runCS(rpo),  
 lgSB = sum(sb),  
 lgCS = sum(cs),  
 lg1b = sum(x1b),  
 lgBB = sum(bb),  
 lgHBP = sum(hbp),  
 lgwSB = lgwSB(lgSB, lgCS, runCS, lg1b, lgBB, lgHBP),  
 wSB = wSB(sb, cs, runCS, lgwSB, x1b, bb, hbp),  
 d3\_off = wSB + batting\_runs  
 )

# view wSB in table format  
wsb\_table <- all\_teams\_hitting %>%  
 select(name, sb, cs, wSB)

# own variation of OFF metric  
d3\_off <- all\_teams\_hitting %>%  
 select(name, d3\_off) %>%  
 arrange((desc(d3\_off)))

I set up the first step to modeling in this next line of code, which I plan to return for predictive analysis. I filtered out seniors from this year because they wont be back next year (some will because of the NCAA eligibility rule that came after COVID, but right now I don’t know for sure which seniors will stay)

# for modeling next years stats  
non\_seniors <- all\_teams\_hitting %>%  
 filter( name %in% c("Evan Maday", "Ricky Padilla", "Brant Kym", "Thomas Miller") | year\_num %in% c(1, 2, 3))

## Visualizations

all\_teams\_hitting %>%  
 select(name, team, avg, wOBA, sb, d3\_off) %>%  
 arrange((desc(d3\_off))) %>%  
 mutate(wOBA = round(wOBA, digits = 3),  
 d3\_off = round(d3\_off, digits = 1)) %>%  
 head(10) %>%  
 knitr::kable()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| name | team | avg | wOBA | sb | d3\_off |
| Gunner Rainey | adrian | 0.411 | 0.451 | 7 | 22.3 |
| Evan Maday | hope | 0.360 | 0.456 | 34 | 19.9 |
| Spencer Baldwin | kalamazoo | 0.422 | 0.475 | 2 | 17.0 |
| Blake Bean | kalamazoo | 0.385 | 0.484 | 6 | 14.7 |
| Spencer Cable | hope | 0.350 | 0.441 | 10 | 13.4 |
| Doug Propson | kalamazoo | 0.358 | 0.434 | 18 | 12.7 |
| Kevin Cardentey | adrian | 0.405 | 0.431 | 1 | 11.6 |
| Thomas Miller | adrian | 0.328 | 0.391 | 9 | 11.1 |
| Brady Wood | adrian | 0.342 | 0.403 | 5 | 9.5 |
| Chase Coselman | kalamazoo | 0.400 | 0.442 | 0 | 9.0 |

The above table shows the players in the league with the best OFF metrics, paired with their Batting Average (AVG), wOBA, stolen bases (sb), and OFF’s.

Now, the lowest OFF’s.

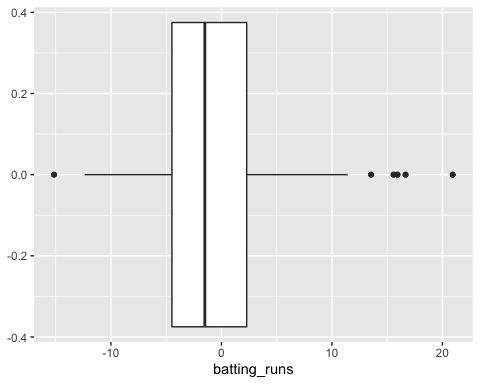
all\_teams\_hitting %>%  
 select(name, team, avg, wOBA, sb, d3\_off) %>%  
 mutate(wOBA = round(wOBA, digits = 3),  
 d3\_off = round(d3\_off, digits = 1)) %>%  
 arrange(d3\_off) %>%  
 head(10) %>%  
 knitr::kable()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| name | team | avg | wOBA | sb | d3\_off |
| Joe Fiorucci | trine | 0.202 | 0.215 | 1 | -17.0 |
| Nolan Chillcut | alma | 0.179 | 0.249 | 0 | -11.8 |
| Kevin Tuttle | alma | 0.302 | 0.314 | 7 | -11.7 |
| Jaret Koin | alma | 0.282 | 0.346 | 7 | -10.1 |
| Matt Hewitt | olivet | 0.250 | 0.283 | 10 | -8.8 |
| Sylas Woll | olivet | 0.247 | 0.267 | 8 | -8.5 |
| Andrew Gilpin | hope | 0.185 | 0.258 | 2 | -8.2 |
| Avery Fulford | trine | 0.180 | 0.220 | 0 | -8.1 |
| Ryan Petersen | calvin | 0.306 | 0.339 | 12 | -7.9 |
| Dylan Williams | calvin | 0.213 | 0.254 | 0 | -6.8 |

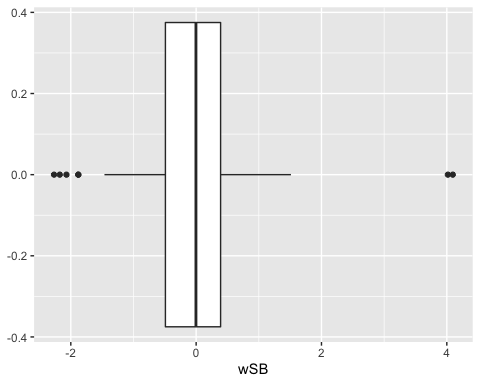
First, I am a little surprised by seeing certain players on the list - specifically Kevin Tuttle and Jaret Koin of Alma and Ryan Petersen of Calvin. These are all players with good Batting Averages in the .300 range, but slightly underwhelming wOBA’s in the low .300 range. Players with wOBA’s close to their normal batting averages don’t hit for power as much - they’re singles hitters. And this is fine - singles are still really good, but OFF only cares about your wOBA. So even if you hit good for average, if you don’t get extra base hits, OFF does not provide a positive number for a you.

The other strange happening I found was that Alma and Calvin’s teams were very middle-of-the-road in OFF this past year collectively. They both had great offenses - Calvin’s was best in the MIAA for most of the year, yet OFF didn’t provide them with super high numbers across the board. Park Factor is certainly part of the reason why - both Calvin and Alma had PF’s over 1, meaning they were slightly more hitter-friendly parks (or, they both were just better at home…I’m not convinced Park Factor is accurate in what it’s trying to compute, especially in a limited number of games at the D3 level). This high PF is weighted against Calvin and Alma’s offenses and adjusts their stats accordingly, which show to be worse than expected.

gf\_boxplot(data = all\_teams\_hitting, ~ batting\_runs)

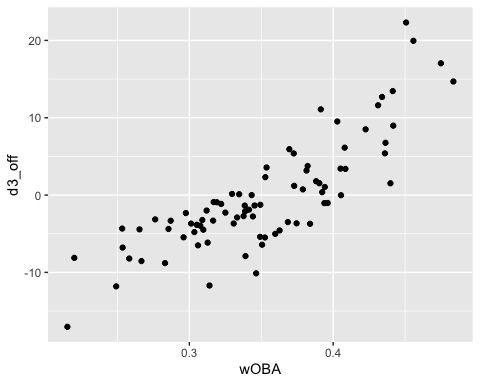


gf\_boxplot(data = all\_teams\_hitting, ~ wSB)



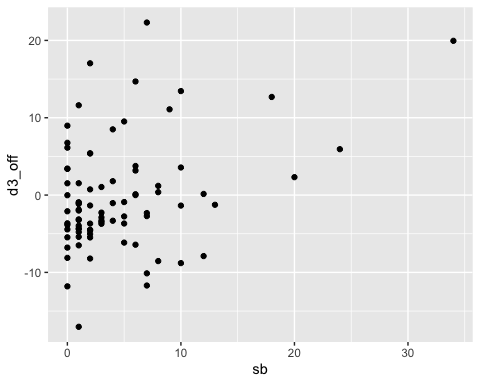
The plots above show the format of both Batting Runs and Base Running Runs. The average is unsurprisingly near 0 in both, as 0 is always league average because these stats add or subtract from the league average. Both stats use a lot of league averages, whether it is league average runs, wSB, or at-bats.

gf\_point(data = all\_teams\_hitting, d3\_off ~ wOBA)



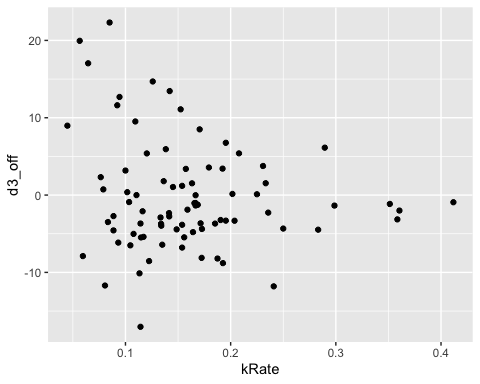
wOBA is very prominent in my version of OFF, so I wasn’t that surprised to see a positive trend in this particular scatter plot above. Better wOBA’s seem to indicate better OFF’s.

gf\_point(data = all\_teams\_hitting, d3\_off ~ sb)



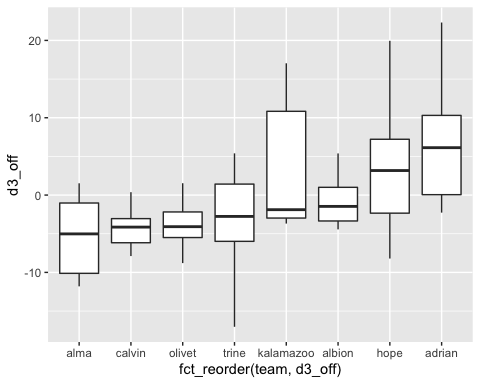
Stolen bases are not all that correlative to higher OFF’s. There are more factors in the base running runs formula than stolen bases, and some people with less stolen bases still have higher OFF’s because they hit better than they run. Hope’s team generally has higher OFF’s that expected because they steal a ton - way more than any other team in the MIAA, so it does help them quite a bit. This isn’t visible in the plot, I just know from following their team.

gf\_point(data = all\_teams\_hitting, d3\_off ~ kRate)



Strikeout rate isn’t in any of the formulas for any of the sabermetrics, but I thought it would be interesting to see if people who strikeout less have higher OFF’s. There is sort of a trumpeting shape going on. Certain players with strikeout rates of 10% or less have varying OFF’s - but it evens out closer to 0 with increased strikeout % rates.

gf\_boxplot (data = all\_teams\_hitting, d3\_off ~ fct\_reorder(team, d3\_off))



Lastly, I made a boxplot showing OFF’s from each of the 8 teams in the MIAA, increasing from lowest to highest. Alma and Calvin are at the bottom, weighed down by their park factors and their apparent lack of power. The two worst teams in the MIAA follow, being Olivet and Trine. Kalamazoo is 4th best, followed by Albion, which had the lowest park factor, weighting their OFF’s slightly higher. Lastly, two offensive powerhouses make it to the top - Hope gets there presumably with a combination of power and speed, and Adrian gets there because they are Adrian and they always win the MIAA.

OFF is interesting to look at in D3 baseball, and yet I’m not sure how I feel about it. Park Factor is one reason why I am mixed about these results. The metric is mainly used in the MLB where stadiums across the MLB differ in altitude and climate. The MIAA takes place only in Michigan and Indiana, and there may be differences in altitude across the state but probably not climate. So differences in PF shouldn’t be that much different in the MIAA. There may still be fields that have shorter fences than others, and it would really be interesting to look at PF’s from years past and compare. If Calvin has produced a PF of over 1 for many years in a row, then I have no problem accepting that maybe Calvin is just a hitters ballpark. But for now, I am disgruntled that Calvin’s solid offense is somehow below average according to OFF. Again, it would be interesting to repeat this process on data from another year.

I like OFF’s use of other good sabermetrics like wOBA and wSB, though. I think these stats are improvements over other stats commonly used in baseball, and it is certainly interesting to look at in the context of D3 collegiate baseball. There aren’t a lot of sabermetrics available in D3 baseball because they don’t matter as much for it. Only 3 D3 players got drafted this year - MLB teams don’t care about D3 players as much as D2 and D1 players. Of course, I care, and its all because of my enjoyment in watching these teams face off every year. There is some pure talent in the MIAA and it is fun to watch the teams compete every year. While I didn’t get to predictive modeling before the deadline, I still hope to complete that in the near future before the summer ends.

# Modeling

Fun part

non\_seniors <- non\_seniors %>%  
 select(team, wOBA, ops, pf, d3\_off) %>%  
 group\_by(team) %>%  
 summarize\_all("mean") %>%  
 mutate(wins = case\_when(  
 team == "adrian" ~ 37,  
 team == "calvin" ~ 22,  
 team == "hope" ~ 25,  
 team == "albion" ~ 13,  
 team == "trine" ~ 6,  
 team == "kalamazoo" ~ 25,  
 team == "olivet" ~ 6,  
 team == "alma" ~ 14  
 ),  
 losses = case\_when(  
 team == "adrian" ~ 10,  
 team == "calvin" ~ 16,  
 team == "hope" ~ 15,  
 team == "albion" ~ 27,  
 team == "trine" ~ 28,  
 team == "kalamazoo" ~ 15,  
 team == "olivet" ~ 24,  
 team == "alma" ~ 20  
 ),  
 win\_pct = wins / (wins + losses))

model1 <- betareg(win\_pct ~ d3\_off, data = non\_seniors)  
summary(model1)

##   
## Call:  
## betareg(formula = win\_pct ~ d3\_off, data = non\_seniors)  
##   
## Standardized weighted residuals 2:  
## Min 1Q Median 3Q Max   
## -1.8439 -0.6467 -0.0911 0.8908 1.8356   
##   
## Coefficients (mean model with logit link):  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.13174 0.21451 -0.614 0.53913   
## d3\_off 0.13772 0.04994 2.757 0.00583 \*\*  
##   
## Phi coefficients (precision model with identity link):  
## Estimate Std. Error z value Pr(>|z|)   
## (phi) 10.673 5.128 2.081 0.0374 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
##   
## Type of estimator: ML (maximum likelihood)  
## Log-likelihood: 4.563 on 3 Df  
## Pseudo R-squared: 0.5008  
## Number of iterations: 34 (BFGS) + 1 (Fisher scoring)

Positive correlation between win % and off. Not all that surprising.

cbind(  
 non\_seniors$team,  
 predict(model1, type = "response"),  
 predict(model1, type = "precision"),  
 predict(model1, type = "variance"),  
 predict(model1, type = "quantile", at = c(0.25, 0.5, 0.75))  
)

##   
## 1 "adrian" "0.656560866359759" "10.6732275105324" "0.0193167395153766"  
## 2 "albion" "0.451582539128782" "10.6732275105324" "0.0212157048476381"  
## 3 "alma" "0.315315966764211" "10.6732275105324" "0.0184946115093679"  
## 4 "calvin" "0.327701506305998" "10.6732275105324" "0.0188733774675425"  
## 5 "hope" "0.654866505477597" "10.6732275105324" "0.0193619429825411"  
## 6 "kalamazoo" "0.591781025708397" "10.6732275105324" "0.0206948971997632"  
## 7 "olivet" "0.283020909449902" "10.6732275105324" "0.0173833735426611"  
## 8 "trine" "0.404789248374559" "10.6732275105324" "0.0206399569063081"  
## q\_0.25 q\_0.5 q\_0.75   
## 1 "0.563272037441928" "0.666662377765452" "0.76015008307944"   
## 2 "0.346054751848316" "0.448449381382629" "0.553911992288921"  
## 3 "0.213286425311918" "0.30341760781155" "0.40521949951038"   
## 4 "0.224926338397443" "0.316593286612664" "0.419153014745442"  
## 5 "0.561388304125053" "0.664859469756082" "0.758534701469161"  
## 6 "0.492357398861058" "0.597715866814812" "0.697261260829682"  
## 7 "0.183398097147595" "0.269073539421712" "0.368441138023819"  
## 8 "0.299353203533026" "0.398633130528977" "0.503943058433904"