Mini Project 1

Brenden Bready

2024-02-08

Assignment note: Some code will be shown throughout to display certain calculations, but all code can be found at the end in the appendix.

1. True Shooting Percentage coefficient

For this problem, I sourced my data from the hoopR package, loading information for all games (play by play) from 2018 to 2021. To find our updated coefficient, our goal is to divide the trips to the free throw line by the total number of free throws shot.

```
# Get data from all games 2018-2021 from hoopR package
pbp = load_nba_pbp(2018:2021)
```

To start, I calculated the total number of free throw shots across the four seasons to be 221,621. This calculation can be seen in the chunk below.

```
# Total number of free throw shots
(numftshots = pbp %>%
  filter(str_detect(type_text, 'Free Throw')) %>% # Filter data to only free throws
  summarise('Free Throw Shots' = n())) # Count number of rows of data (total free throws)
```

```
## # A tibble: 1 x 1
##   `Free Throw Shots`
##                <int>
## 1             221621
```

Getting the total number of shots to the line can be a bit more complicated. To do this, I first filtered the data to only free throws, and then filtered to only include the final shot of a 2 attempt or 3 attempt free throw (so that and-1 opportunities are excluded). After this, I took out all technical and clear path fouls, as those are accounted for in our possessions formula. These will leave us with just the total number of free throw trips to the line, excluding and-1s and technicals.

```
# Number of trips to the line
(numft = pbp %>%
  filter(str_detect(type_text, 'Free Throw'), # Filter to only free throws
         str_detect(type_text, '2 of 2|3 of 3'), # Filter to final shots
         !str_detect(type_text, 'Technical|Clear Path|Flagrant')) %>% # Take out techs
  summarise('Trips to line' = n())) # Count the total rows left
```

```
## # A tibble: 1 x 1
##   `Trips to line`
##             <int>
## 1           93775
```

From here, we can simply divide our trips to the line by our total free throw shots to get our updated coefficient of 0.423

```
## [1] "Updated Coefficient:  0.423"
```

To test if the coefficient is significantly different, we can perform a simple t-test. Our test statistic is $t = \frac{0.423-0.44}{s/\sqrt{n}}$. Given that our n (number of games) is 4934, our test statistic will always be very high for any reasonable value of s ($s < 0.2$), and thus our p-value is ≈ 0. This means that we have significant evidence that our true coefficient is less than 0.44. This overestimate may mean that we are slightly underestimating true shooting percentage for most players, leading to an improper analysis.

2. Baseball

To begin the baseball analysis, I used the bref_daily_batter() and bref_daily_pitcher() functions from baseballr to gain information on all batters and pitchers that played in the league during the desired time. Details for the data download are in the appendix. As a note, there is no data from the 2020 season through the end of May for the 2020 season, so this is not counted towards this analysis.

All datasets appear in the same form. Examples for the beginning of the historial dataset (2016-2018) are seen below in Tables 1 and 2.

Table 1: OPS Historical Data (All games 2016-2018)

bbref_id	Name	G	AB	OPS
lindofr01	Francisco Lindor	475	1916	0.837
machama01	Manny Machado	475	1902	0.855
blackch02	Charlie Blackmon	457	1847	0.932
goldspa01	Paul Goldschmidt	470	1730	0.929
markani01	Nick Markakis	480	1815	0.763
bettsmo01	Mookie Betts	447	1820	0.917

Table 2: WHIP Historical Data (All games 2016-2018)

bbref_id	Name	G	IP	WHIP
scherma01	Max Scherzer	98	649.2	0.927
verlaju01	Justin Verlander	101	647.2	1.024
klubeco01	Corey Kluber	94	633.2	0.974
porceri01	Rick Porcello	99	617.2	1.188
salech01	Chris Sale	91	599.0	0.967
roarkta01	Tanner Roark	97	571.2	1.258

a) Using prior based on average player

In class, we discussed how the likelihood function of pass completion could be modeled using a Bernoulli distribution. In this case, we want to use a Stein Estimator in order to get future estimates for the OPS and WHIPs of our players. Overall means for OPS and WHIP can be seen below.

```
# Average player OPS
avgOPS = mean(batters1618$OPS, na.rm = T)
paste("Average OPS 2016-2018: ", round(avgOPS, 3))
```

```
## [1] "Average OPS 2016-2018:  0.488"
```

```
# Average player WHIP
avgWHIP = mean(pitchers1618$WHIP, na.rm = T)
paste("Average WHIP 2016-2018: ", round(avgWHIP, 3))
```

```
## [1] "Average WHIP 2016-2018:  1.576"
```

Next, we need to find our c value and then our Stein estimations. As a note, our c values are small, meaning we have very little information about each player individually. This makes sense given that we are performing the analysis on only the overall mean as a whole. This then results in predictions that are all nearly identical to each other, very similar to the overall mean. Mean and variance for our OPS and WHIP predictions are shown below for the 2019 season. I will show the code for the 2019 calculation, but code for the 2021 calculation will be in the appendix

```
# Stein predictions OPS 2019 based on 2016-2018 mean player
c = 1 - (length(batters1618$OPS)-3)*var(batters1618$OPS, na.rm = T)/
  sum((batters19$OPS - avgOPS)^2, na.rm=T) # calculating c value
predictionsOPS19 = avgOPS + c*(batters19$OPS - avgOPS) # Getting stein estimates
paste("Mean of predictions for 2019 OPS: ",
      round(mean(predictionsOPS19, na.rm = T), 3)) # Mean of predictions for OPS
```

```
## [1] "Mean of predictions for 2019 OPS:  0.45"
```

```
paste("Variance of predictions for 2019 OPS: ",
      round(var(predictionsOPS19, na.rm = T), 3)) # Variance of predictions for OPS
```

```
## [1] "Variance of predictions for 2019 OPS:  0.032"
```

```
# Stein predictions WHIP 2019 based on 2016-2018 mean player
c = 1 - (length(pitchers1618$WHIP)-3)*var(pitchers1618$WHIP, na.rm = T)/
  sum((pitchers19$WHIP - avgWHIP)^2, na.rm=T) # Calculating c value
predictionsWHIP19 = avgWHIP + c*(pitchers19$WHIP - avgWHIP) # Getting stein estimates
paste("Mean of predictions for 2019 WHIP: ",
      round(mean(predictionsWHIP19, na.rm = T), 3)) # Mean of predictions for WHIP
```

```
## [1] "Mean of predictions for 2019 WHIP:  1.585"
```

```
paste("Variance of predictions for 2019 WHIP: ",
      round(var(predictionsWHIP19, na.rm = T), 3)) # Variance of predictions for WHIP
```

```
## [1] "Variance of predictions for 2019 WHIP:  0.163"
```

To analyze how well our predictions performed for the 2019 season, I will use mean square error. The calculation is shown below.

```
# MSE for OPS and WHIP
paste("MSE for OPS 2019: ",
      round(mean((predictionsOPS19 - batter19total$OPSfull)^2, na.rm=T), 3))
```

```
## [1] "MSE for OPS 2019:  0.236"
```

```
paste("MSE for WHIP 2019: ",
      round(mean((predictionsWHIP19 - pitcher19total$WHIPfull)^2, na.rm=T), 3))
```

```
## [1] "MSE for WHIP 2019:  0.745"
```

For 2020, due to COVID 19 we have no data for the season through May 31st. For this reason, we will just use the overall mean as our prediction for all players. MSE's are shown below.

```
# Overall mean (2016-2018) as prediction, MSE shown below for 2020
paste("MSE for OPS 2020: ",
      round(mean((avgOPS - battersFull20$OPS)^2, na.rm=T), 3))
```

```
## [1] "MSE for OPS 2020:  0.083"
```

```
paste("MSE for WHIP 2020: ",
      round(mean((avgWHIP - pitchersFull20$WHIP)^2, na.rm=T), 3))
```

```
## [1] "MSE for WHIP 2020:  1.429"
```

Similar to 2019, calculations for 2021 are shown below.

```
## [1] "Mean of OPS 2021 predictions:  0.442"
```

```
## [1] "Variance of OPS 2021 predictions:  0.13"
```

```
## [1] "Mean of WHIP 2021 predictions:  1.713"
```

```
## [1] "Variance of WHIP 2021 predictions:  1.165"
```

To analyze how well our predictions performed for the 2021 season, I will use mean square error. The calculation is shown below.

```
## [1] "MSE of OPS 2021 predictions:  0.388"
```

```
## [1] "MSE of WHIP 2021 predictions:  2.465"
```

Our MSE's for both OPS and WHIP are very high when we only use the overall averages for all players. This is because our Stein estimates are heavily weighted on the overall means, and thus overall estimates are very similar to the overall mean. When base our predictions based on the players individually, we will hopefully achieve better results. Results from this section are summarized in the following table.

Table 3: MSEs using overall mean prior

Year	OPS.MSE	WHIP.MSE
2019	0.24	0.75
2020	0.08	1.42
2021	0.39	2.46

b) Using Prior based on each player average

To do the analysis based on each player, I will begin by joining the data to ensure all players line up correctly. We only want to include players that played between 2016-2018, as well as 2019, 2020, or 2021.

Once the data is clean and all in one form, we are able to perform the player by player analysis using the Stein estimator. This time, we will use each player's average from 2016-2018 rather than the overall average. Again, I will show the code for 2019 here, but for 2021 the code will be in the appendix.

```
# Stein predictions for OPS 2019 player specific
pbar19 = bat19$OPS # P bar for stein (historical data)
phat19 = bat19$OPSearly # P hat for stein (current season data)
c = 1 - (1 - 3)*var(batters1618$OPS, na.rm = T)/
  sum((phat19 - pbar19)^2,na.rm = T) # c value for stein
preds19 = pbar19 + c*(phat19-pbar19) # predictions

# Calculating our MSE for 2019
paste("MSE of OPS 2019 predictions:",
      round(mean((preds19 - bat19$OPSfull)^2, na.rm = T),3))
```

```
## [1] "MSE of OPS 2019 predictions: 0.049"
```

Similar to last time, we will use just the previous player averages for 2020 predictions because there is no current season data.

```
preds20 = bat20$OPS.x # Setting predictions to historical averages
paste("MSE of OPS 2020 predictions:",
      round(mean((preds20 - bat20$OPS.y)^2,na.rm = T),3))# Calculating our MSE for 2020
```

```
## [1] "MSE of OPS 2020 predictions: 0.046"
```

Again for 2021, we will follow the same steps as 2019.

```
## [1] "MSE of OPS 2021 predictions: 0.022"
```

The analysis for WHIP will be the same for that of OPS. Analysis as follows:

```r
# Stein estimates for WHIP 2019 based on player specific info
pbar19 = pitch19$WHIP # P bar value (historical values)
phat19 = pitch19$WHIPearly # P hat value (current season data)
c = 1 - (1 - 3)*var(pitch19$WHIP, na.rm = T)/
  sum((phat19 - pbar19)^2,na.rm = T) # c value
preds19 = pbar19 + c*(phat19-pbar19) # WHIP predictions 2019

# Calculating our MSE for 2019
paste("MSE of WHIP 2019 predictions:",
      round(mean((preds19 - pitch19$WHIPfull)^2, na.rm = T),3))
```

```
## [1] "MSE of WHIP 2019 predictions: 0.459"
```

Similar to last time, we will use just the previous player averages for 2020 predictions because there is no current season data.

```r
# MSE calculation for 2020 based on player previous averages
preds20 = pitch20$WHIP.x # Setting predictions to historical averages
paste("MSE of WHIP 2020 predictions:",
      round(mean((preds20-pitch20$WHIP.y)^2,na.rm = T),3))# Calculating our MSE for 2020
```

```
## [1] "MSE of WHIP 2020 predictions: 0.686"
```

Again for 2021, we will follow the same steps as 2019.

```
## [1] "MSE of WHIP 2021 predictions:  0.493"
```

The table below summarizes the results based on MSE

Table 4: MSEs using player specific priors

Year	OPS.MSE	WHIP.MSE
2019	0.05	0.46
2020	0.05	0.69
2021	0.02	0.49

c) Comparing "on pace" and summary

To compute the predictions for what the player is "on pace" for, I will compare the predictions of the early OPS and WHIP to what is it at the end of the season. I expect to see a greater variability in these predictions, given that not much of the season has been played yet. As a note, this will only be done for 2019 and 2021, because there is no early season data for 2020.

```r
# Prediction and MSE for predictions based on "on pace" 2019
predOPS19 = batter19total$OPSearly # Prediction for 2019 OPS
paste("MSE of OPS 2019 by on pace:",
      round(mean((predOPS19 - batter19total$OPSfull)^2,na.rm=T),3))# MSE for 2019 OPS
```

```
## [1] "MSE of OPS 2019 by on pace: 0.054"
```

```r
# Prediction and MSE for predictions based on "on pace" 2021
predOPS21 = batter21total$OPSearly # Prediction for 2021 OPS
paste("MSE of OPS 2021 by on pace:",
      round(mean((predOPS21 - batter21total$OPSfull)^2,na.rm=T),3))# MSE for 2021 OPS
```

```
## [1] "MSE of OPS 2021 by on pace: 0.04"
```

Now for WHIP:

```r
# Prediction and MSE for predictions based on "on pace" 2019
predWHIP19 = pitcher19total$WHIPearly # Prediction for 2019 WHIP
paste("MSE of WHIP 2019 by on pace:",
      round(mean((predWHIP19-pitcher19total$WHIPfull)^2,na.rm=T),3))# MSE for 2019 WHIP
```

```
## [1] "MSE of WHIP 2019 by on pace: 0.433"
```

```r
# Prediction and MSE for predictions based on "on pace" 2021
predWHIP21 = pitcher21total$WHIPearly # Prediction for 2021 WHIP
paste("MSE of WHIP 2021 by on pace:",
      round(mean((predWHIP21-pitcher21total$WHIPfull)^2,na.rm=T),3))# MSE for 2021 WHIP
```

```
## [1] "MSE of WHIP 2021 by on pace: 0.382"
```

Summary of Results and all MSEs:

Table 5: MSE summary from parts a, b, and c

Year	OPS.MSE..a.	OPS.MSE..b.	OPS.MSE..c.	WHIP.MSE..a.	WHIP.MSE..b.	WHIP.MSE..c.
2019	0.24	0.05	0.05	0.75	0.46	0.43
2020	0.08	0.05	NA	1.42	0.69	NA
2021	0.39	0.03	0.04	2.46	0.49	0.38

To summarize, the best predictions came from part (b) where we were able to use each players historical record to make their future predictions. While some of the MSE values for part (c) are lower, I would not find them as reliable as the variance in the predictions can be extraordinarily high. In a future analysis for more accurate predictions, we could use data for each season the player has played in the league, but give weights to each of the seasons. More recent seasons would be weighted higher than the first seasons, and this could help for player age if a player is going towards their peak or closer to retirement. If we wanted to dive even deeper, we could also look at how their hitting style has changed over time with more data. Maybe over time they are hitting more ground balls, pop flies, home runs; there are many options. Historical data is a good start to this analysis, but there are always more options on how to improve and finely tune our predictions.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
knitr::opts_chunk$set(fig.pos = "!h", fig.align = "center")
ggplot2::theme_set(ggplot2::theme_bw())
# Loading necessary libraries
library(tidyverse)
library(hoopR)
library(Lahman)
library(retrosheet)
library(baseballr)
library(lubridate)
library(kableExtra)
library(knitr)


# Get data from all games 2018-2021 from hoopR package
pbp = load_nba_pbp(2018:2021)
# Total number of free throw shots
(numftshots = pbp %>%
  filter(str_detect(type_text, 'Free Throw')) %>% # Filter data to only free throws
  summarise('Free Throw Shots' = n())) # Count number of rows of data (total free throws)
# Number of trips to the line
(numft = pbp %>%
  filter(str_detect(type_text, 'Free Throw'), # Filter to only free throws
         str_detect(type_text, '2 of 2|3 of 3'), # Filter to final shots
         !str_detect(type_text, 'Technical|Clear Path|Flagrant')) %>% # Take out techs
  summarise('Trips to line' = n())) # Count the total rows left
# Outputting our final updated coefficient
paste("Updated Coefficient: ", round(as.numeric(numft/numftshots), 3))


# Getting At bats and OPS for each batter from 2016 to 2018 (previous seasons)
batters1618 = bref_daily_batter("2016-01-01", "2018-12-31") %>%
  select(bbref_id, Name, G, AB, OPS) # Selecting only desired columns

# At bats and OPS for each batter through end of May 2019
batters19 = bref_daily_batter("2019-01-01", "2019-05-31") %>%
  select(bbref_id, Name, G, AB, OPS) # Selecting only desired columns

# At bats and OPS for each batter through end of 2019 season
battersFull19 = bref_daily_batter("2019-01-01", "2019-12-31") %>%
  select(bbref_id, Name, G, AB, OPS) # Selecting only desired columns

# At bats and OPS for each batter through end of 2020 season
battersFull20 = bref_daily_batter("2020-01-01", "2020-12-31") %>%
  select(bbref_id, Name, G, AB, OPS) # Selecting only desired columns

# At bats and OPS for each batter through end of May 2021
batters21 = bref_daily_batter("2021-01-01", "2021-05-31") %>%
  select(bbref_id, Name, G, AB, OPS) # Selecting only desired columns

# At bats and OPS for each batter through end of 2021 season
```

```r
battersFull21 = bref_daily_batter("2021-01-01", "2021-12-31") %>%
  select(bbref_id, Name, G, AB, OPS) # Selecting only desired columns

# Getting innings pitched and whip for each pitcher 2016 to 2021
pitchers1618 = bref_daily_pitcher("2016-01-01", "2018-12-31") %>%
  select(bbref_id, Name, G, IP, WHIP) # Selecting only desired columns

# At bats and OPS for each batter through end of May 2019
pitchers19 = bref_daily_pitcher("2019-01-01", "2019-05-31") %>%
  select(bbref_id, Name, G, IP, WHIP) # Selecting only desired columns

# At bats and OPS for each batter through end 2019 season
pitchersFull19 = bref_daily_pitcher("2019-01-01", "2019-12-31") %>%
  select(bbref_id, Name, G, IP, WHIP) # Selecting only desired columns

# At bats and OPS for each batter through end 2020 season
pitchersFull20 = bref_daily_pitcher("2020-01-01", "2020-12-31") %>%
  select(bbref_id, Name, G, IP, WHIP) # Selecting only desired columns

# At bats and OPS for each batter through end of May 2021
pitchers21 = bref_daily_pitcher("2021-01-01", "2021-05-31") %>%
  select(bbref_id, Name, G, IP, WHIP) # Selecting only desired columns

# At bats and OPS for each batter through end 2021 season
pitchersFull21 = bref_daily_pitcher("2021-01-01", "2021-12-31") %>%
  select(bbref_id, Name, G, IP, WHIP) # Selecting only desired columns


# Table for OPS header
kable(head(batters1618), caption = "OPS Historical Data (All games 2016-2018)")


# Table for WHIP header
kable(head(pitchers1618), caption = "WHIP Historical Data (All games 2016-2018)")


# Joining batter and pitcher datasets for 2019 and 2021
batter19total = left_join(batters19, battersFull19,
                          by = c("bbref_id", "Name")) %>% # join by name and reference
  rename(OPSearly = OPS.x, OPSfull = OPS.y) # rename columns for ease
batter21total = left_join(batters21, battersFull21,
                          by = c("bbref_id", "Name")) %>% # join by name and reference
  rename(OPSearly = OPS.x, OPSfull = OPS.y) # rename columns for ease

pitcher19total = left_join(pitchers19, pitchersFull19,
                          by = c("bbref_id", "Name")) %>% # join by name and reference
  rename(WHIPearly = WHIP.x, WHIPfull = WHIP.y) # rename columns for ease
pitcher21total = left_join(pitchers21, pitchersFull21,
                          by = c("bbref_id", "Name")) %>% # join by name and reference
  rename(WHIPearly = WHIP.x, WHIPfull = WHIP.y) # rename columns for ease


# Average player OPS
```

```r
avgOPS = mean(batters1618$OPS, na.rm = T)
paste("Average OPS 2016-2018: ", round(avgOPS, 3))
# Average player WHIP
avgWHIP = mean(pitchers1618$WHIP, na.rm = T)
paste("Average WHIP 2016-2018: ", round(avgWHIP, 3))
# Stein predictions OPS 2019 based on 2016-2018 mean player
c = 1 - (length(batters1618$OPS)-3)*var(batters1618$OPS, na.rm = T)/
  sum((batters19$OPS - avgOPS)^2, na.rm=T) # calculating c value
predictionsOPS19 = avgOPS + c*(batters19$OPS - avgOPS) # Getting stein estimates
paste("Mean of predictions for 2019 OPS: ",
      round(mean(predictionsOPS19, na.rm = T), 3)) # Mean of predictions for OPS
paste("Variance of predictions for 2019 OPS: ",
      round(var(predictionsOPS19, na.rm = T), 3)) # Variance of predictions for OPS
# Stein predictions WHIP 2019 based on 2016-2018 mean player
c = 1 - (length(pitchers1618$WHIP)-3)*var(pitchers1618$WHIP, na.rm = T)/
  sum((pitchers19$WHIP - avgWHIP)^2, na.rm=T) # Calculating c value
predictionsWHIP19 = avgWHIP + c*(pitchers19$WHIP - avgWHIP) # Getting stein estimates
paste("Mean of predictions for 2019 WHIP: ",
      round(mean(predictionsWHIP19, na.rm = T), 3)) # Mean of predictions for WHIP
paste("Variance of predictions for 2019 WHIP: ",
      round(var(predictionsWHIP19, na.rm = T), 3)) # Variance of predictions for WHIP
# MSE for OPS and WHIP
paste("MSE for OPS 2019: ",
      round(mean((predictionsOPS19 - batter19total$OPSfull)^2, na.rm=T), 3))
paste("MSE for WHIP 2019: ",
      round(mean((predictionsWHIP19 - pitcher19total$WHIPfull)^2, na.rm=T), 3))
# Overall mean (2016-2018) as prediction, MSE shown below for 2020
paste("MSE for OPS 2020: ",
      round(mean((avgOPS - battersFull20$OPS)^2, na.rm=T), 3))
paste("MSE for WHIP 2020: ",
      round(mean((avgWHIP - pitchersFull20$WHIP)^2, na.rm=T), 3))
# Stein predictions OPS 2021 based on 2016-2018 mean player
c = 1 - (length(batters1618$OPS)-3)*var(batters1618$OPS, na.rm = T)/
  sum((batters21$OPS - avgOPS)^2, na.rm=T) # calculating c value
predictionsOPS21 = avgOPS + c*(batters21$OPS - avgOPS) # Getting stein estimates
paste("Mean of OPS 2021 predictions: ",
      round(mean(predictionsOPS21, na.rm = T), 3)) # Mean of predictions for OPS
paste("Variance of OPS 2021 predictions: ",
      round(var(predictionsOPS21, na.rm = T),3)) # Variance of predictions for OPS


# Stein predictions WHIP 2021 based on 2016-2018 mean player
c = 1 - (length(batters1618$OPS)-3)*var(pitchers1618$WHIP, na.rm = T)/
  sum((pitchers21$WHIP - avgWHIP)^2, na.rm=T) # Calculating c value
predictionsWHIP21 = avgWHIP + c*(pitchers21$WHIP - avgWHIP) # Getting stein estimates
paste("Mean of WHIP 2021 predictions: ",
      round(mean(predictionsWHIP21, na.rm = T), 3)) # Mean of predictions for WHIP
paste("Variance of WHIP 2021 predictions: ",
      round(var(predictionsWHIP21, na.rm = T),3)) # Variance of predictions for WHIP


# MSE for OPS and WHIP 2021
paste("MSE of OPS 2021 predictions: ",
```

```
      round(mean((predictionsOPS21 - batter21total$OPSfull)^2, na.rm=T),3))
paste("MSE of WHIP 2021 predictions: ",
      round(mean((predictionsWHIP21 - pitcher21total$WHIPfull)^2, na.rm=T),3))


# Summary table for OPS and WHIP from overall mean as prior
mses <- data.frame(Year = c(2019, 2020, 2021),
                   'OPS MSE' = c(0.24,0.08, 0.39),
                   'WHIP MSE' = c(0.75,1.42, 2.46))
kable(mses, caption="MSEs using overall mean prior")


# Join all datasets to only include players playing in all of desired seasons
bat19 = inner_join(batters1618, batter19total, by = c("bbref_id", "Name"))
bat20 = inner_join(batters1618, battersFull20, by = c("bbref_id", "Name"))
bat21 = inner_join(batters1618, batter21total, by = c("bbref_id", "Name"))

pitch19 = inner_join(pitchers1618, pitcher19total, by = c("bbref_id", "Name"))
pitch20 = inner_join(pitchers1618, pitchersFull20, by = c("bbref_id", "Name"))
pitch21 = inner_join(pitchers1618, pitcher21total, by = c("bbref_id", "Name"))


# Stein predictions for OPS 2019 player specific
pbar19 = bat19$OPS # P bar for stein (historical data)
phat19 = bat19$OPSearly # P hat for stein (current season data)
c = 1 - (1 - 3)*var(batters1618$OPS, na.rm = T)/
  sum((phat19 - pbar19)^2,na.rm = T) # c value for stein
preds19 = pbar19 + c*(phat19-pbar19) # predictions

# Calculating our MSE for 2019
paste("MSE of OPS 2019 predictions:",
      round(mean((preds19 - bat19$OPSfull)^2, na.rm = T),3))
preds20 = bat20$OPS.x # Setting predictions to historical averages
paste("MSE of OPS 2020 predictions:",
      round(mean((preds20 - bat20$OPS.y)^2,na.rm = T),3))# Calculating our MSE for 2020
# Stein predictions for OPS 2021, same as 2019
pbar21 = bat21$OPS # P bar value (historical data)
phat21 = bat21$OPSearly # p hat value (current season data)
c = 1 - (1 - 3)*var(batters1618$OPS, na.rm = T)/
  sum((phat21 - pbar21)^2,na.rm = T) # c value
preds21 = pbar21 + c*(phat21-pbar21) # predictions

# Calculating our MSE for 2021
paste("MSE of OPS 2021 predictions:",
      round(mean((preds21 - bat21$OPSfull)^2, na.rm = T),3))


# Stein estimates for WHIP 2019 based on player specific info
pbar19 = pitch19$WHIP # P bar value (historical values)
phat19 = pitch19$WHIPearly # P hat value (current season data)
c = 1 - (1 - 3)*var(pitch19$WHIP, na.rm = T)/
  sum((phat19 - pbar19)^2,na.rm = T) # c value
preds19 = pbar19 + c*(phat19-pbar19) # WHIP predictions 2019
```

```r
# Calculating our MSE for 2019
paste("MSE of WHIP 2019 predictions:",
      round(mean((preds19 - pitch19$WHIPfull)^2, na.rm = T),3))
# MSE calculation for 2020 based on player previous averages
preds20 = pitch20$WHIP.x # Setting predictions to historical averages
paste("MSE of WHIP 2020 predictions:",
      round(mean((preds20-pitch20$WHIP.y)^2,na.rm = T),3))# Calculating our MSE for 2020
# Stein estimates for WHIP 2021 based on player specific info
pbar21 = pitch21$WHIP # P bar from historical data
phat21 = pitch21$WHIPearly # p hat from early season data
c = 1 - (1 - 3)*var(pitch21$WHIP, na.rm = T)/
  sum((phat21 - pbar21)^2,na.rm = T) # c value
preds21 = pbar21 + c*(phat21-pbar21) # Predictions WHIP 2021

# Calculating our MSE for 2021
paste("MSE of WHIP 2021 predictions: ",
      round(mean((preds21 - pitch21$WHIPfull)^2, na.rm = T),3))


# Table to summarise MSE using player specific priors
mses <- data.frame(Year = c(2019, 2020, 2021),
                   'OPS MSE' = c(0.05,0.05, 0.02),
                   'WHIP MSE' = c(0.46,0.69, 0.49))
kable(mses, caption="MSEs using player specific priors")


# Prediction and MSE for predictions based on "on pace" 2019
predOPS19 = batter19total$OPSearly # Prediction for 2019 OPS
paste("MSE of OPS 2019 by on pace:",
      round(mean((predOPS19 - batter19total$OPSfull)^2,na.rm=T),3))# MSE for 2019 OPS
# Prediction and MSE for predictions based on "on pace" 2021
predOPS21 = batter21total$OPSearly # Prediction for 2021 OPS
paste("MSE of OPS 2021 by on pace:",
      round(mean((predOPS21 - batter21total$OPSfull)^2,na.rm=T),3))# MSE for 2021 OPS
# Prediction and MSE for predictions based on "on pace" 2019
predWHIP19 = pitcher19total$WHIPearly # Prediction for 2019 WHIP
paste("MSE of WHIP 2019 by on pace:",
      round(mean((predWHIP19-pitcher19total$WHIPfull)^2,na.rm=T),3))# MSE for 2019 WHIP
# Prediction and MSE for predictions based on "on pace" 2021
predWHIP21 = pitcher21total$WHIPearly # Prediction for 2021 WHIP
paste("MSE of WHIP 2021 by on pace:",
      round(mean((predWHIP21-pitcher21total$WHIPfull)^2,na.rm=T),3))# MSE for 2021 WHIP
# Table to summarize all results
mses <- data.frame(Year = c(2019, 2020, 2021),
                   'OPS MSE (a)' = c(0.24,0.08, 0.39),
                   'OPS MSE (b)' = c(0.05,0.05, 0.03),
                   'OPS MSE (c)' = c(0.05, NA, 0.04),
                   'WHIP MSE (a)' = c(0.75,1.42, 2.46),
                   'WHIP MSE (b)' = c(0.46,0.69, 0.49),
                   'WHIP MSE (c)' = c(0.43,NA, 0.38))
kable(mses, caption="MSE summary from parts a, b, and c")
```