John Incantalupo and Austin Nolan

MTH-395 Research in Sabermetrics

December 6, 2022

The Advantages and Disadvantages of Calculating WAR Using xFIP

We began our research by looking to see how xFIP will alter the WAR statistic instead of a pitcher's wOBA allowed, and how this translates to Cape League teams' winning percentages. xFIP is a regressed version of FIP and measures the expected rate of home runs a pitcher *should have* allowed based on the number of fly balls allowed, rather than purely looking at actual home runs allowed for each pitcher. This is assuming a league-average home run-to-fly ball percentage. In the MLB, this number is typically between about 9 and 10%. However, when we looked at our Cape League data for the 2022 season, this number was only 5.7%. This number tells us that the pitchers allow fewer home runs per fly ball in the Cape Cod Baseball League than in Major League Baseball. There are several variables we can attribute to this, such as Statcast. Statcast was introduced into the MLB in 2015, with a series of cameras implemented to all parks by the company Chyron Hego, and a radar created by TrackMan, that can track all players, plays, events, situations, etc. Since Statcast took over the MLB and analytical departments across the league, we saw a massive increase in home runs and better overall seasons for hitters in the MLB in the following few years. This can be attributed to why the MLB home run to fly ball rate is about double what it is in the CCBL. A major reason for the CCBL home run to fly ball rate only being 5.7% could be because many of these players are swinging a wooden bat for the first time in an extended period (or ever) in a competitive setting like the Cape League, where MLB scouts are at every game. Many Cape League players come from some of the best college baseball schools in the country. In college, these talented players are

used to using aluminum bats, where the ball has more pop to it and can carry further and harder off the barrel. Another reason for the home run to fly ball rate to be this low could be purely the immaturity of the Cape League's players' bodies in comparison to those in the MLB. Many of them are still in their low-20's, still filling out their physical frame, and not yet at their peak strength from a physical standpoint. Trackman is used for the CCBL, but they keep their data secretive, so it is tough to distinguish how much these Cape League teams are using this data and implementing new strategies/methods of practice based off these results. In terms of the MLB, once Statcast took over, players began to learn how to elevate the ball better with a higher exit velocity that would be essential for becoming a more effective hitter, and for turning these deep fly balls into home runs, thus raising the home run to fly ball rate. Home run to fly ball rate is the variable that is replacing the number of actual home runs allowed in xFIP. It is easy to tell why this value is so different from the MLB to the CCBL and it gives us some background information as to why we are getting the numbers we have, and we can begin to look at how it is calculated.

xFIP is calculated by distributing a weight to the home run to fly ball rate, walks and hit-by-pitches, and strikeouts, dividing this by the number of innings pitched, and adding a FIP constant. The equation looks like this:

$$xFIP = \frac{13\times(Fly\ Balls \times LgHR/FB\%) + 3\times(BB + HBP) - 2\times K}{IP} + FIP\ constant$$

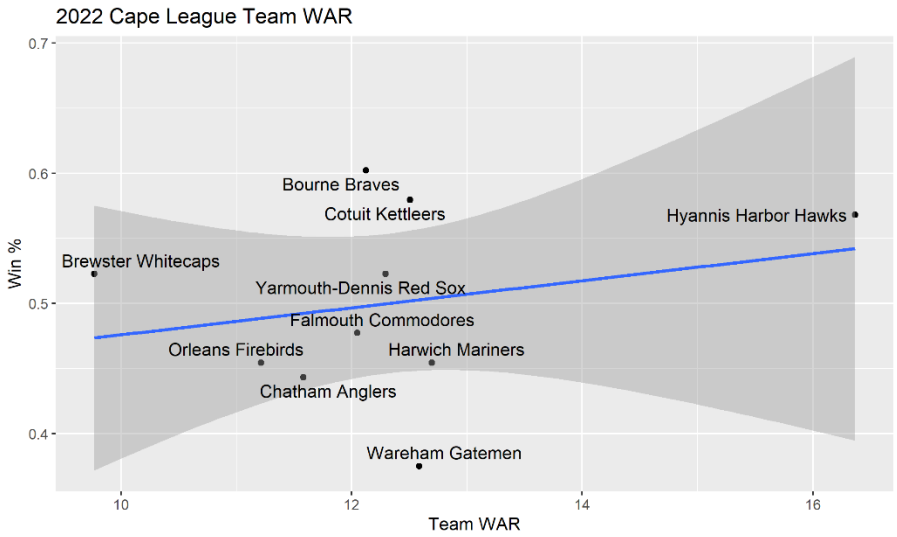
This formula is almost identical to the FIP formula, with the exception of how the number of "home runs" are calculated. The FIP constant is calculated by subtracting the league average FIP from the league average ERA. In our case, the FIP constant was 3.32 for the 2022 CCBL season. From the statistic FIP, we are looking to create a better indication of how good a pitcher truly is

by removing the defensive and baserunning aspects of the game. Where xFIP differs from FIP is that we can use xFIP to determine how a pitcher may fare based on the league average home run to fly ball rate. As Piper Slowinski from FanGraphs explains, pitchers can control the number of fly balls they give up based off pitch location, velocity, spin rate, and many other variables. However, only a small percentage of pitchers can alter their home run to fly ball rate. This is interesting because it can have both positive and negative impacts. xFIP can be very beneficial for us to use because we can see how many fly balls a pitcher allows, and what they are expected to allow in the future. However, the negative aspects of xFIP are important to be aware of. Analyzing xFIP for anybody in any league can go poorly. As Slowinski described, "only a limited set of pitchers" can truly influence their home run to fly ball rate. This can be by getting more players to swing under the ball, causing more short fly balls in the shallow outfield, infield, and foul territory. Also, xFIP is more of a predictor of future performance rather than an assessment of past performance. In fact, Slowinski states that "xFIP has one of the highest correlations with ERA of all the pitching metrics." Therefore, in a setting such as the Cape League, xFIP can be a good statistic to use to predict how each of these young players will fare throughout their hopefully successful careers.

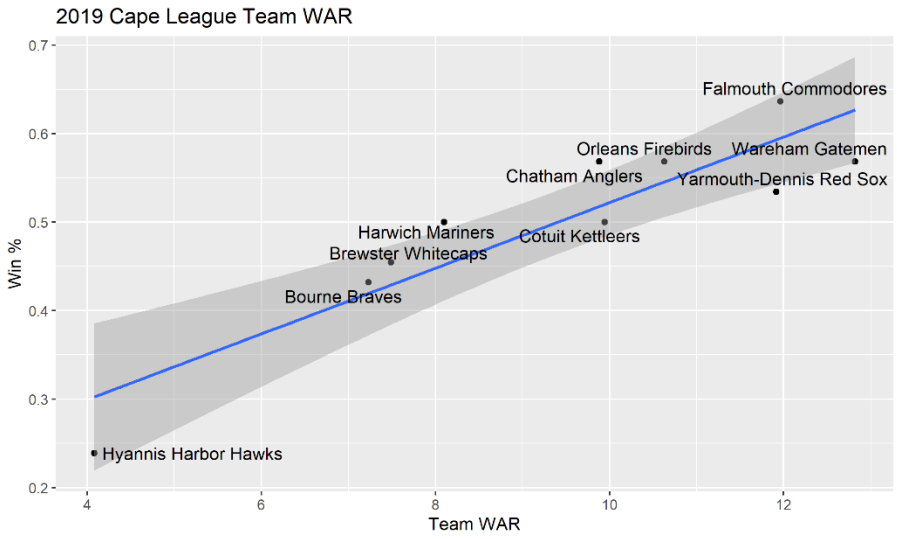
In order to calculate the xFIP data for the 2022 Cape Cod Baseball League season, we must find the league home run to fly ball rate. We went about this by calculating the total number of home runs and fly balls that each pitcher gave up. Then, we took the sum of home runs and fly balls and divided the two totals to give us a percentage. However, upon analyzing the CCBL play-by-play data, we noticed that there were a few home runs that were classified as line drives. However, the CCBL play-by-play data features two types of line drives: line drives and weak drives. Therefore, we decided to count the non-weak line drives as part of our fly ball

total. Once we had our fly balls totals as well as the league average home run to fly ball rate, we can calculate each pitcher's xFIP using the above formula. Then, we needed to figure out how to turn each pitcher's xFIP into runs, which we would then use to calculate WAR. Thankfully, FanGraphs uses FIP to calculate their pitcher WAR, and they are rather open about their process. So, we decided to borrow that process and use it for xFIP. First, we needed to calculate the xFIPR9, or xFIP runs per 9 innings, for each pitcher. This was done by taking the initial xFIP and adding the league average runs allowed per 9 and subtracting the league average ERA. We then used xFIPR9 to calculate the wRAA, or weighted runs above average, for each pitcher. To do this, we first subtracted the league average RA9 from xFIPR9. This gave us RAAP9, which is essentially wRAA per 9 innings. Then, we scaled RAAP9 to each pitcher's volume by multiplying it by the number of outs each pitcher recorded and then dividing the product by 27, which results in wRAA. Once we have found the wRAA for each pitcher, we can then calculate each pitcher's wRC by taking wRAA and subtracting the league average number of runs that would have scored over the number of batters that the pitcher faced. Since a good pitcher would have prevented fewer runs than the average pitcher, an above-average pitcher would have a negative wRC. Another positive of xFIP is that we do not need to calculate park factors. The only aspect of FIP that would require a park factor would be a home run factor. Since xFIP calculates the number of home runs that a pitcher would have given up given the league average home run to fly ball rate, it essentially can serve as its own park factor. Meanwhile, we then calculated the replacement level for the Cape League using an end date before July 3 or a start date after July 23 and a maximum of 19 batters faced and applied it to each pitcher to come up with their wRAR, or weighted runs above replacement. And finally, we used the runs per win formula to convert wRAR to WAR. Once we successfully calculated each Cape League player's

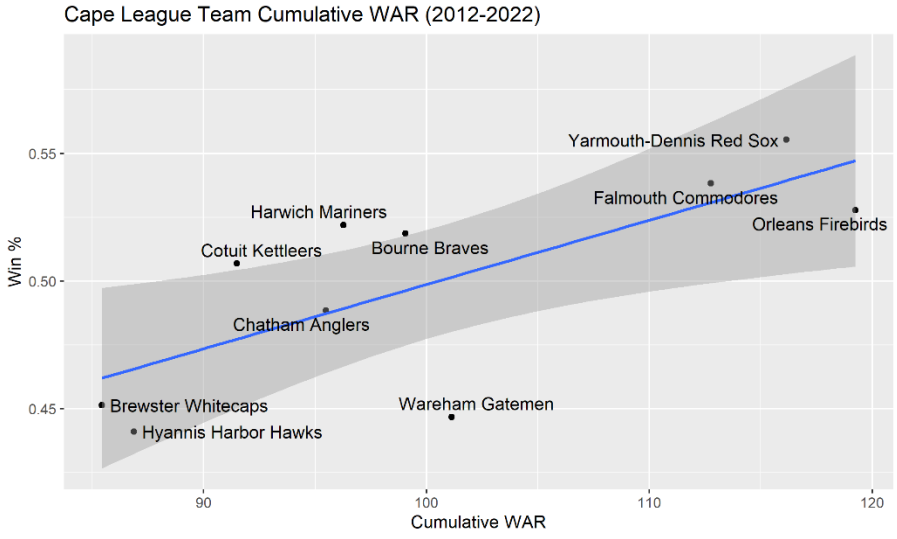
WAR, we wanted to see if WAR had a high correlation with team win percentage. To do this, we found each team's cumulative WAR by summing the WARs of every player that played for that team and plotted each team's cumulative WAR against their winning percentage (shown below).



As one can see from observing the above scatterplot, there was not much correlation between each team's cumulative WAR and win percentage. In fact, the correlation coefficient was just 0.2438424. This made us worried, as it showed the perhaps xFIP was not a good metric for measuring Cape League pitchers. We then decided to perform these same calculations to past CCBL season to see if that correlation coefficient would remain consistently low. We ran the same R program on every CCBL season since 2012, except for 2020 of course. To our surprise, we found that the lack of correlation between team WAR and win percentage in 2022 was merely a fluke, as the correlation coefficients from other seasons were much higher. There were still some correlation coefficients that were less than 0.7 in a few seasons, but the results did suggest that our new WAR formula could potentially correlate well with win percentage. The correlation was particularly high during the 2019 CCBL season, as the coefficient was a staggering 0.9067639 (scatterplot shown on top of the next page).



Lastly, we decided to test the correlation of WAR and team win percentage across each season cumulatively. To do this, we took each team's cumulative win percentage from the past ten Cape League seasons and plotted it against each team's WAR total from those seasons (shown below). We found that each team's cumulative win percentage and WAR from the past ten Cape League seasons generated a correlation coefficient of 0.7369054, which indicates a strong correlation between the two. The one team that was a true outlier was the Wareham Gateman, as their total WAR indicates that they should have won a lot more games than they did over the past 10 years.



When conducting our research, we were hoping to see a strong correlation between our new WAR formula and a team's win percentage in the Cape League. Although there were some Cape League seasons that posted such a strong correlation, such as 2019, the results were mostly inconsistent. The average correlation coefficient for the last 10 Cape League seasons was only 0.6944627, while the cumulative correlation for each Cape League team across the past 10 seasons was 0.7369054. We were hoping for a larger correlation, but these numbers truly display how to view xFIP as a statistic. Even xFIP may not have been the measurement of a pitcher's value during the Cape League season, it can tell us a pitcher's value in a future season. In fact, if one were to look at the most valuable Cape League seasons for a pitcher since 2012, one will run into some recognizable names, such as Sean Manaea, Dakota Hudson, and Alek Manoah. Each of these pitchers led the Cape League in pitcher WAR during their respective seasons on the Cape. Using xFIP as the main pitching formula in our WAR formula may not have told us who the best pitcher was in the Cape last season, but it can serve as an accurate predictor for who the MLB stars of tomorrow will be. One should look out for Grant Taylor and Liam Sullivan, two of the top WAR leaders during the 2022 Cape League season, in the coming years, as they are on the right track towards successful Major League careers.

R Codes

cWAR Code:

```
#cWAR22.r
#Working file for analyzing the 2022 Cape Cod Baseball League data
#And generating an improved WAR formula
#MTH 395 - Research in Sabermetrics - Fall 2022

#wOBAscale = 1.3008 - very high! (only 2013 and '15 are higher; last year was
1.1146)
#Weights for wOBA are comparable to the Dead-Ball era of MLB
#League OBP = .325
#Runs per game = 8.009
#Park factors for 2012-present (10 seasons)

#options("install.lock"=FALSE)
#install.packages("tidyverse")
library(tidyverse)
library(stringr)
library(lubridate)
library(ggrepel)
library(data.table)
library(abind)

CWS <- as.Date(0, origin = "2022-07-03")
ASG <- as.Date(0, origin = "2022-07-23")

cape22 <- as_tibble(readRDS("2022 CCBL Play by Play.rds"), .rows = NULL,
.name_repair = "minimal")
#Function for turning the raw play-by-play data into a data frame of plays

xRE <- c(0.469, 1.353, 1.080, 1.956, 0.827, 1.720, 1.391, 2.237,
         0.248, 0.928, 0.633, 1.344, 0.486, 1.099, 0.847, 1.512,
         0.093, 0.328, 0.302, 0.531, 0.209, 0.453, 0.414, 0.715)
#Expected run expectancy for wOBA = .325, team runs/game = 4.264

get_plays <- function(tpbp) {
  temp <- data.frame() #Data frame called "cape22" in original

  for (i in 1:length(tpbp)) {

    for (j in 1:2) {
      #Load each data frame
      boxscore <- tpbp[[i]][[j]]

      #Filter so that only events show
      boxscore <- filter(boxscore, BatterReached | !is.na(OutNumber) |
                           !is.na(BaseOutEvent) | !is.na(RunningEvent))

      boxscore <- select(boxscore, -"AtBatCount")
      if (j == 1)
        boxscore %>%
        mutate(GameID = tpbp[[i]][[2]]$GameID[[1]]) -> boxscore

      #Keep this part, modifying the column title to get "runs_roi"

      boxscore %>%
        group_by(Inning) %>%
```

```
        summarize(outs_inning = max(OutNumber, na.rm = TRUE),
                  runs_inning = sum(RunsScoredOnPlay),
                  runs_start = first(TeamRuns),
                  max_runs = runs_inning + runs_start) -> half_innings
      boxscore %>%
        inner_join(half_innings, by = "Inning") %>%
        mutate(runs_roi = max_runs - TeamRuns) ->
        boxscore

      #Add the states
      boxscore %>%
        mutate(bases =
                 paste(ifelse(is.na(Runner1B), 0, 1),
                       ifelse(is.na(Runner2B), 0, 1),
                       ifelse(is.na(Runner3B), 0, 1), sep = ""),
               state = paste(bases, Outs)) ->
        boxscore

      #Add the count for each batted-ball event
      boxscore %>%
        mutate(count = paste(Balls, sep = "-", Strikes))->
        boxscore

      #Add each (team, game) to the data so far
      temp <- bind_rows(temp, boxscore)
    }
  }
  temp %>%
    mutate(Date = as.Date(Date, format = "%m/%d/%Y")) -> temp
  return(temp)
}

plays <- get_plays(cape22)

#This is to fix our fly ball data (We are assuming that this HR was a fly ball.)
plays$TypeOfHit[[9707]] = "F"

#Create the run expectancy matrix
plays %>%
  filter(outs_inning == 3) -> playsC #Only consider completed half-innings
playsC %>%
  group_by(state) %>%
  summarize(Mean = mean(runs_roi), N = n()) %>%
  mutate (Outs = substr(state, 5, 5)) %>%
  arrange(Outs) -> runs

runs_out <- matrix(round(runs$Mean, 4), 8, 3)
dimnames(runs_out)[[2]] <- c("0 outs", "1 out", "2 outs")
dimnames(runs_out)[[1]] <- c("000", "001", "010", "011",
                             "100", "101", "110", "111")

runs <- tibble(runs, xRE)
buffer <- 50
runs %>%
  mutate(sRE = (Mean*N + xRE*buffer)/(N+buffer)) -> runs


#Add the run expectancy and run value of each play
v <- 0 * 1:length(plays$state)
for (i in 1:length(v)) {
```

```
  v[[i]] <- runs$sRE[[which(runs$state == plays$state[[i]])]]
}
plays %>%
  mutate(run_exp = v,
         run_value = RunsScoredOnPlay - run_exp) ->
  plays

for (i in 1:(length(plays$state) - 1)) {
  if(plays$Inning[[i]] == plays$Inning[[i + 1]])
    plays$run_value[[i]] <- plays$run_value[[i]] + plays$run_exp[[i + 1]]
}

#Add score differential - easier now!

plays %>%
  mutate(score_diff = TeamRuns - OpponentRuns) -> plays

#Label each play by type
play_levels <- c("Other", "E", "Out", "BB", "HBP", "1B", "2B", "3B", "HR")
plays %>%
  mutate(play_type = "Other",
         other_type = "") -> plays
for (i in 1:length(plays$play_type)) {
  if (!is.na(plays$Base4Comment[[i]]))
    if(str_detect(plays$Base4Comment[[i]], "HR"))
      plays$play_type[[i]] <- "HR"
  if (!is.na(plays$Base3Comment[[i]]))
    if(str_detect(plays$Base3Comment[[i]], "3B"))
      plays$play_type[[i]] <- "3B"
  if (!is.na(plays$Base2Comment[[i]]))
    if(str_detect(plays$Base2Comment[[i]], "2B"))
      plays$play_type[[i]] <- "2B"
  if (!is.na(plays$Base1Comment[[i]])) {
    if(str_detect(plays$Base1Comment[[i]], "1B"))
      plays$play_type[[i]] <- "1B"
    if(str_detect(plays$Base1Comment[[i]], "HBP"))
      plays$play_type[[i]] <- "HBP"
    if(str_detect(plays$Base1Comment[[i]], "BB") &
!str_detect(plays$Base1Comment[[i]], "IBB"))
      plays$play_type[[i]] <- "BB"
    if(str_detect(plays$Base1Comment[[i]], "E"))
      plays$play_type[[i]] <- "E"
  }
  if (!is.na(plays$OutComment[[i]]))
    if (!plays$BatterReached[[i]] & !str_detect(plays$OutComment[[i]], "SH"))
      plays$play_type[[i]] <- "Out"
  if (!is.na(plays$Base1Comment[[i]]))
    if(!plays$BatterReached[[i]] | str_detect(plays$Base1Comment[[i]], "FC"))
      plays$play_type[[i]] <- "Out"
  if (str_detect(plays$Description[[i]], "steals"))
    plays$other_type[[i]] <- "SB"
  if (str_detect(plays$Description[[i]], "caught stealing"))
    plays$other_type[[i]] <- "CS"
  if (str_detect(plays$Description[[i]], "picked off"))
    plays$other_type[[i]] <- "PO"
  if (str_detect(plays$Description[[i]], "wild pitch"))
    plays$other_type[[i]] <- "WP"
  if (str_detect(plays$Description[[i]], "passed ball"))
    plays$other_type[[i]] <- "PB"
  if (str_detect(plays$Description[[i]], "balk"))
```

```
    plays$other_type[[i]] <- "BK"

}

#Calculate the weights for wOBA

plays %>%
  group_by(play_type) %>%
  summarize(weight = mean(run_value),
            N = n()) -> wOBA
league_OBP <- sum(wOBA$N[wOBA$play_type %in% c("1B", "2B", "3B", "HR", "BB",
"HBP")]) /
  sum(wOBA$N[wOBA$play_type %in% c("Out", "E", "1B", "2B", "3B", "HR", "BB",
"HBP")])
wOBA %>%
  mutate(lin_wt = weight - weight[wOBA$play_type == "Out"]) -> wOBA
league_wOBA <- sum(wOBA$N[wOBA$play_type %in% c("1B", "2B", "3B", "HR", "BB",
"HBP")] *
                     wOBA$lin_wt[wOBA$play_type %in% c("1B", "2B", "3B", "HR",
"BB", "HBP")]) /
  sum(wOBA$N[wOBA$play_type %in% c("Out", "E", "1B", "2B", "3B", "HR", "BB",
"HBP")])
wOBA_scale <- league_OBP / league_wOBA
wOBA %>%
  mutate(true_wt = lin_wt * wOBA_scale) -> wOBA
league_runs <- sum(plays$RunsScoredOnPlay)

plays %>%
  group_by(other_type) %>%
  summarize(weight = mean(run_value),
            N = n()) -> others
SB_wt <- others$weight[which(others$other_type == "SB")]
CS_wt <- others$weight[which(others$other_type == "CS")]
PO_wt <- others$weight[which(others$other_type == "PO")]
WP_wt <- others$weight[which(others$other_type == "WP")]
PB_wt <- others$weight[which(others$other_type == "PB")]
BK_wt <- others$weight[which(others$other_type == "BK")]

#Park factors now calculated in ParkFactors.r
read.csv("CapeParks.csv") -> Parks
v <- 0 * 1:length(plays$OffenseTeam)
for (i in 1:length(v)) {
  if (plays$Top_Btm[[i]] == "Btm")
    v[[i]] <- Parks$PF[[which(Parks$OffenseTeam == plays$OffenseTeam[[i]])]]
  else
    v[[i]] <- Parks$PF[[which(Parks$OffenseTeam == plays$DefenseTeam[[i]])]]
}
plays %>%
  mutate(PF = v) -> plays

#Calculate wOBA for individual batters
v <- 0 * plays$run_value
for (i in 1:length(plays$play_type)) {
  if(plays$play_type[[i]] != "E" & plays$play_type[[i]] != "Other")
    v[[i]] <- wOBA$true_wt[[which(wOBA$play_type == plays$play_type[[i]])]]
}
plays %>%
  mutate(true_wt = v) -> plays

plays %>%
```

```
  filter(play_type %in% c("Out", "E", "1B", "2B", "3B", "HR", "BB", "HBP")) ->
playsB

league_PA <- length(playsB$run_value)

playsB %>%
  group_by(BatterName) %>%
  summarize(batter_wOBA = sum(true_wt) / n(),
            PA = n(),
            G = length(unique(GameID)),
            start_date = min(Date),
            end_date = max(Date),
            Team = OffenseTeam[last(which(Date == end_date))],
            PF = mean(PF),
            RE24 = sum(run_value),
            BA = sum(play_type %in% c("1B", "2B", "3B", "HR")) / sum(play_type
%in% c("1B", "2B", "3B", "HR", "Out", "E")),
            OBP = sum(play_type %in% c("1B", "2B", "3B", "HR", "BB", "HBP")) / PA,
            SLG = (sum(play_type == "1B") + 2 * sum(play_type == "2B") + 3 *
sum(play_type == "3B") + 4 * sum(play_type == "HR"))/ sum(play_type %in% c("1B",
"2B", "3B", "HR", "Out", "E")),
            BsR = SB_wt * sum(str_detect(Base2Comment, "SB") |
                              str_detect(Base3Comment, "SB") |
                              str_detect(Base4Comment, "SB"), na.rm = TRUE)
            + CS_wt * sum(str_detect(OutComment, "CS"), na.rm = TRUE),
            RE24.PA = RE24/PA) -> Batting

#Convert wOBA to runs
Batting %>%
  mutate(wRAA = (batter_wOBA - league_OBP) / wOBA_scale * PA,
         wRC = wRAA + league_runs / league_PA * PA + BsR,
         wRCPlus = (wRC / PA + (1 - PF/ 100) * league_runs / league_PA)
         / (sum(wRC) / league_PA) * 100) %>%
  arrange(desc(wRC)) -> Batting

#Test the model
OPS <- lm(batter_wOBA ~ OBP + SLG, data = Batting)

summary(Batting$batter_wOBA)
cor.test(Batting$batter_wOBA, Batting$BA)
cor.test(Batting$batter_wOBA, Batting$OBP)
cor.test(Batting$batter_wOBA, Batting$SLG)

#Breakdown by ball-strike count: final, then pass-through
playsB %>%
  group_by(count) %>%
  summarize(count_wOBA = mean(true_wt),
            count_RE = mean(run_value)) -> CountFinal
CountFinal %>%
  mutate(delta_RE = count_RE - count_RE[[1]]) -> CountFinal

playsB %>%
  mutate(c00 = TRUE,
         c10 = grepl("^B", Pitches),
         c01 = grepl("^[SCF]", Pitches),
         c20 = grepl("^B{2,}", Pitches),
         c11 = grepl("^B[SCF]|[SCF]B", Pitches),
         c02 = grepl("^[SCF]{2,}", Pitches),
         c30 = grepl("^B{3,}", Pitches),
         c21 = grepl("^[SCF]BB|B[SCF]B|BB[SCF]", Pitches),
```

```
          c31 = grepl("^[SCF]BBB|B[SCF]BB|BB[SCF]B|[SCF]BBB", Pitches),
          c12 = grepl("^B[SCF][SCF]|[SCF]B[SCF]|[SCF][SCF]F*B", Pitches),
          c22 = grepl("^BB[SCF][SCF]|B[SCF]B[SCF]|[SCF]BB[SCF]|
                      B[SCF][SCF]F*B|[SCF]B[SCF]F*B|[SCF][SCF]F*BF*B", Pitches),
          c32 = grepl("^[SCF]*B[SCF]*B[SCF]*B", Pitches) &
            grepl("^B*[SCF]B*[SCF]", Pitches)
  ) -> playsB
CountPassThrough <- playsB %>%
  select(run_value, true_wt, c00, c10, c20, c30, c01, c11, c21, c31, c02, c12,
c22, c32)
CountsTidy <- CountPassThrough %>%
  gather(key = "Count", value = "PassThrough", -run_value, -true_wt)
CountwOBA <- CountsTidy %>%
  filter(PassThrough == 1) %>%
  group_by(Count) %>%
  summarize(N = n(), value = mean(true_wt))
CountwOBA <- CountwOBA %>%
  mutate(balls = str_sub(Count, 2, 2),
         strikes = str_sub(Count, 3, 3))
CountPlot <- CountwOBA %>%
  ggplot(aes(x = strikes, y = balls, fill = value)) +
  geom_tile() +
  geom_text(aes(label = round(value, 3))) +
  scale_fill_gradient2("wOBA", low = "blue", high = "red", mid = "white", midpoint
= league_OBP)

#Calculate wOBA allowed for pitchers
playsB %>%
  group_by(P) %>%
  summarize(pitcher_wOBA = sum(true_wt) / n(),
            BF = n(),
            G = length(unique(GameID)),
            start_date = min(Date),
            end_date = max(Date),
            Team = DefenseTeam[last(which(Date == end_date))],
            PF = mean(PF),
            RE24 = sum(run_value),
            HR = sum(play_type == "HR"),
            BB = sum(play_type %in% c("BB", "HBP")),
            K = sum(str_detect(OutComment, "^K") | str_detect(Base1Comment, "^K"),
na.rm = TRUE),
            nGB = sum(TypeOfHit %in% c("G", "B", "D"), na.rm = TRUE) -
sum(TypeOfHit %in% c("F", "P", "W", "L"), na.rm = TRUE),
            FB.LD = sum(TypeOfHit %in% c("F", "L"), na.rm = TRUE),
            H = sum(play_type %in% c("1B", "2B", "3B", "HR")),
            RA = sum(!is.na(Base4Comment)),
            Earned = sum(as.numeric(ER), na.rm = TRUE)) -> Pitching

plays %>%
  group_by(P) %>%
  summarize(IPOuts = sum(BatterReached == FALSE & play_type != "Other") +
              sum(BaseOutEvent, na.rm = TRUE) +
              sum(str_detect(OutComment, "TP"), na.rm = TRUE)
  ) -> PitchingOuts

inner_join(Pitching, PitchingOuts) -> Pitching

league_IPOuts <- sum(Pitching$IPOuts)
league_ERA <- sum(Pitching$Earned) / (league_IPOuts / 27)
league_RA9 <- sum(Pitching$RA) / (league_IPOuts / 27)
```

```
league_FIP <- (13 * sum(Pitching$HR) + 3 * sum(Pitching$BB) - 2 * sum(Pitching$K))
/ (league_IPOuts / 3)
league_HR.FB <- sum(Pitching$HR) / sum(Pitching$FB.LD) * 100

Pitching %>%
  mutate(HR.FB = HR / FB.LD,
         ERA = Earned * 27 / IPOuts,
         RA9 = RA * 27 / IPOuts,
         WHIP = (BB + H) / (IPOuts / 3),
         FIP = (13 * HR + 3 * BB - 2 * K) / (IPOuts / 3) + (league_ERA -
league_FIP),
         xFIP = ((13 * FB.LD / league_HR.FB) + 3 * BB - 2 * K) / (IPOuts / 3) +
(league_ERA - league_FIP)
  ) -> Pitching

#Convert xFIP to runs
Pitching %>%
  mutate(xFIPR9 = xFIP + league_RA9 - league_ERA) -> Pitching #xFIP runs per 9

Pitching %>%
  mutate(RAAP9 = xFIPR9 - league_RA9, #Runs Above Average per 9
         wRAA = RAAP9 * (IPOuts / 27), #Cumulative runs above average
         wRC = wRAA - league_runs / league_PA * BF,
         FIPMinus = FIP * (2 - PF / 100) / league_ERA * 100) %>%
  arrange(wRC) -> Pitching

#Test the model
FIPfit <- lm(pitcher_wOBA ~ I(HR * 3 / IPOuts) + I(BB * 3 / IPOuts) + I(K * 3 /
IPOuts), data = Pitching)

summary(Pitching$pitcher_wOBA)
Pitching %>%
  filter(IPOuts > 0) -> PitchingF

cor.test(PitchingF$pitcher_wOBA, PitchingF$ERA)
cor.test(PitchingF$pitcher_wOBA, PitchingF$RA9)
cor.test(PitchingF$pitcher_wOBA, PitchingF$WHIP)
cor.test(PitchingF$pitcher_wOBA, PitchingF$FIP)

#Calculate coefficients for SIERA
PitchingF %>%
  mutate(KR = K / BF,
         BBR = BB / BF,
         nGBR = nGB / BF) -> PitchingF
SIERA <- lm(RA9 ~ KR + BBR + nGBR + I(KR^2)
            #+ I(BBR^2)
            + I(nGBR*abs(nGBR))
            #+ I(KR * BBR)
            + I(KR * nGBR) + I(BBR * nGBR), data = PitchingF)

#Calculate baseline for replacement level for batters
Batting %>%
  mutate(isTemp = (end_date <= CWS | start_date >= ASG)
         & PA < 40) -> Batting

Batting %>%
  filter(isTemp == TRUE) -> BTemp

BReplace <- sum(BTemp$wRC) / sum(BTemp$PA)
RPW <- 9 * (league_runs / (league_IPOuts / 3)) * 1.5 + 3
```

```
#Calculate WAR for batters
Batting %>%
  mutate(wRAR = (wRC - BReplace * PA) * 100 / PF,
         WAR = wRAR / RPW) -> Batting
Batting %>%
  arrange(desc(WAR)) -> Batting

#Calculate baseline for replacement level for pitchers
Pitching %>%
  mutate(isTemp = (end_date <= CWS | start_date >= ASG)
         & BF < 20) -> Pitching

Pitching %>%
  filter(isTemp == TRUE) -> PTemp

PReplace <- sum(PTemp$wRC, na.rm = TRUE) / sum(PTemp$BF, na.rm = TRUE)

Pitching %>%
  filter(IPOuts > 0) -> PitchingF #For "Finite"

#Calculate WAR for pitchers
PitchingF %>%
  mutate(wRAR = (PReplace * BF - wRC),
         WAR = wRAR / RPW) -> PitchingF
PitchingF %>%
  arrange(desc(WAR)) -> PitchingF

#Calculate team WAR and test our model
Standings <- read.csv("CCBL22Standings.csv")
Standings %>%
  mutate(Wpct = (Wins + 0.5 * Ties) / (Wins + Losses + Ties)) -> Standings
#May add Pythagorean win pct if .csv includes runs scored and runs allowed

Batting %>%
  group_by(Team) %>%
  summarize(batting_WAR = sum(WAR)) -> off_WAR

PitchingF %>%
  group_by(Team) %>%
  summarize(pitching_WAR = sum(WAR)) -> def_WAR

inner_join(off_WAR, def_WAR) -> Team_WAR
inner_join(Standings, Team_WAR) -> Standings
Standings %>%
  mutate(WAR = batting_WAR + pitching_WAR) -> Standings

cor(Standings$WAR, Standings$Wpct)
ggplot(Standings, aes(x = WAR, y = Wpct)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_text_repel(aes(label = Team)) +
  ggtitle("2022 Cape League Team WAR") +
  xlab("Team WAR") +
  ylab("Win %")
WAR_fit <- lm(Standings$Wpct ~ Standings$WAR)
confint(WAR_fit, level = 0.8)

write.csv(Standings, "WARStandings22.csv")
```

Cumulative Team Data Code:

```
read.csv("WARStandings12.csv") -> WARStandings12
read.csv("WARStandings13.csv") -> WARStandings13
read.csv("WARStandings14.csv") -> WARStandings14
read.csv("WARStandings15.csv") -> WARStandings15
read.csv("WARStandings16.csv") -> WARStandings16
read.csv("WARStandings17.csv") -> WARStandings17
read.csv("WARStandings18.csv") -> WARStandings18
read.csv("WARStandings19.csv") -> WARStandings19
read.csv("WARStandings21.csv") -> WARStandings21
read.csv("WARStandings22.csv") -> WARStandings22

WARStandings12 %>%
  mutate(Year = 2012) -> WARStandings12
WARStandings13 %>%
  mutate(Year = 2013) -> WARStandings13
WARStandings14 %>%
  mutate(Year = 2014) -> WARStandings14
WARStandings15 %>%
  mutate(Year = 2015) -> WARStandings15
WARStandings16 %>%
  mutate(Year = 2016) -> WARStandings16
WARStandings17 %>%
  mutate(Year = 2017) -> WARStandings17
WARStandings18 %>%
  mutate(Year = 2018) -> WARStandings18
WARStandings19 %>%
  mutate(Year = 2019) -> WARStandings19
WARStandings21 %>%
  mutate(Year = 2021) -> WARStandings21
WARStandings22 %>%
  mutate(Year = 2022) -> WARStandings22

TotalWARStandings <- rbind(WARStandings12,
                           WARStandings13,
                           WARStandings14,
                           WARStandings15,
                           WARStandings16,
                           WARStandings17,
                           WARStandings18,
                           WARStandings19,
                           WARStandings21,
                           WARStandings22)

TotalWARStandings %>%
  group_by(Team) %>%
  summarize(Wins = sum(Wins),
            Losses = sum(Losses),
            Ties = sum(Ties),
            Wpct = (sum(Wins) + 0.5 * sum(Ties)) / (sum(Wins) + sum(Losses) +
sum(Ties)),
            batting_WAR = sum(batting_WAR),
            pitching_WAR = sum(pitching_WAR),
            WAR = sum(WAR)) -> TeamCumStandings

cor(TeamCumStandings$WAR, TeamCumStandings$Wpct)
ggplot(TeamCumStandings, aes(x = WAR, y = Wpct)) +
  geom_point() +
```

```
geom_smooth(method = "lm") +
geom_text_repel(aes(label = Team)) +
ggtitle("Cape League Team Cumulative WAR (2012-2022)") +
xlab("Cumulative WAR") +
ylab("Win %")
```

Bibliography

Expected fielding independent pitching (xFIP) } glossary. MLB.com. (n.d.). Retrieved November 30, 2022, from https://www.mlb.com/glossary/advanced-stats/expected-fielding-independent-pitching

Slowinski, P. (n.d.). *War for pitchers*. Sabermetrics Library. Retrieved November 30, 2022, from https://library.fangraphs.com/war/calculating-war-pitchers/#:~:text=Here%E2%80%99s%20the%20basic%20construction:%20WAR%20%3D%20%5B%20%5B,a%20pitcher%E2%80%99s%20WAR%20is%20to%20calculate%20their%20FIP

Slowinski, P. (n.d.). *XFIP*. Sabermetrics Library. Retrieved November 30, 2022, from https://library.fangraphs.com/pitching/xfip/#:~:text=They%20can%20control%20how%20many,useful%20statistic%20if%20used%20properly.

Waldo, M. (2021, February 21). *xFIP vs ERA: The players to target and the guys to avoid*. Fantasy Baseball 2021 | Fantistics. Retrieved November 30, 2022, from https://www.insiderbaseball.com/blog/2021/02/xfip_vs_era_the_players_to_target_and_the_guys_to_avoid.html#:~:text=xFIP%2C%20or%20expected%20fielding%20independent,%2Dpitches%2C%20and%20flyballs%20allowed.