Time Series Baseball Batting Analysis

SYS 5581 Time-Series & Forecasting

Andres Izquierdo

3/14/2022

Abstract

This project will be using Time-Series analysis to forecast the statistical performance of batters based on their previous performance. This report will be tracking and forecasting batters OPS (On-base-plus slugging).

Introduction

Every year 30 MLB teams spend millions of dollars on their Research & Development (R&D) departments to try and use analytics to guide their decisions on which players to draft, re-sign, cut from the team, and sign from free agency all in an effort to win the ultimate prize in baseball, a World Series Championship. Teams look to bolster their chances of winning by looking at players and trying to decide, predict, if they would be a good addition to the team offensively as they then sign these promising player to multi-year million dollar contracts that if they don't live up to expectations may be the equivalent to burning a huge pile of money. Many teams use forecasting methods as ways to see if batters are following an upward or downward trajectory. This Time-Series project will look at using player historical data to predict their performance for upcoming seasons.

The data and the data-generating process

The raw data I will be analyzing in this project is batting statistics for baseball players collected from Major League Baseball (MLB) games. The baseball data set is Lahman's Baseball Database which has baseball statistics going all the way back to the 1800s. I will mainly be focusing on batting statistics from the year 1955 to 2020.

Out of the statistics in this dataset I will calculate On-base-plus slugging (adds the hitter's on base percentage (number of times reached base—by any means—divided by total plate appearances) to their slugging percentage (total bases divided by at bats)) and use this value as my predictor in the model.

The batting data is generated by at bats taken by the batter and their performance during that at bat. Whether they strikeout, get on base, score, etc. The outcome of the batter performance also depends on the performance of the pitcher during that at bat, this interaction will be approximated as a stochastic process where we will be predicting the future batting averages and OPS of these players. OPS is the most useful indicator in determining the overall performance of a batter.

This analysis will be done on statistics of yearly frequencies to forecast batting statistics for future years.

No transformation will be done on the data.

Exploratory data analysis

```r
# Data cleaning and set up.
library(dplyr)
```

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

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

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

```r
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(tsibble)
```

```
##
## Attaching package: 'tsibble'

## The following object is masked from 'package:lubridate':
##
##     interval

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

```r
library(stats)
library(fpp3)
```

```
## -- Attaching packages -------------------------------------------- fpp3 0.4.0 --

## v tibble      3.1.6     v feasts      0.2.2
## v tidyr       1.1.4     v fable       0.3.1
## v tsibbledata 0.4.0
## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

Batting Symbols and Definitions

playerID Player ID code yearID Year stint player's stint (order of appearances within a season) teamID Team lgID League G Games AB At Bats R Runs H Hits 2B Doubles 3B Triples HR Homeruns RBI Runs Batted In SB Stolen Bases CS Caught Stealing BB Base on Balls SO Strikeouts IBB Intentional walks HBP Hit by pitch SH Sacrifice hits SF Sacrifice flies GIDP Grounded into double plays

```r
#Locating file
path <- here::here("Lahman-master", "data", "Batting.RData")

load(path)

# Setting starting year as 1920 beginning of the live-ball era.
Start_Year = 1953

# removing missing statistics
#Batting.clean <- na.omit(Batting)

Batting$stint <- 1
names(Batting)[3] <- 'Seasons'

# Combining stats for players that were on different teams during the same year.
Batting.merge <- group_by(Batting, playerID, yearID) %>%
  mutate(G = sum(G),
         AB = sum(AB),
         R = sum(R),
         H = sum(H),
         X2B = sum(X2B),
         X3B = sum(X3B),
         HR = sum(HR),
         RBI = sum(RBI),
         SB = sum(SB),
         CS = sum(CS),
         BB = sum(BB),
         SO = sum(SO),
         IBB = sum(IBB),
         HBP = mean(HBP),
         SH = sum(SH),
         SF = sum(SF),
         GIDP = sum(GIDP))  %>%
  ungroup()

# Getting Rid of IBB Column as we do not need it for analysis.
```

```r
Batting.merge <- subset(Batting.merge, select = -c(IBB))

# Setting UP Batting Average Statistic
Batting.merge <- Batting.merge %>% mutate(Bavg = H/AB)

# Replacing NaN for all the times a batter got 0 hits and 0 at bats, can't divide 0/0, so replacing wit
Batting.merge[is.na(Batting.merge)] = 0

# Setting UP Slugging Percentage Statistic
Batting.merge <- Batting.merge %>% mutate(SLG = (H+X2B*2+X3B*3+HR*4)/AB)

# Setting UP On Base Percentage Statistic
Batting.merge <- Batting.merge %>% mutate(OBP = (H+BB+HBP)/(AB+BB+HBP+SF))

# Setting UP Slugging Percentage Statistic
Batting.merge <- Batting.merge %>% mutate(OPS = (OBP+SLG))

# removing missing statistics after OPS, SLG, and OPS calculations
Batting.merge <- na.omit(Batting.merge)

# Getting rid of duplicates
Batting.merge <- Batting.merge [!duplicated(Batting.merge[c(1,2)]),]


Batting.merge <- group_by(Batting.merge, playerID) %>%
  mutate(Seasons = sum(Seasons))

# keeping statistics from 1955 on as that is when all statistics started to be tracked.
Batting.merge <- Batting.merge %>% filter(yearID >= Start_Year)
Batting.merge %>%
  mutate(yearID = lubridate::as_date(yearID)) %>%
  mutate(playerID = as.factor(playerID)) %>%
  mutate(yearID   = as_date(yearID)) %>%
  as_tsibble(key = playerID, index = yearID)
```

```
## # A tsibble: 52,434 x 25 [1D]
## # Key:       playerID [9,896]
## # Groups:    playerID [9,896]
##    playerID  yearID     Seasons teamID lgID      G    AB     R     H   X2B   X3B
##    <fct>     <date>       <dbl> <fct>  <fct> <int> <int> <int> <int> <int> <int>
##  1 aardsda01 1975-06-30       3 CHN    NL       45     2     0     0     0     0
##  2 aardsda01 1975-07-02       3 BOS    AL       47     1     0     0     0     0
##  3 aardsda01 1975-07-09       3 ATL    NL       33     1     0     0     0     0
##  4 aaronha01 1975-05-09      23 ML1    NL      122   468    58   131    27     6
##  5 aaronha01 1975-05-10      23 ML1    NL      153   602   105   189    37     9
##  6 aaronha01 1975-05-11      23 ML1    NL      153   609   106   200    34    14
##  7 aaronha01 1975-05-12      23 ML1    NL      151   615   118   198    27     6
##  8 aaronha01 1975-05-13      23 ML1    NL      153   601   109   196    34     4
##  9 aaronha01 1975-05-14      23 ML1    NL      154   629   116   223    46     7
## 10 aaronha01 1975-05-15      23 ML1    NL      153   590   102   172    20    11
## # ... with 52,424 more rows, and 14 more variables: HR <int>, RBI <int>,
## #   SB <int>, CS <int>, BB <int>, SO <int>, HBP <dbl>, SH <int>, SF <int>,
## #   GIDP <int>, Bavg <dbl>, SLG <dbl>, OBP <dbl>, OPS <dbl>
```

```r
Batting_tsbl <- as_tsibble(Batting.merge, key = playerID, index = yearID)
Batting_tsbl
```

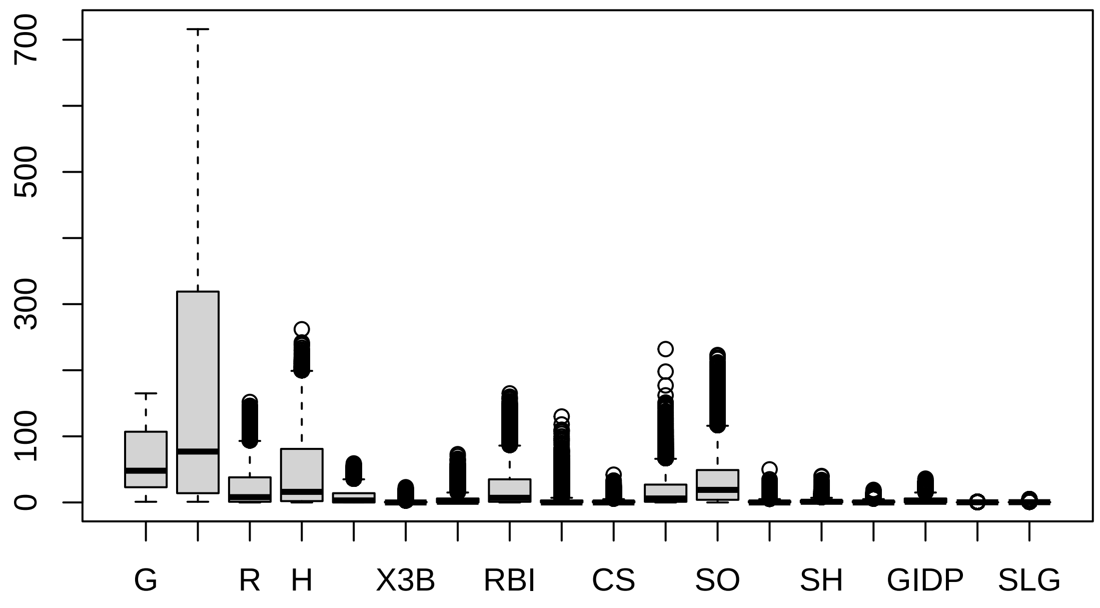
```
## # A tsibble: 52,434 x 25 [1Y]
## # Key:       playerID [9,896]
## # Groups:    playerID [9,896]
##    playerID  yearID Seasons teamID lgID      G    AB     R     H   X2B   X3B
##    <chr>      <int>   <dbl> <fct>  <fct> <int> <int> <int> <int> <int> <int>
##  1 aardsda01   2006       3 CHN    NL       45     2     0     0     0     0
##  2 aardsda01   2008       3 BOS    AL       47     1     0     0     0     0
##  3 aardsda01   2015       3 ATL    NL       33     1     0     0     0     0
##  4 aaronha01   1954      23 ML1    NL      122   468    58   131    27     6
##  5 aaronha01   1955      23 ML1    NL      153   602   105   189    37     9
##  6 aaronha01   1956      23 ML1    NL      153   609   106   200    34    14
##  7 aaronha01   1957      23 ML1    NL      151   615   118   198    27     6
##  8 aaronha01   1958      23 ML1    NL      153   601   109   196    34     4
##  9 aaronha01   1959      23 ML1    NL      154   629   116   223    46     7
## 10 aaronha01   1960      23 ML1    NL      153   590   102   172    20    11
## # ... with 52,424 more rows, and 14 more variables: HR <int>, RBI <int>,
## #   SB <int>, CS <int>, BB <int>, SO <int>, HBP <dbl>, SH <int>, SF <int>,
## #   GIDP <int>, Bavg <dbl>, SLG <dbl>, OBP <dbl>, OPS <dbl>
```

```r
# Filling the gaps in players careers so to not eliminate players who had gaps in their careers due to
Batting_tsbl <- fill_gaps(Batting_tsbl)
Batting_tsbl <- Batting_tsbl %>% mutate_at(c(3,6:25), ~replace_na(.,0))
```

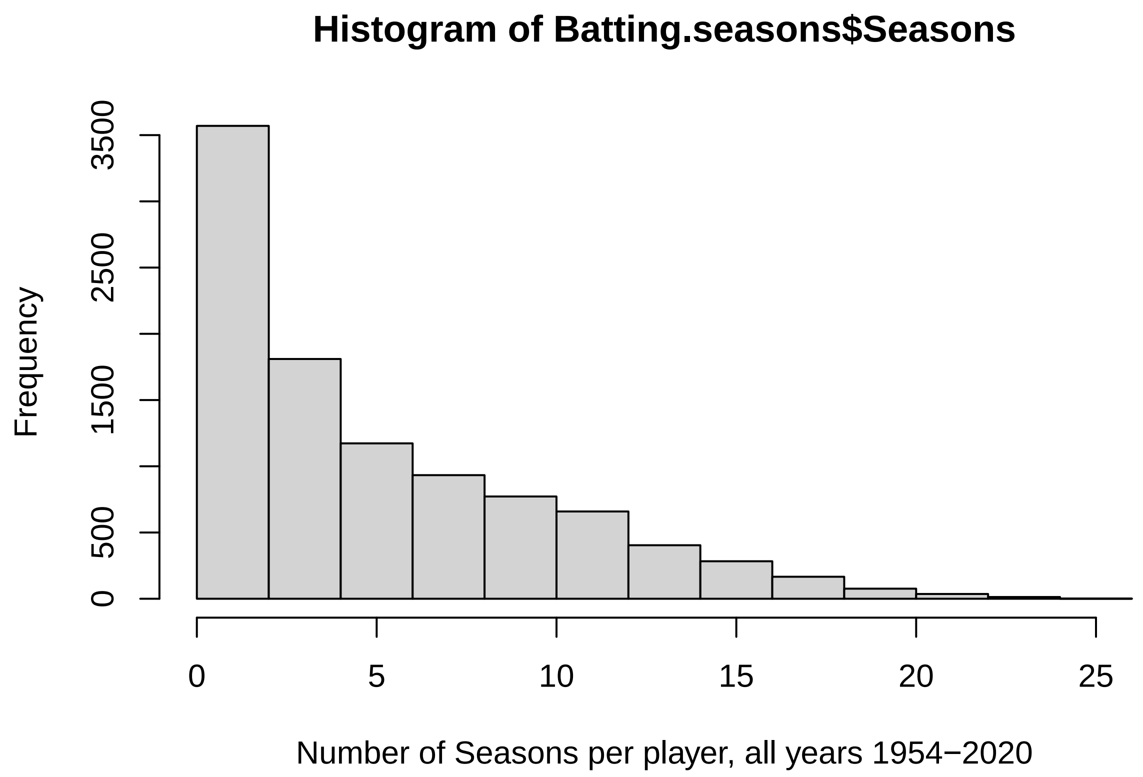
```r
Batting.seasons <- Batting.merge %>% distinct(playerID, .keep_all = TRUE)
Batting.num <- Batting.merge[,c(6:23)]
summary(Batting.num)
```

```
##        G                AB               R                H         
##  Min.   :  1.00   Min.   :  1.0   Min.   :  0.00   Min.   :  0.00  
##  1st Qu.: 23.00   1st Qu.: 14.0   1st Qu.:  1.00   1st Qu.:  2.00  
##  Median : 48.00   Median : 77.0   Median :  8.00   Median : 16.00  
##  Mean   : 64.04   Mean   :176.6   Mean   : 22.87   Mean   : 45.64  
##  3rd Qu.:107.00   3rd Qu.:319.0   3rd Qu.: 38.00   3rd Qu.: 81.00  
##  Max.   :165.00   Max.   :716.0   Max.   :152.00   Max.   :262.00  
##       X2B              X3B             HR              RBI        
##  Min.   : 0.000   Min.   : 0.0   Min.   : 0.000   Min.   :  0.00  
##  1st Qu.: 0.000   1st Qu.: 0.0   1st Qu.: 0.000   1st Qu.:  1.00  
##  Median : 3.000   Median : 0.0   Median : 1.000   Median :  7.00  
##  Mean   : 8.204   Mean   : 1.1   Mean   : 4.799   Mean   : 21.57  
##  3rd Qu.:14.000   3rd Qu.: 1.0   3rd Qu.: 6.000   3rd Qu.: 35.00  
##  Max.   :59.000   Max.   :23.0   Max.   :73.000   Max.   :165.00  
##        SB                CS              BB                SO        
##  Min.   :  0.000   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00  
##  1st Qu.:  0.000   1st Qu.: 0.00   1st Qu.:  1.00   1st Qu.:  4.00  
##  Median :  0.000   Median : 0.00   Median :  6.00   Median : 19.00  
##  Mean   :  3.085   Mean   : 1.47   Mean   : 16.96   Mean   : 31.73  
##  3rd Qu.:  3.000   3rd Qu.: 2.00   3rd Qu.: 27.00   3rd Qu.: 49.00  
##  Max.   :130.000   Max.   :42.00   Max.   :232.00   Max.   :223.00  
##       HBP               SH                SF               GIDP
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.000   Median : 1.000   Median : 0.000   Median : 1.000
##  Mean   : 1.333   Mean   : 1.911   Mean   : 1.418   Mean   : 3.959
##  3rd Qu.: 2.000   3rd Qu.: 3.000   3rd Qu.: 2.000   3rd Qu.: 6.000
##  Max.   :50.000   Max.   :40.000   Max.   :19.000   Max.   :36.000
##       Bavg             SLG
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1373   1st Qu.:0.1800
##  Median :0.2286   Median :0.3765
##  Mean   :0.2024   Mean   :0.3501
##  3rd Qu.:0.2700   3rd Qu.:0.4989
##  Max.   :1.0000   Max.   :5.0000
```

```
boxplot(Batting.num)
```

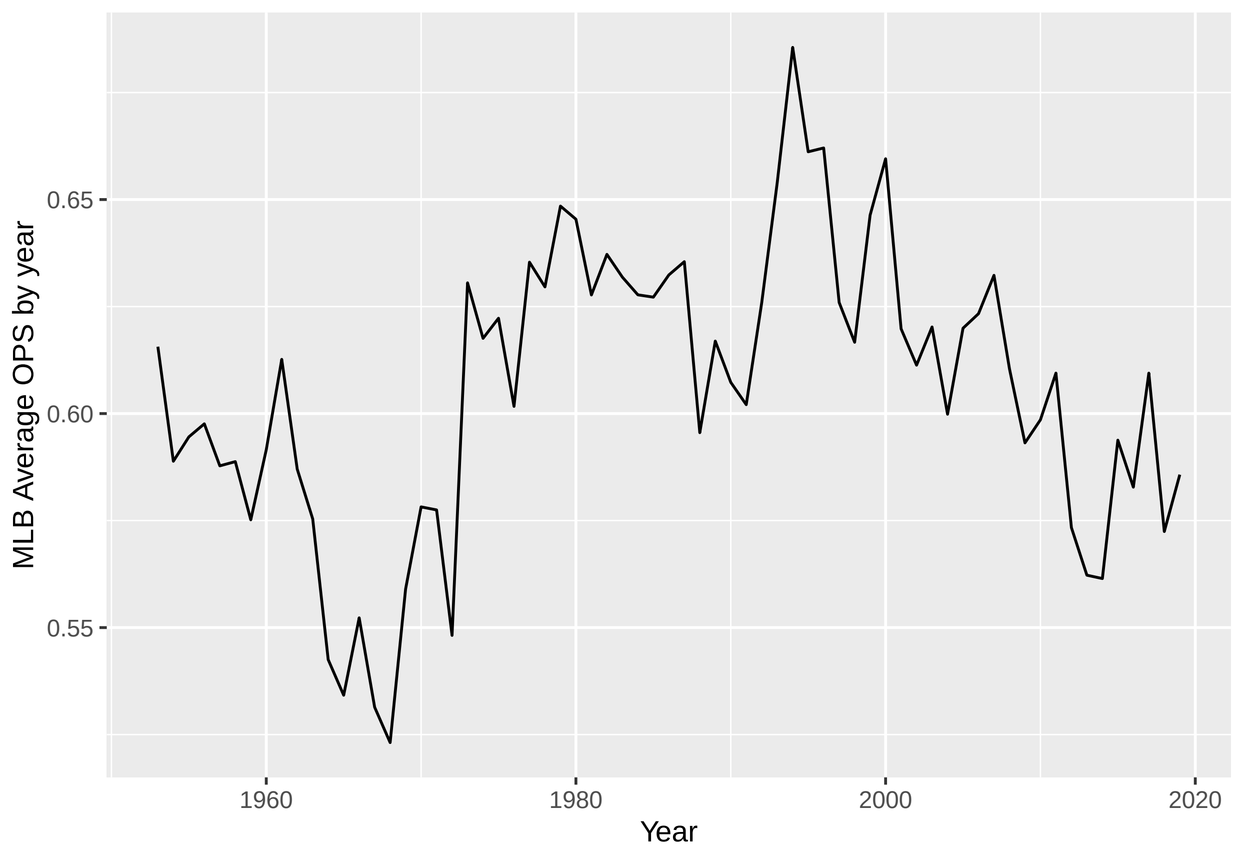


```
hist(Batting.seasons$Seasons, xlab = "Number of Seasons per player, all years 1954-2020")
```



```r
# Maybe do a histogram of number years played.
```

```r
Batting.merge %>%
  filter(yearID <= 2019) %>%
  group_by(yearID) %>%
  summarise(OPS = mean(OPS)) %>%
  ggplot(aes(x=yearID,y=OPS)) +
    geom_line() + xlab("Year") + ylab("MLB Average OPS by year")
```



Now when we look at the average OPS graph we don't really see a trend, we can see a peak in the 30s, late 90s, and early 2000s but there is no steady increase or decrease in the overall trend of the League OPS.

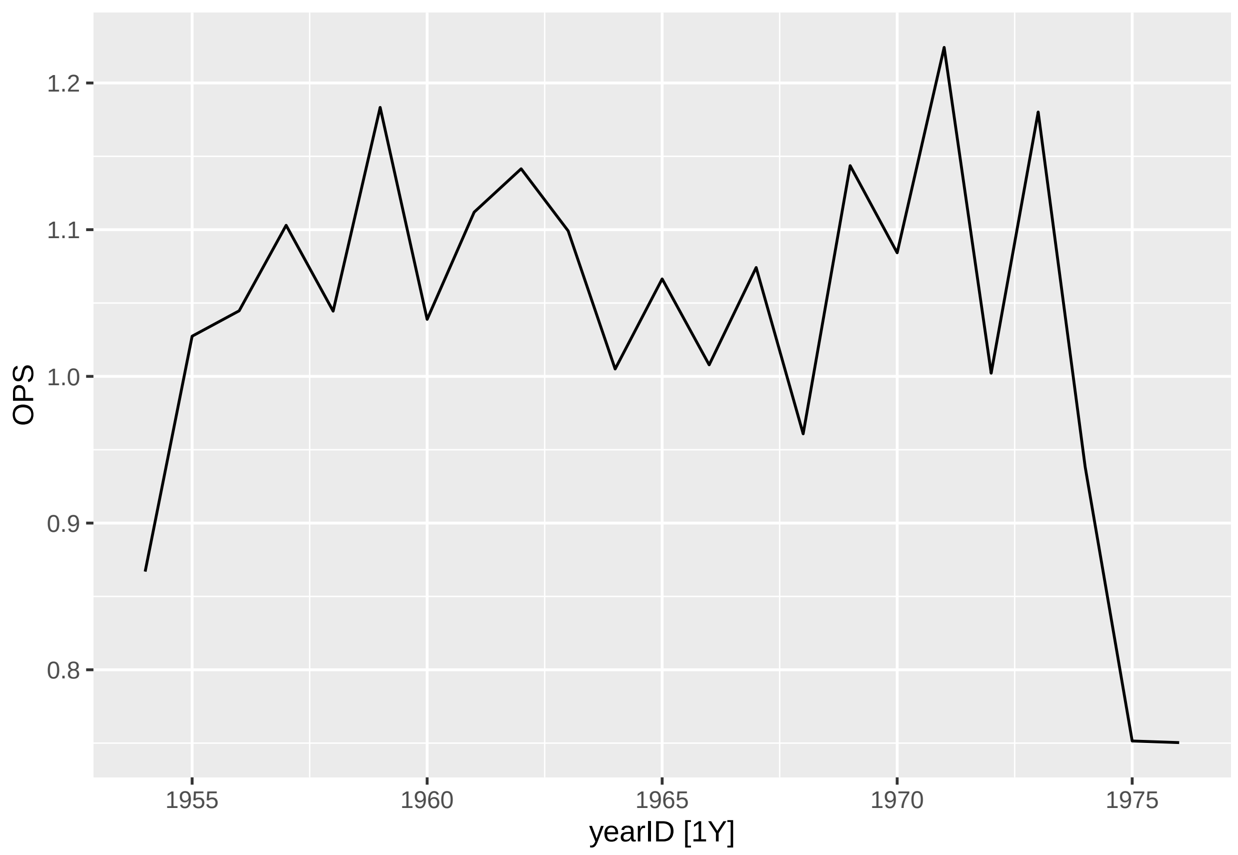
Correlation analysis

For the correlation analysis we will be looking at 5 individual players careers and the OPS numbers they posted for each season of their career. We will look at Hank Aaron, Ichiro Suzuki, Derek Jeter, David Otriz, Jayson Werth, and Anthony Rendon. These players range from having long careers, to careers where they did not play for a whole season due to injury, and careers that are still going.

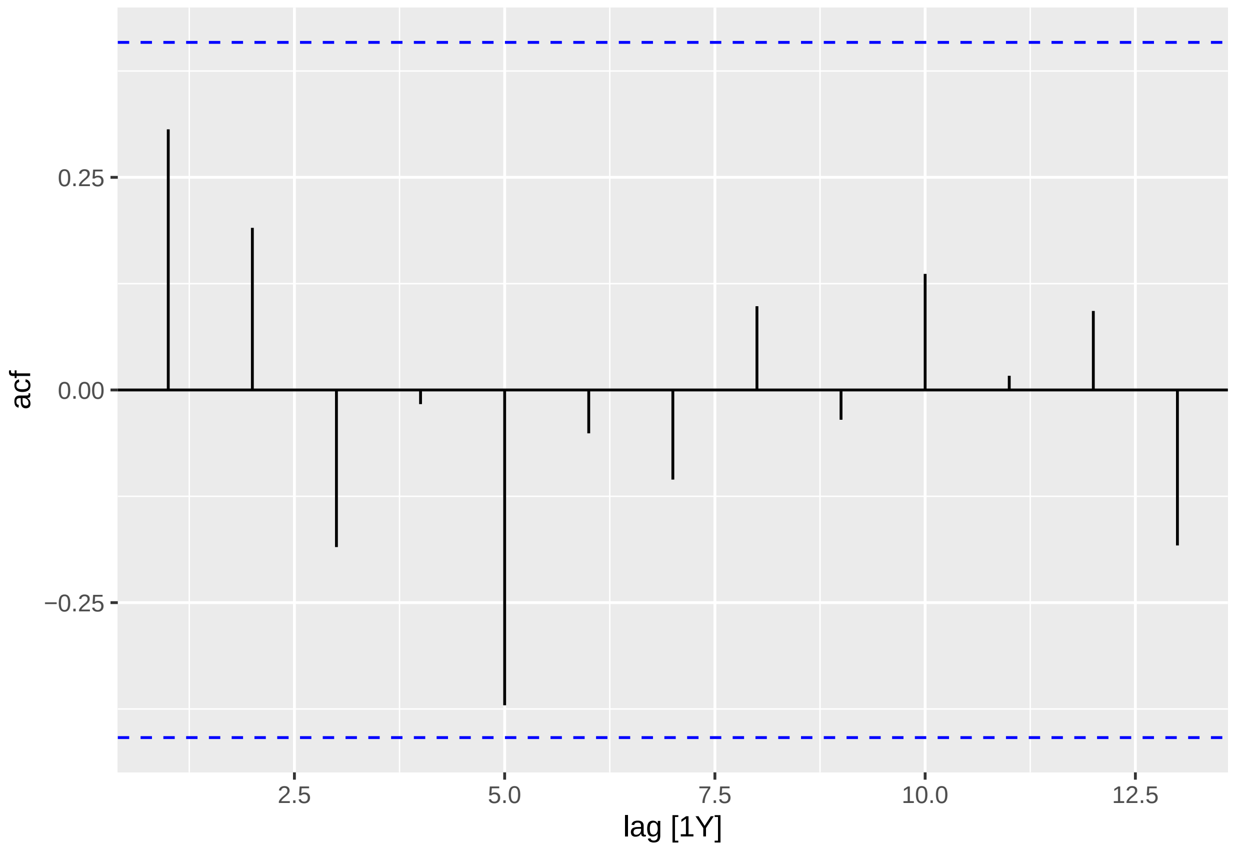
Hank Aaron OPS

```
# Hank Aaron OPS plot
Batting_tsbl %>%
  filter(playerID == "aaronha01") -> aaronha01_tsbl

aaronha01_tsbl %>% autoplot(OPS)
```



```r
# Hank Aaron's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "aaronha01") %>%
  ACF(OPS) %>%
  autoplot(main = "Hank Aaron OPS ACF")
```

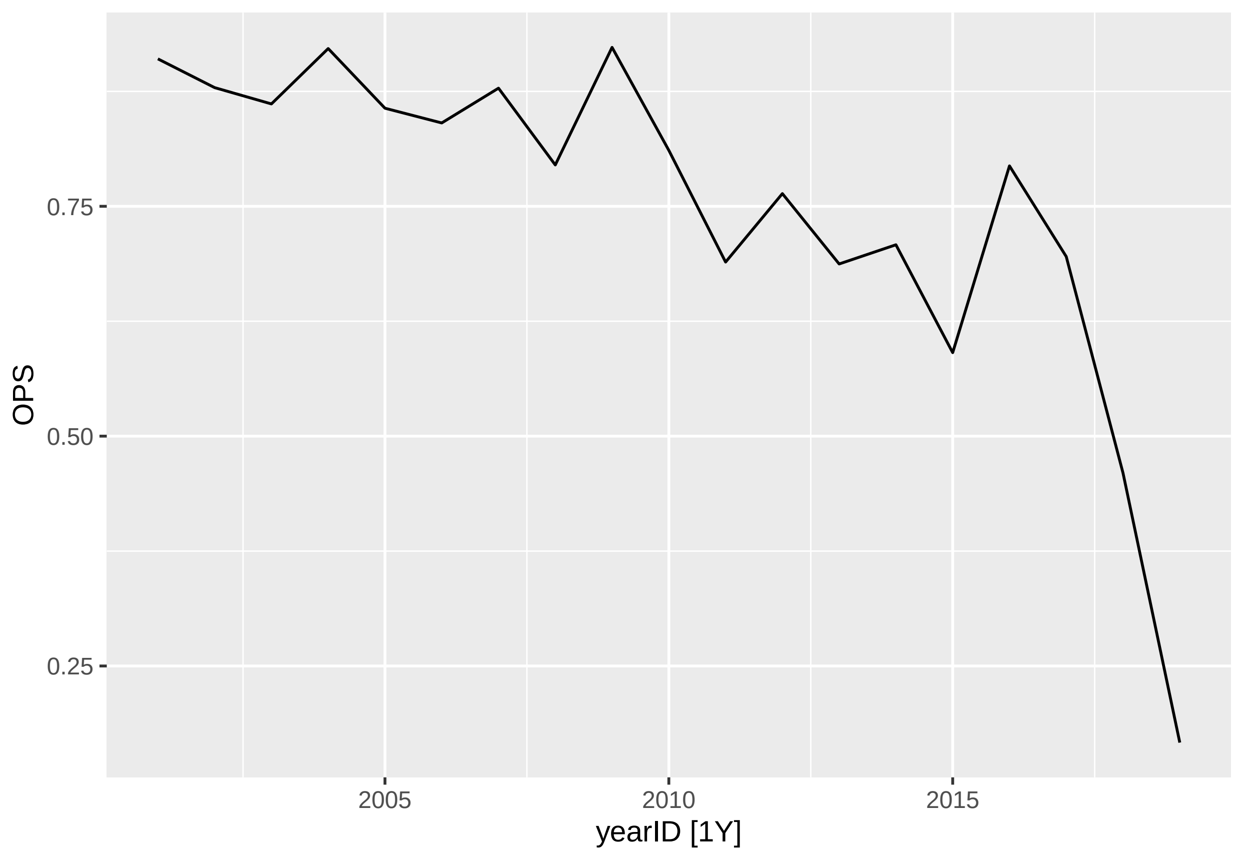


```
# Normalizing the number of Seasons
aaronha01_tsbl$Seasons <- 1:nrow(aaronha01_tsbl)
```

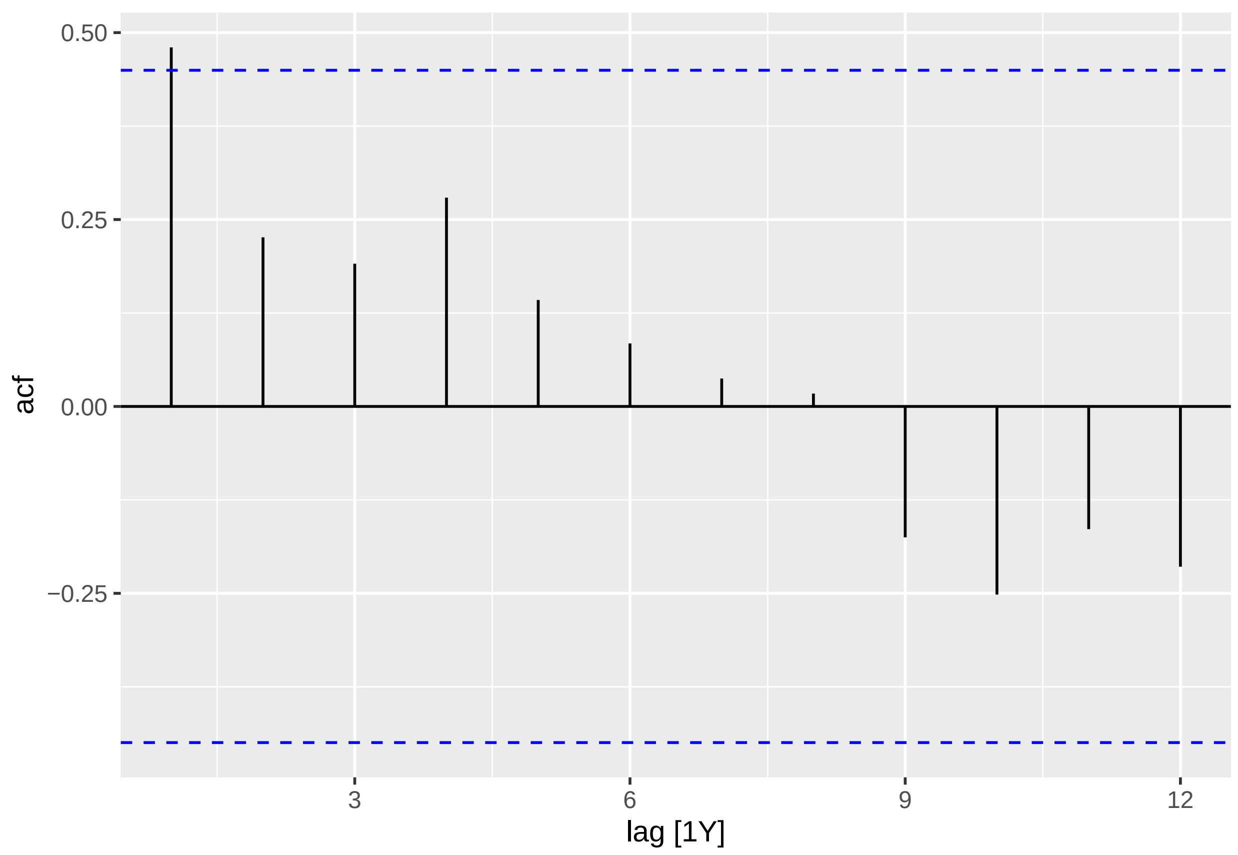
Ichiro Suzuki OPS

```
# Ichiro Suzuki's OPS plot
Batting_tsbl %>%
  filter(playerID == "suzukic01") -> suzukic01_tsbl

suzukic01_tsbl %>% autoplot(OPS)
```



```r
# Ichiro Suzuki's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "suzukic01") %>%
  ACF(OPS) %>%
  autoplot()
```

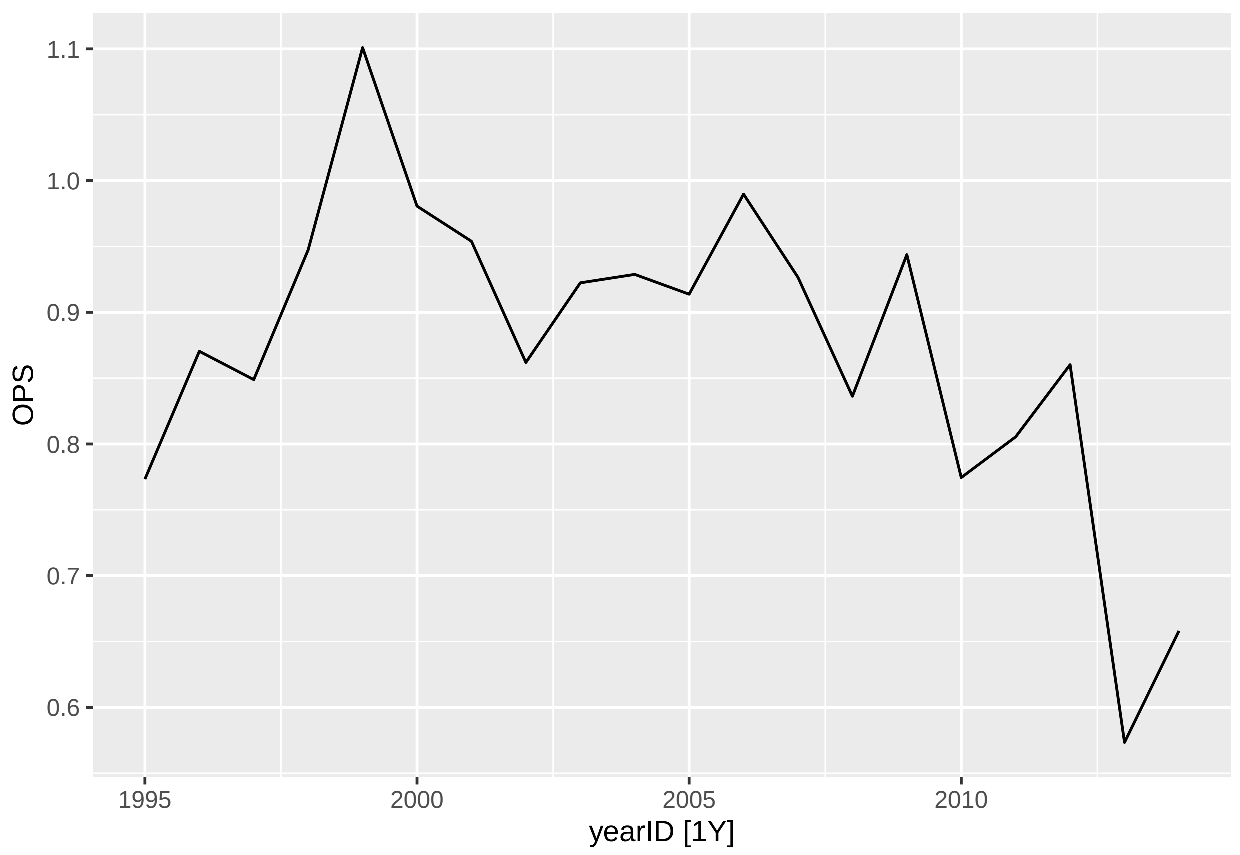


```r
# Normalizing the number of Seasons
suzukic01_tsbl$Seasons <- 1:nrow(suzukic01_tsbl)
```

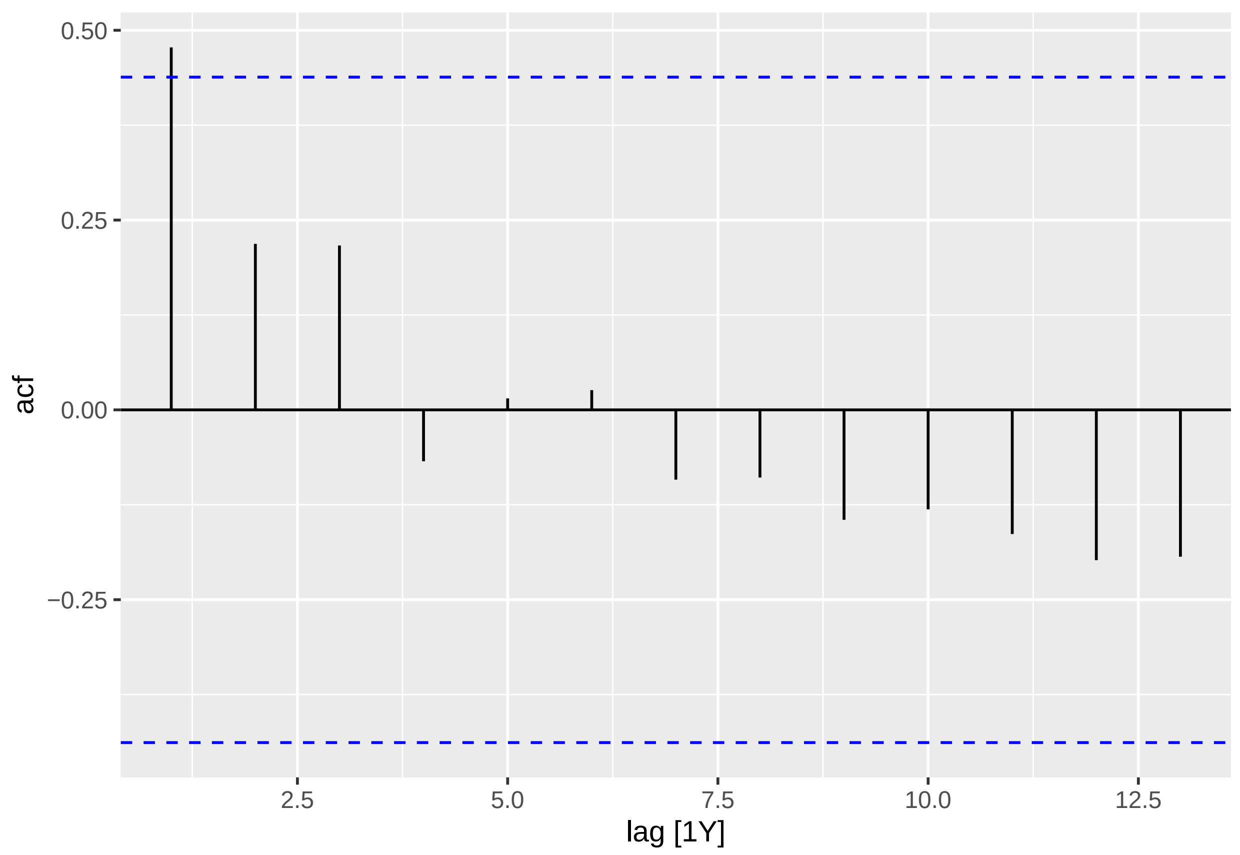
Derek Jeter OPS

```r
# Derek Jeter's OPS plot
Batting_tsbl %>%
  filter(playerID == "jeterde01") -> jeterde01_tsbl

jeterde01_tsbl %>% autoplot(OPS)
```



```
# Derek Jeter's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "jeterde01") %>%
  ACF(OPS) %>%
  autoplot()
```

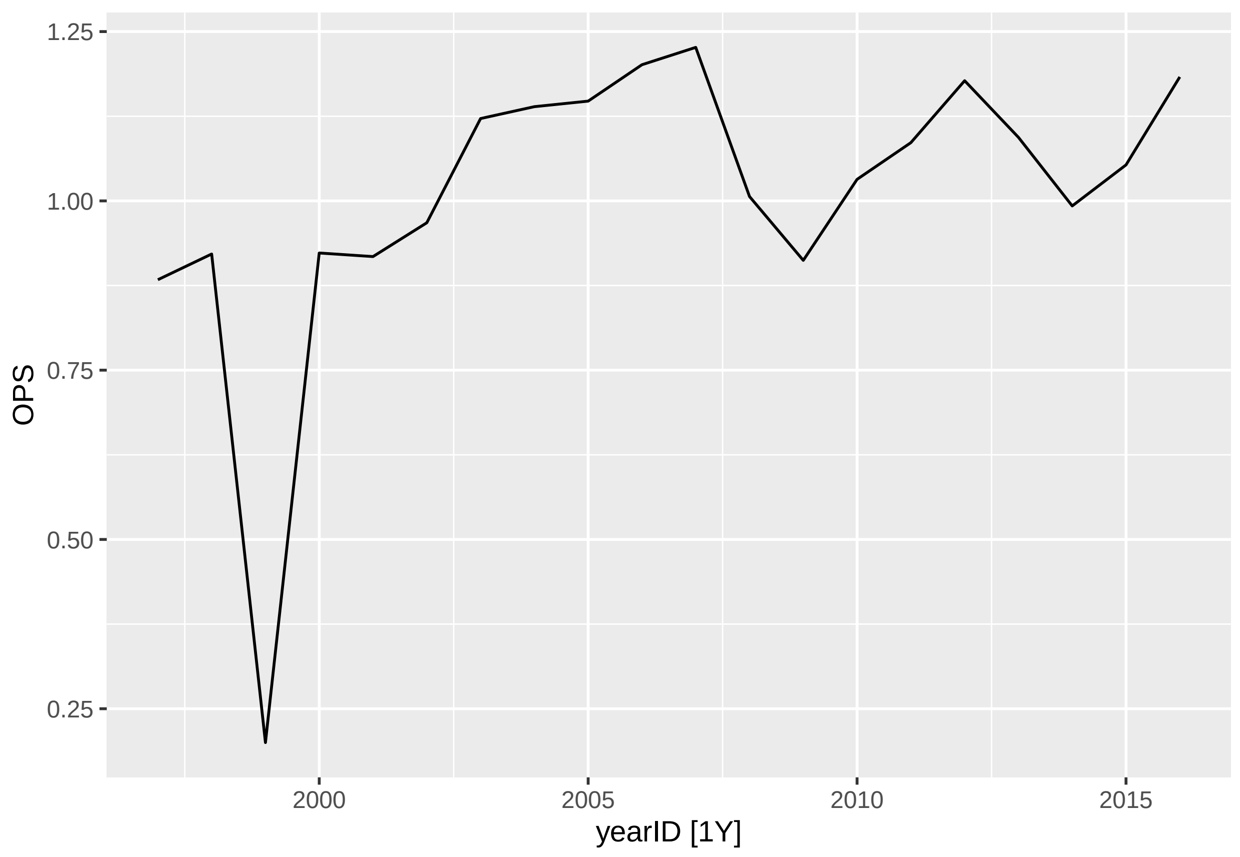


```r
# Normalizing the number of Seasons
jeterde01_tsbl$Seasons <- 1:nrow(jeterde01_tsbl)
```

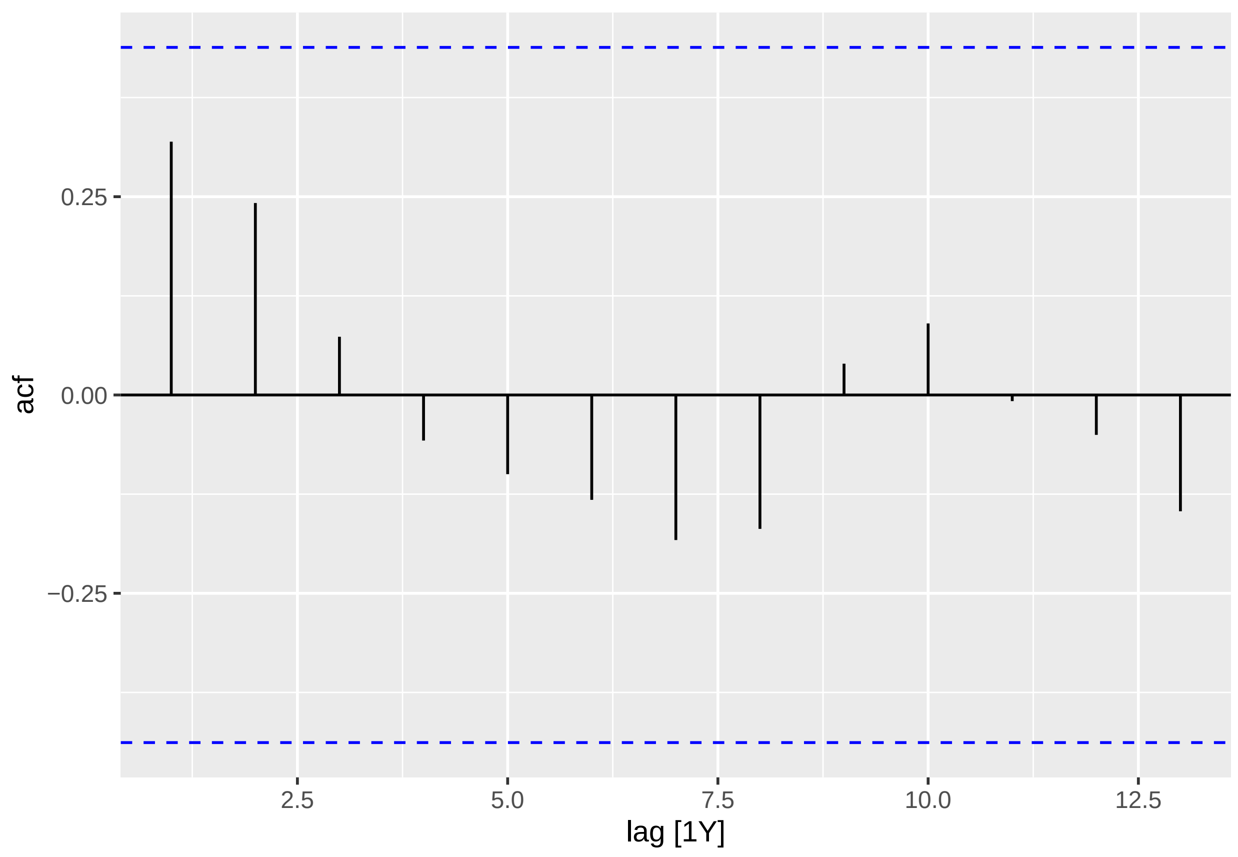
David Ortiz OPS

```r
# David Ortiz's OPS plot
Batting_tsbl %>%
  filter(playerID == "ortizda01") -> ortizda01_tsbl

ortizda01_tsbl %>% autoplot(OPS)
```



```
# David Ortiz's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "ortizda01") %>%
  ACF(OPS) %>%
  autoplot()
```

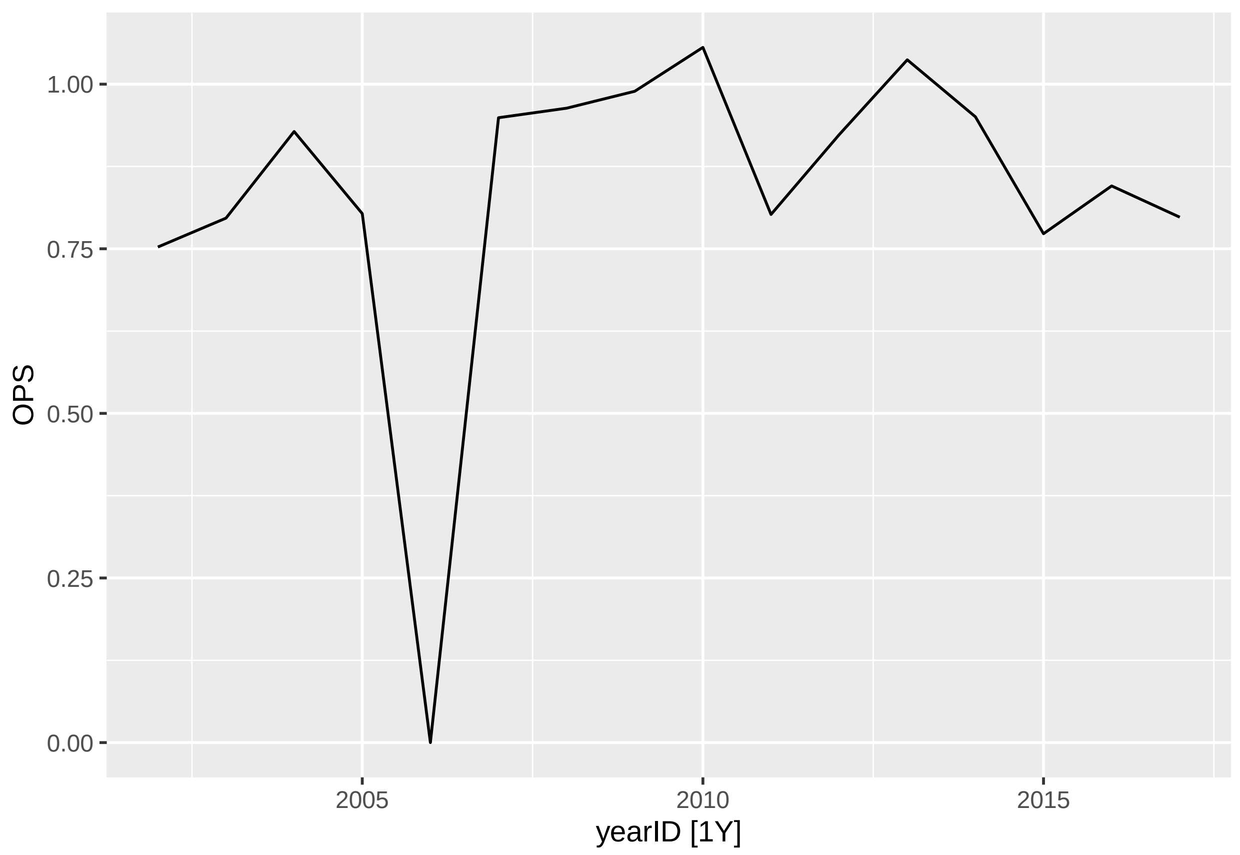


```
# Normalizing the number of Seasons
ortizda01_tsbl$Seasons <- 1:nrow(ortizda01_tsbl)
```

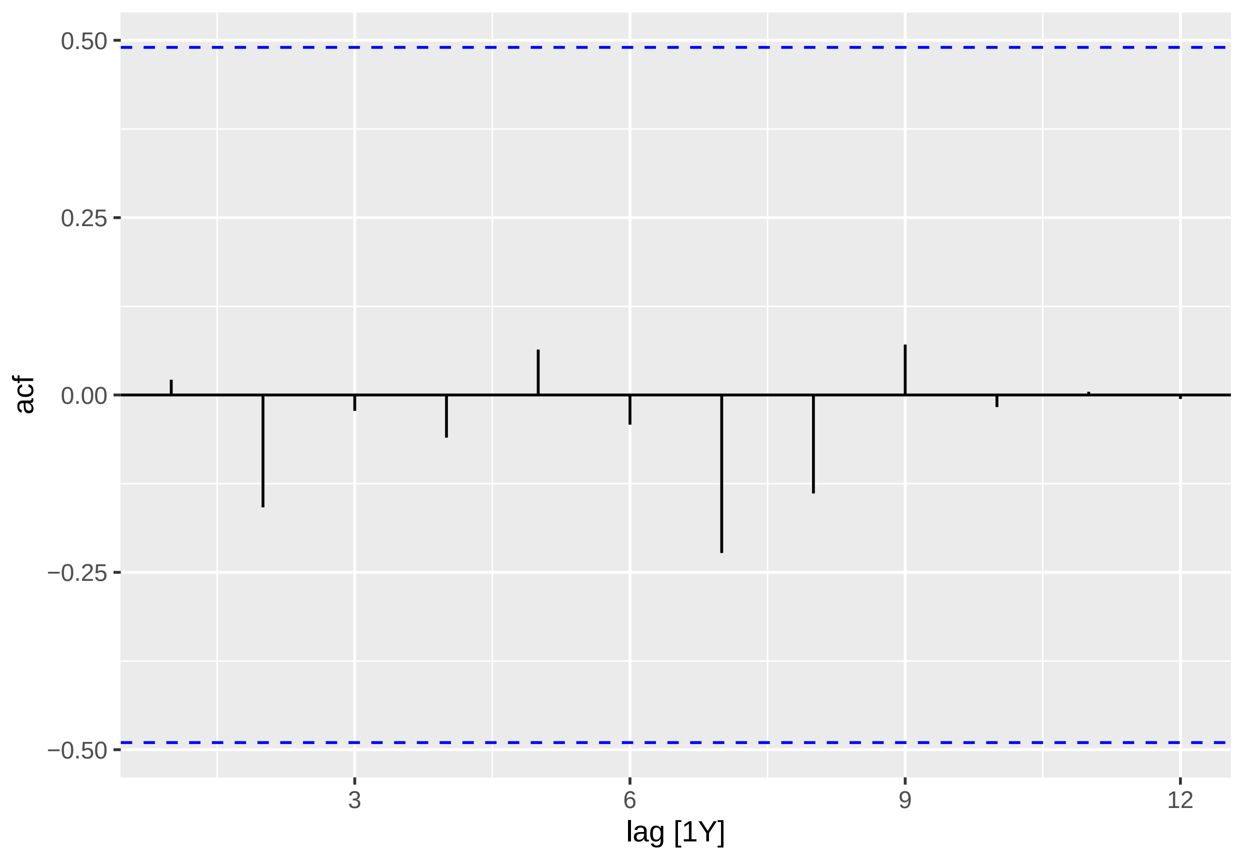
Jayson Werth OPS

```
# Jayson Werth's OPS plot
Batting_tsbl %>%
  filter(playerID == "werthja01") -> werthja01_tsbl

werthja01_tsbl %>% autoplot(OPS)
```



```
# Jayson Werth's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "werthja01") %>%
  ACF(OPS) %>%
  autoplot()
```

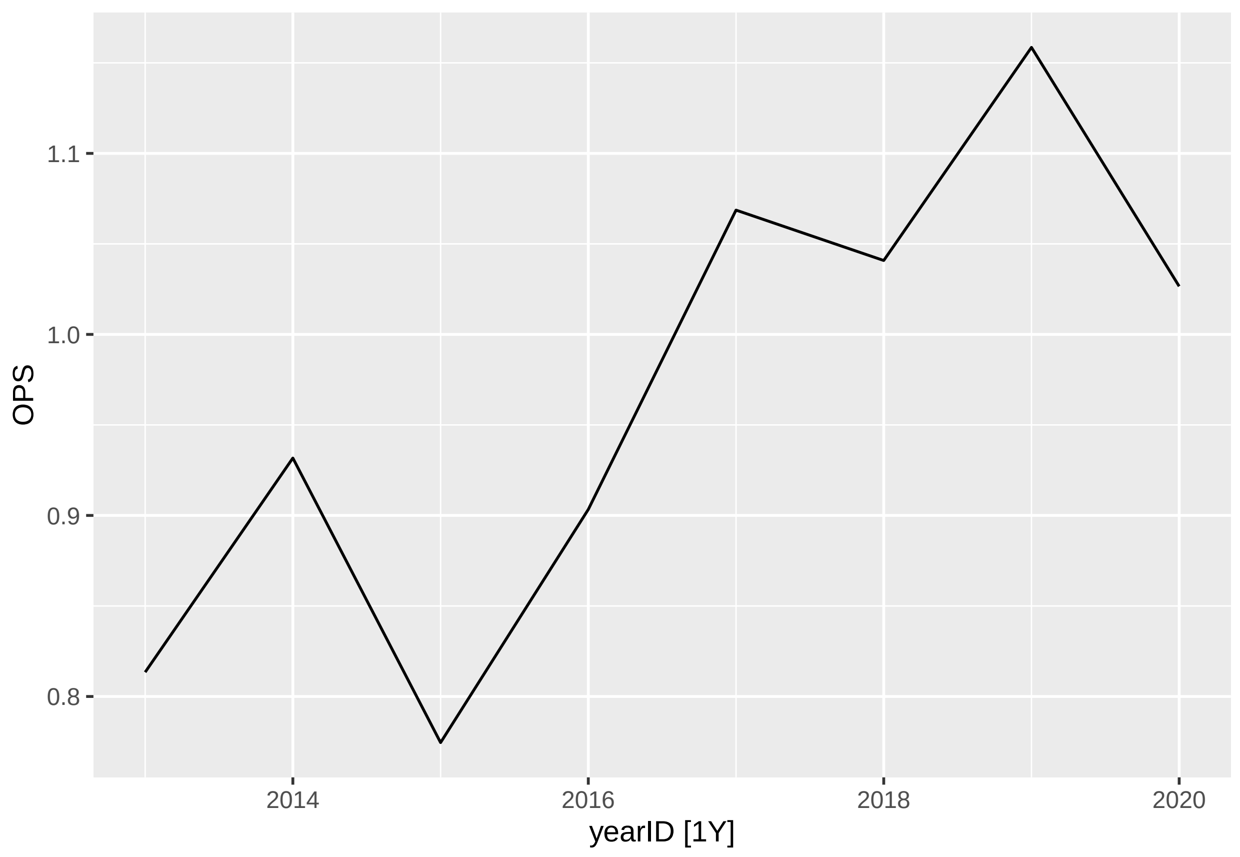


```r
# Normalizing the number of Seasons
werthja01_tsbl$Seasons <- 1:nrow(werthja01_tsbl)
```

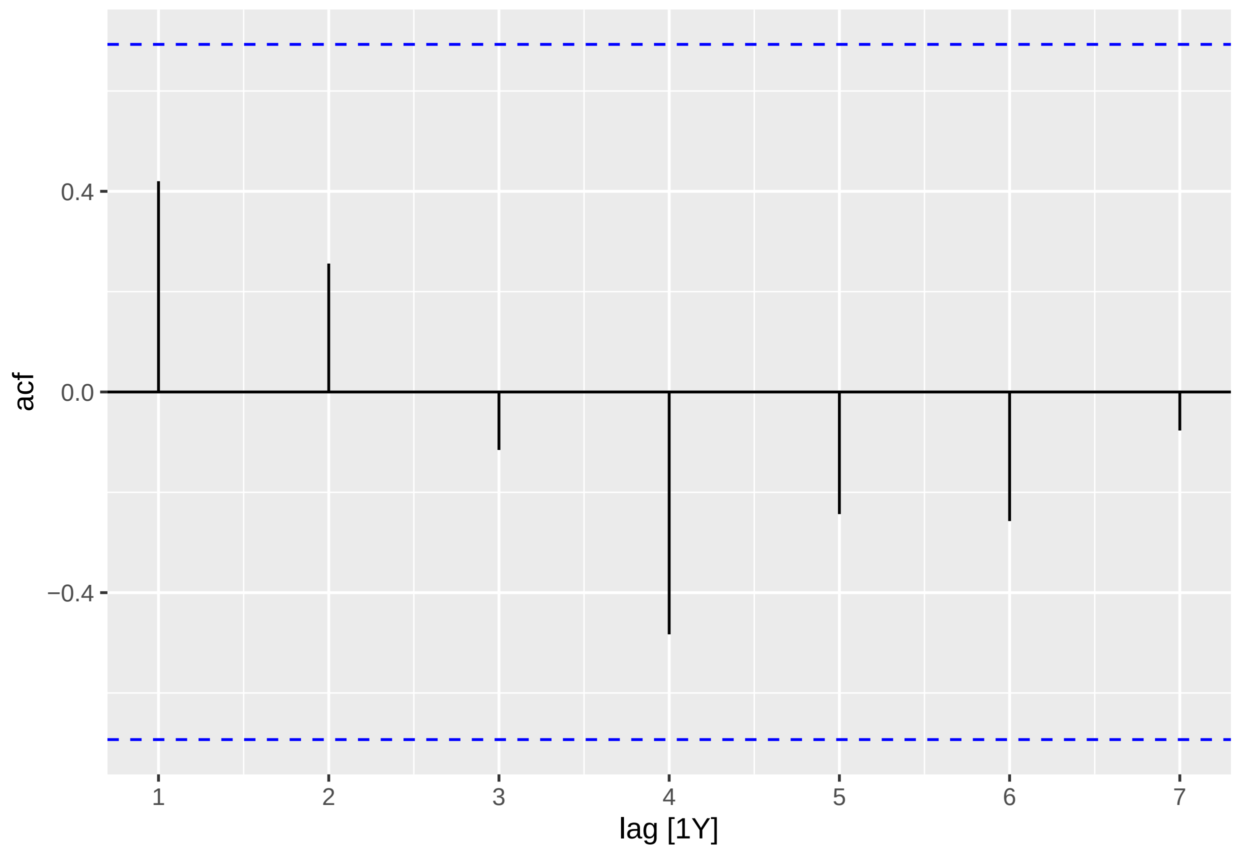
Anthony Rendon OPS

```r
# Anthony Rendon's OPS plot
Batting_tsbl %>%
  filter(playerID == "rendoan01") -> rendoan01_tsbl

rendoan01_tsbl %>% autoplot(OPS)
```



```
# Anthony Rendon's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "rendoan01") %>%
  ACF(OPS) %>%
  autoplot()
```



```
# Normalizing the number of Seasons
rendoan01_tsbl$Seasons <- 1:nrow(rendoan01_tsbl)
```

Looking at each players OPS ACF plots there seems to be a sudden spike at the first lag that seems to tail off as the lags increase. The majority of the lags stay well below the threshold levels

Formal model of data-generating process:

The OPS statistic does not have significant autocorrelation and will be modeled to follow an AR(1) process. Stationarity will be assumed for the data generating process.

The AR(1) model will be formulated by: $y_{t,i}\ _{,j} = \alpha y_{t-1} + \varepsilon$

Where Y t,i,j stands for: t: year in career (normalized). i: player i. j: stat j OPS.

There error is: $\varepsilon \sim N(0, \sigma^2)$

Discussion of the statistical model:

The model captures the data-generating process because it captures the information of the lagged predictors taking the information into account to make future predictions for future years. The error term will take all other factors that are not captured in the data such as pitching faced, etc.

Selected Analytic Approach

This project will be looking at yearly time intervals analyzing the performance of batters in order to forecast On-base-plus slugging.

Analyzing these two variables during yearly time intervals will be done by using lagged regression, AR(p).

Since this project is looking at hundreds of MLB players none with the exact same career length and performance the model will have to adapt to each player in order to determine the p in the AR and ARMA models to use their statistics from previous years to forecast the performance of future years. This will be done by using recent (past year) data and past (entire career previous to the most recent year) data, as training data, to make forecasts on the next couple of seasons, test data, and compare performances with the test data.

Analytic Approach Justification

I am using recent and past performance periods as batters tend to go on streaks in their careers that are unexpected. One year a batter may be the league MVP (Most Valuable Play) while the next year their performance drops significantly. Keeping a series of isolated data from the model I will then be able to use that series as test data to compare results.