Times Series Predictions On Batting OPS of Baseball Players

Andres Izquierdo

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. For this Project we will look at 6 former players careers and develop models that accurately predict their OPS for the last 20% of their careers and compare how they do using different models. The player we will use for our model development will be Hank Aaron, Ichiro Suzuki, Derek Jeter, David Ortiz, Pete Rose, and Barry Bonds. At the end of the project we will use the most optimal model to make predictions on young prospects and players in their middle years to predict how they will perform in future years.

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.

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

Bavg: Batting Average

SLG: Slugging

OBP: On-base Percentage

OPS: On-base Plus Slugging

Data Cleaning and Set Up

```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
```

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

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

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

```
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
```

```
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()
```

```r
library(glue)
library(fable)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

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

load(path)

# Setting starting year as 1953 beginning of the Modern era.
Start_Year = 1953

# Changin Stint column to Season column and counting all the seasons for each player
Batting$stint <- 1
names(Batting)[3] <- 'Season'

first_year <- Batting %>%
              group_by(playerID) %>%
              summarize(firstyear = as.numeric(min(yearID)))

Batting <- Batting %>% left_join(first_year, by="playerID")

Batting <- Batting %>% mutate(Season = yearID - firstyear + 1)

Batting <- Batting %>% filter(firstyear >= Start_Year)

Batting <- Batting[,-23]

# 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.
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)]),]

# Building tsibble
Batting.merge %>%
  mutate(Season = lubridate::as_date(Season)) %>%
  mutate(playerID = as.factor(playerID)) %>%
  mutate(Season   = as_date(Season)) %>%
  as_tsibble(key = playerID, index = Season)
```

```
## # A tsibble: 49,882 x 25 [1D]
## # Key:       playerID [9,382]
##    playerID  yearID Season     teamID lgID      G    AB     R     H   X2B   X3B
##    <fct>      <int> <date>     <fct>  <fct> <int> <int> <int> <int> <int> <int>
##  1 aardsda01   2006 1970-01-04 CHN    NL       45     2     0     0     0     0
##  2 aardsda01   2008 1970-01-06 BOS    AL       47     1     0     0     0     0
##  3 aardsda01   2015 1970-01-13 ATL    NL       33     1     0     0     0     0
##  4 aaronha01   1954 1970-01-02 ML1    NL      122   468    58   131    27     6
##  5 aaronha01   1955 1970-01-03 ML1    NL      153   602   105   189    37     9
##  6 aaronha01   1956 1970-01-04 ML1    NL      153   609   106   200    34    14
##  7 aaronha01   1957 1970-01-05 ML1    NL      151   615   118   198    27     6
##  8 aaronha01   1958 1970-01-06 ML1    NL      153   601   109   196    34     4
##  9 aaronha01   1959 1970-01-07 ML1    NL      154   629   116   223    46     7
## 10 aaronha01   1960 1970-01-08 ML1    NL      153   590   102   172    20    11
## # ... with 49,872 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>
```

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

```
## # A tsibble: 49,882 x 25 [1]
## # Key:       playerID [9,382]
##    playerID yearID Season teamID lgID      G    AB     R     H   X2B   X3B    HR
##    <chr>     <int>  <dbl> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
##  1 aardsda~   2006      3 CHN    NL       45     2     0     0     0     0     0
##  2 aardsda~   2008      5 BOS    AL       47     1     0     0     0     0     0
##  3 aardsda~   2015     12 ATL    NL       33     1     0     0     0     0     0
##  4 aaronha~   1954      1 ML1    NL      122   468    58   131    27     6    13
##  5 aaronha~   1955      2 ML1    NL      153   602   105   189    37     9    27
##  6 aaronha~   1956      3 ML1    NL      153   609   106   200    34    14    26
##  7 aaronha~   1957      4 ML1    NL      151   615   118   198    27     6    44
##  8 aaronha~   1958      5 ML1    NL      153   601   109   196    34     4    30
##  9 aaronha~   1959      6 ML1    NL      154   629   116   223    46     7    39
## 10 aaronha~   1960      7 ML1    NL      153   590   102   172    20    11    40
## # ... with 49,872 more rows, and 13 more variables: 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>
```

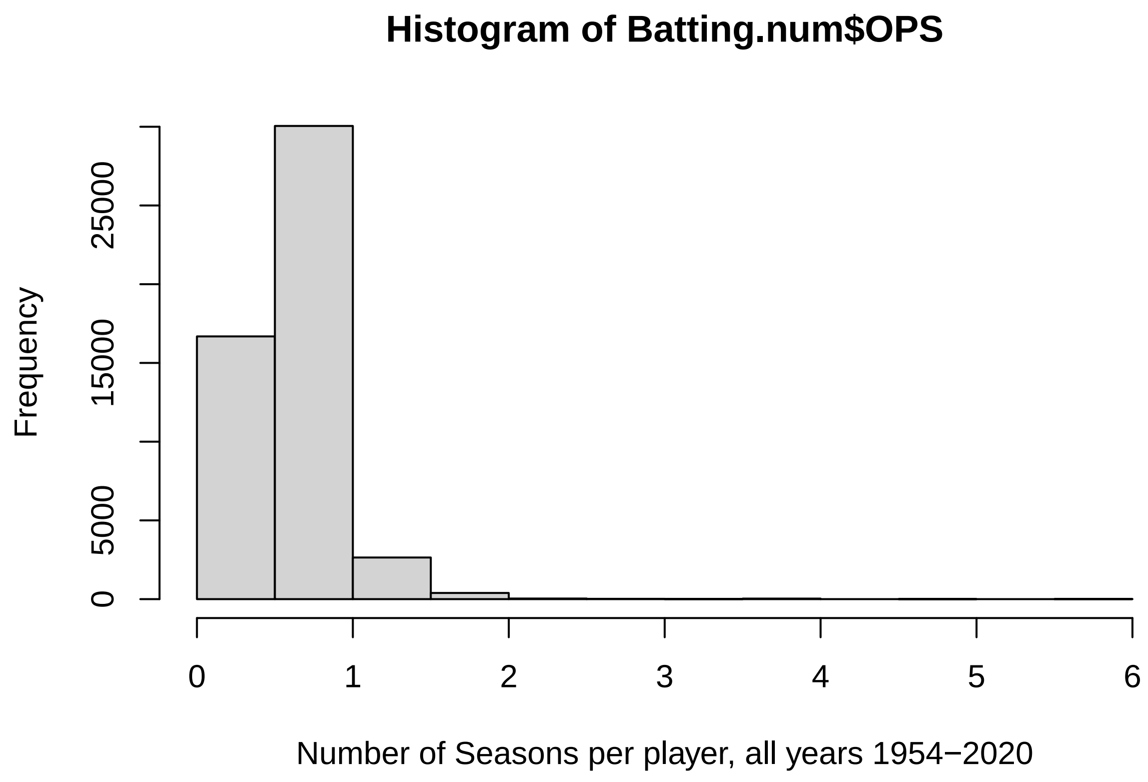
```
# Removing Players that have gaps in their careers
Batting_tsbl_gaps <- has_gaps(Batting_tsbl)
Batting_tsbl <- Batting_tsbl %>% left_join(Batting_tsbl_gaps, by="playerID")
Batting_tsbl <- Batting_tsbl[Batting_tsbl$.gaps != "TRUE",]
Batting_tsbl <- Batting_tsbl[,-26]
```

Exploratory data analysis

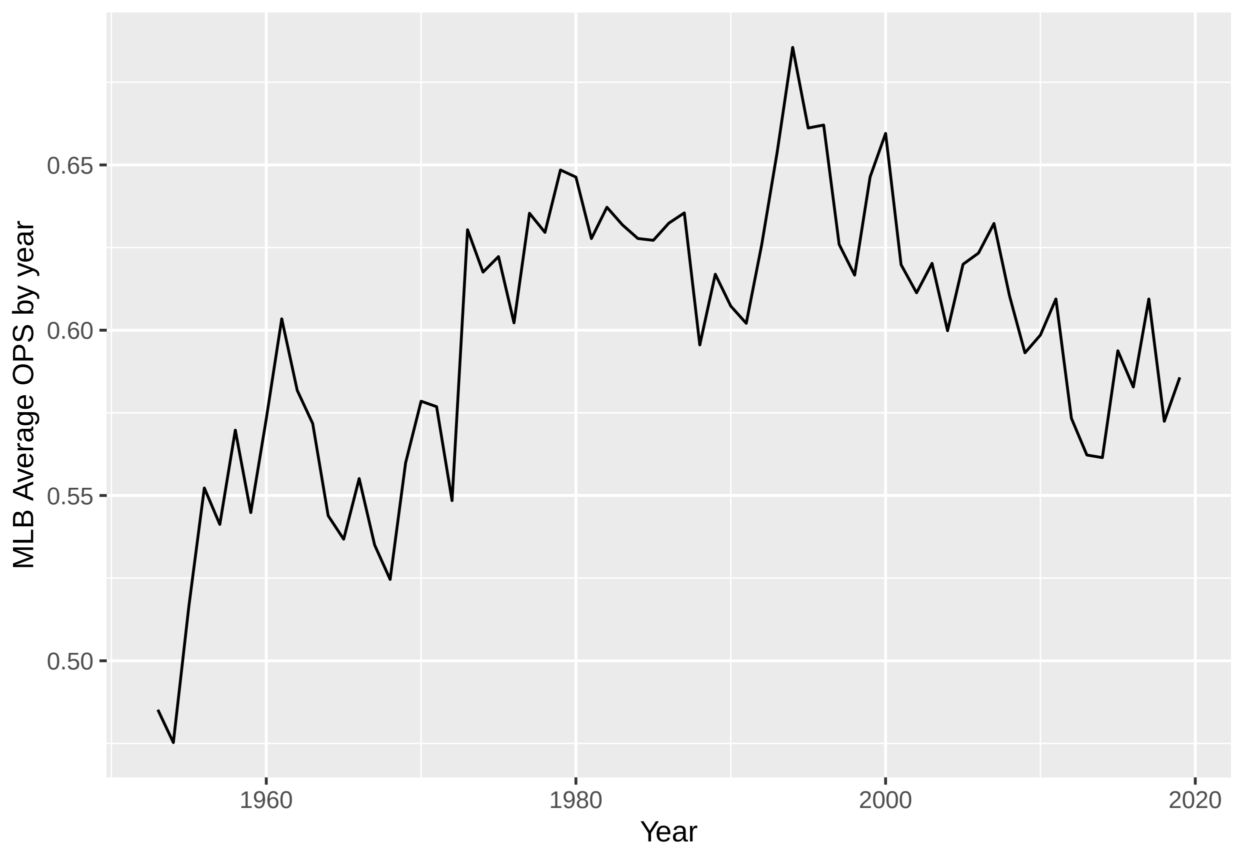
```
Batting.seasons <- Batting.merge %>% distinct(playerID, .keep_all = TRUE)
Batting.num <- Batting.merge[,c(25)]
summary(Batting.num)
```

```
##       OPS
##  Min.   :0.0000
##  1st Qu.:0.3750
##  Median :0.6711
##  Mean   :0.6063
##  3rd Qu.:0.8276
##  Max.   :6.0000
```

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



```
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")
```

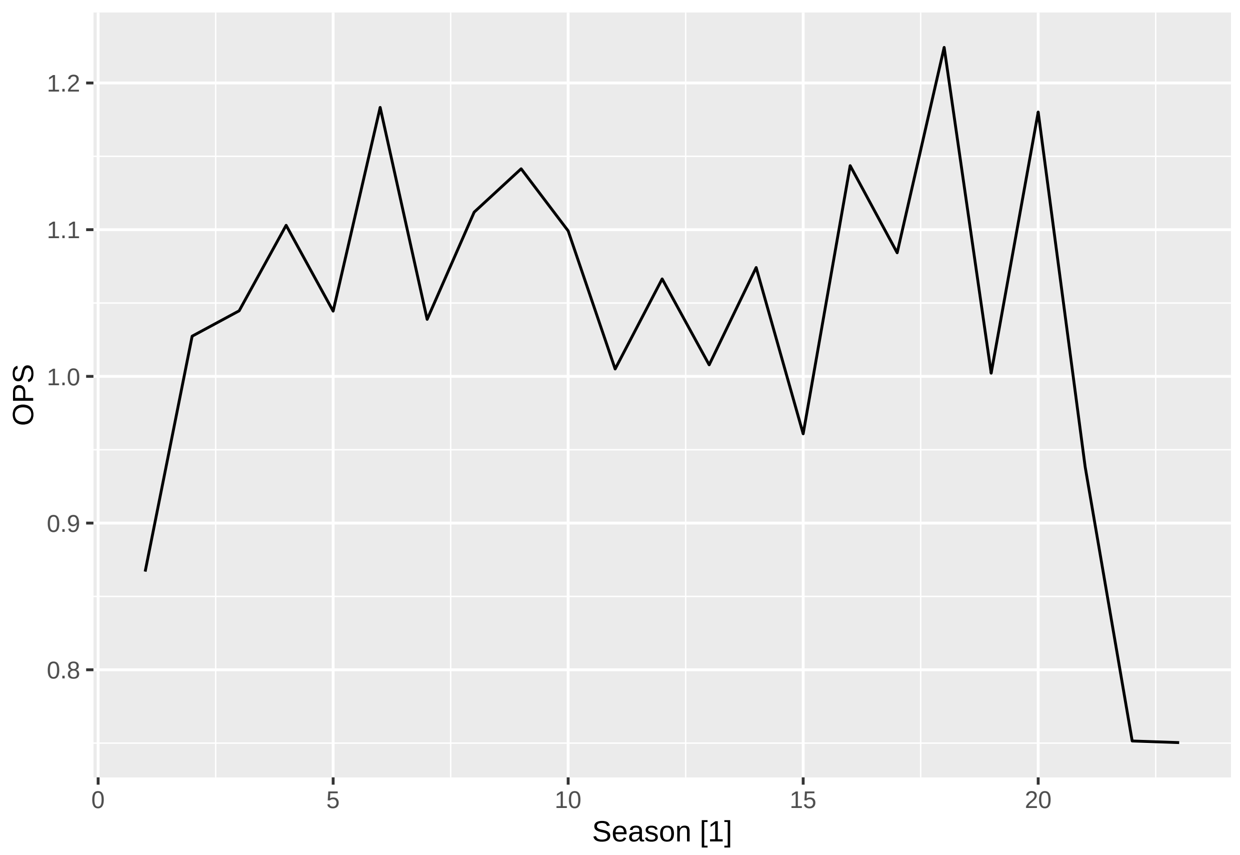


Correlation analysis

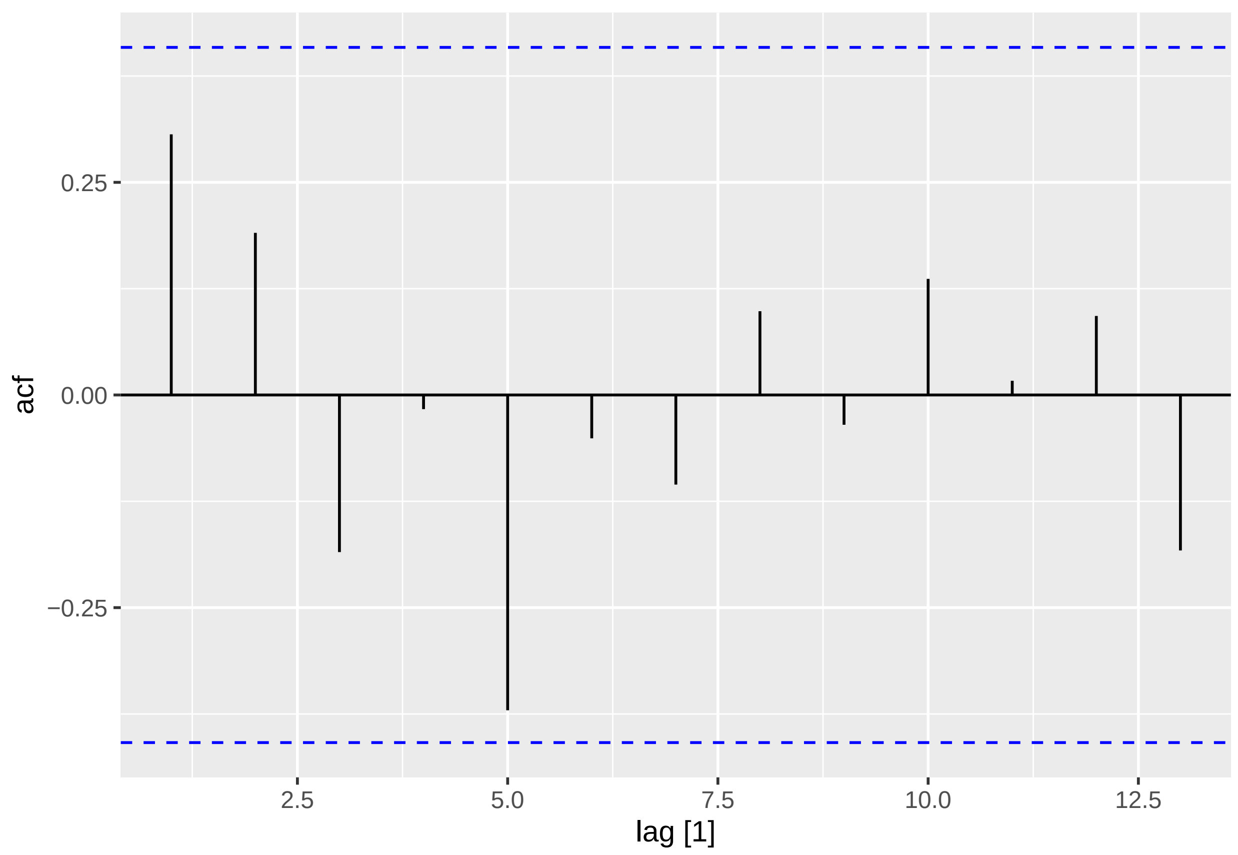
For the correlation analysis we will be looking at six individual former players careers and the OPS numbers they posted for each season of their career. We will look at Hank Aaron, Ichiro Suzuki, Derek Jeter, David Otriz, Pete Rose, and Barry Bonds.

Hank Aaron OPS

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

aaronha01_tsbl %>% autoplot(OPS)
```

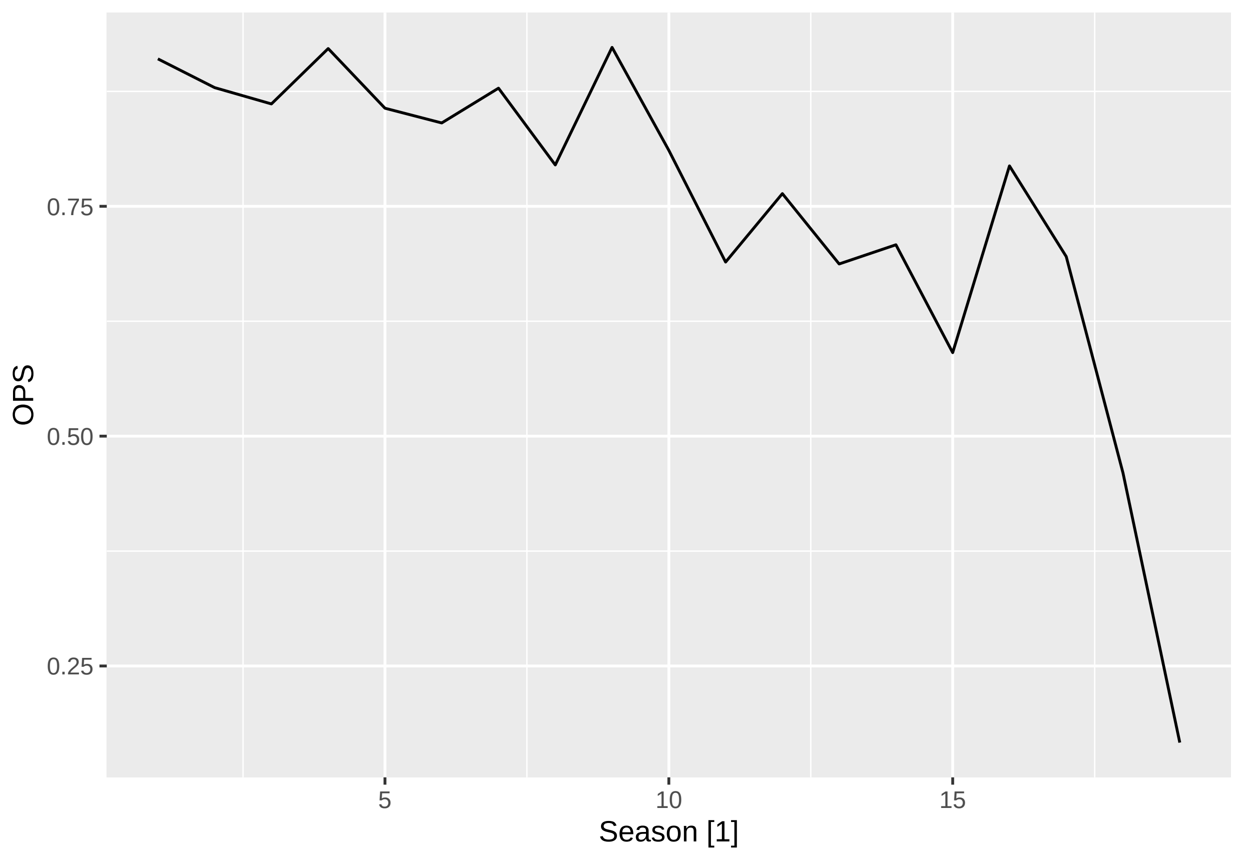


```
# Hank Aaron's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "aaronha01") %>%
  ACF(OPS) %>%
  autoplot(main = "Hank Aaron OPS ACF")
```

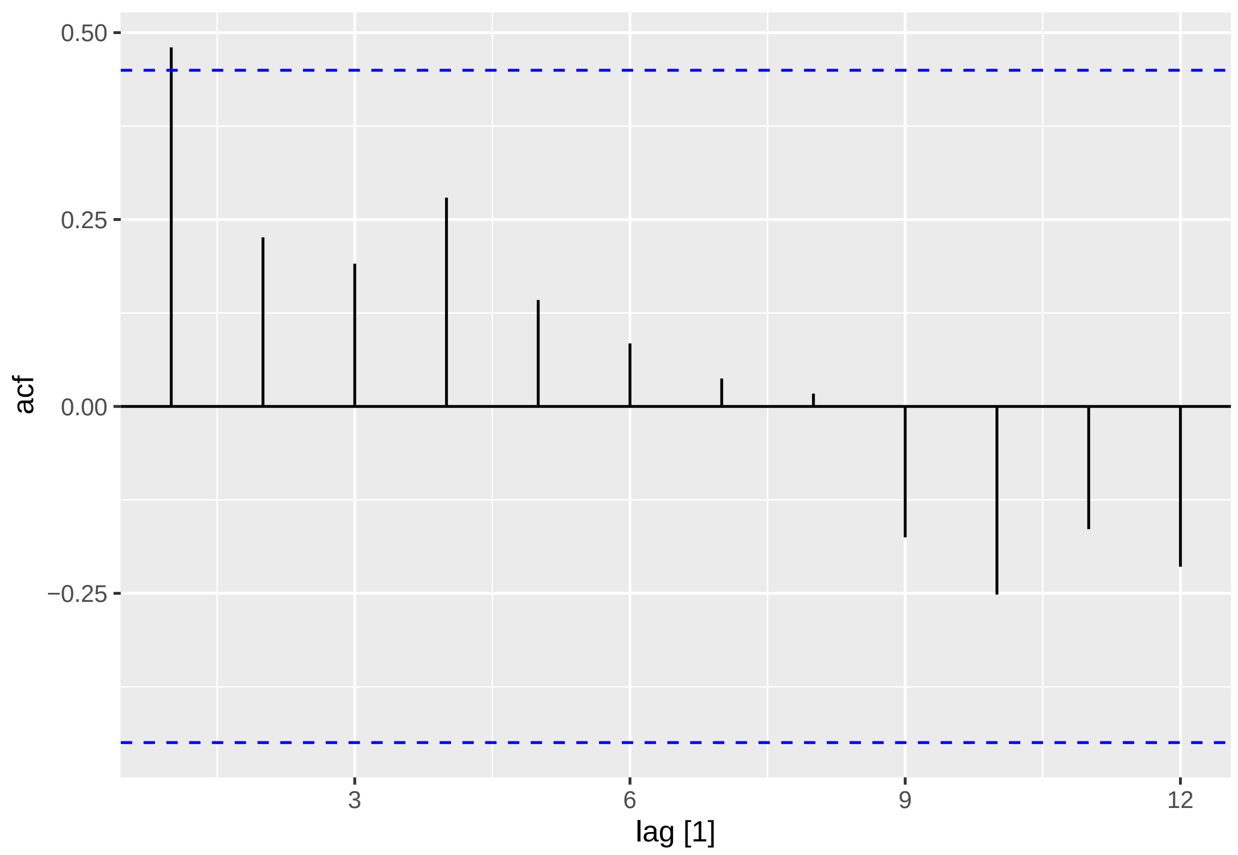


Ichiro Suzuki OPS

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

suzukic01_tsbl %>% autoplot(OPS)
```

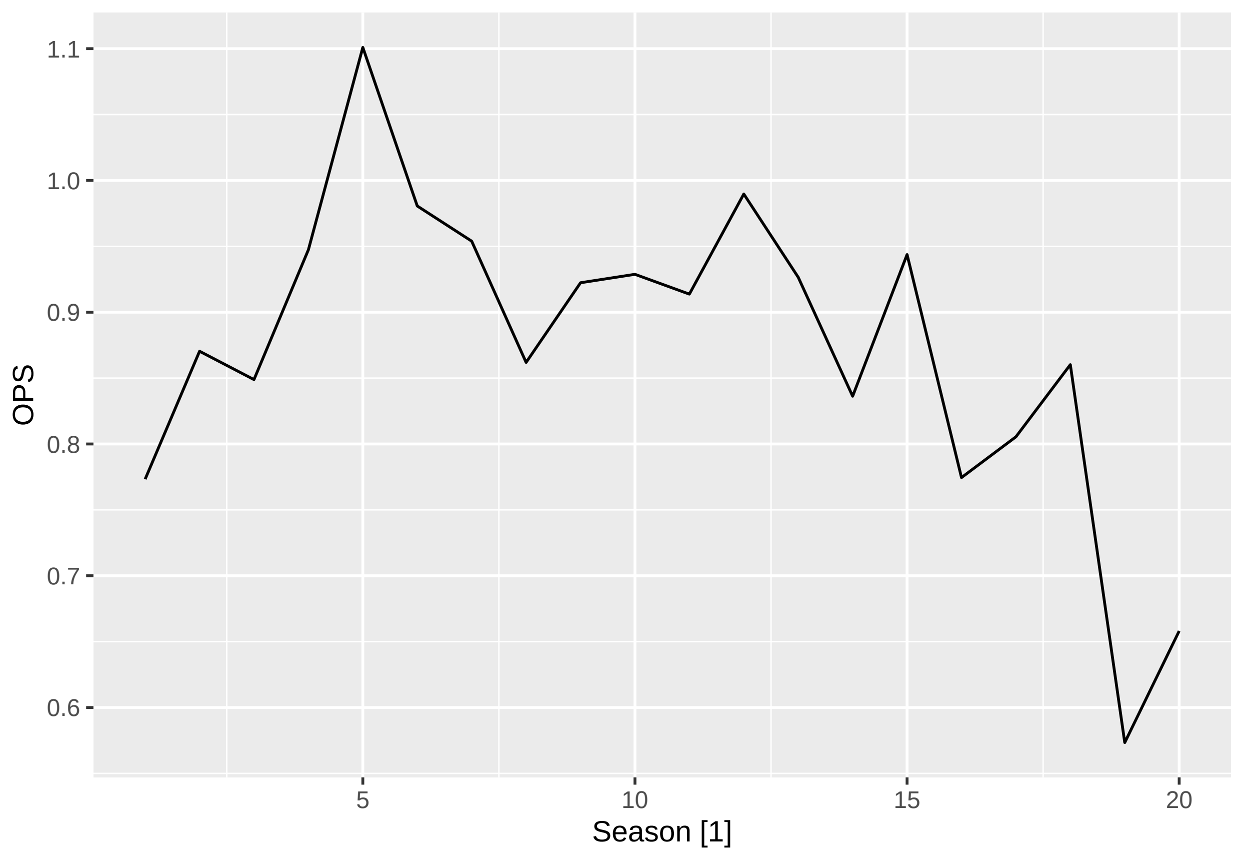


```
# Ichiro Suzuki's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "suzukic01") %>%
  ACF(OPS) %>%
  autoplot()
```

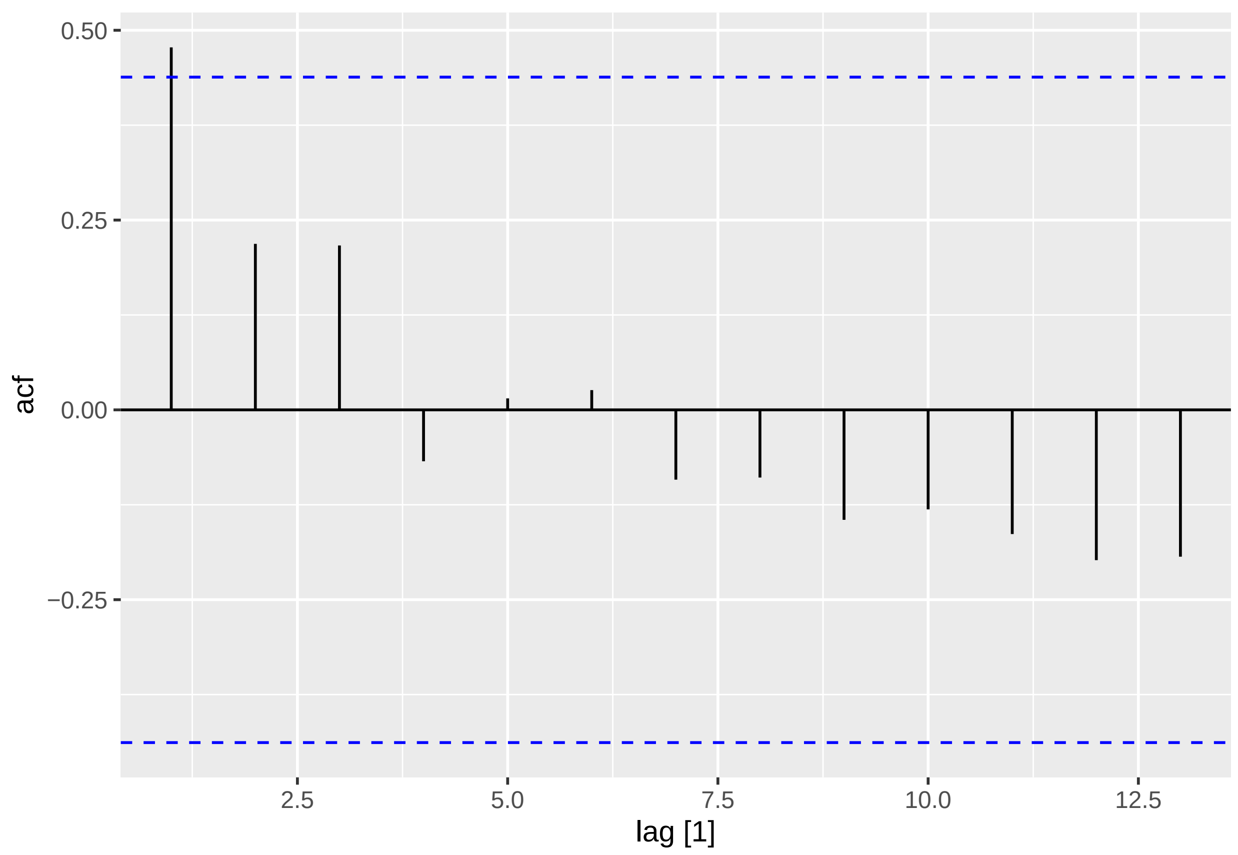


Derek Jeter OPS

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

jeterde01_tsbl %>% autoplot(OPS)
```

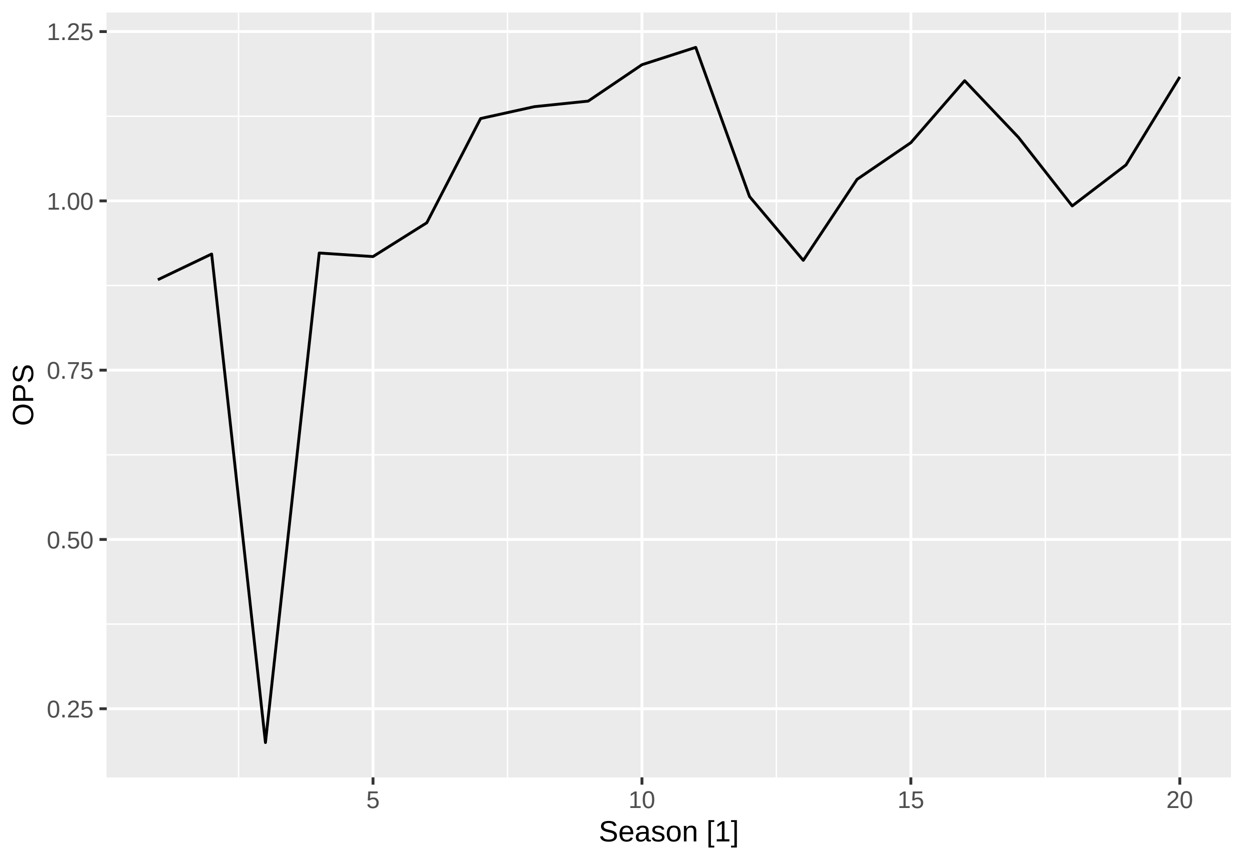


```
# Derek Jeter's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "jeterde01") %>%
  ACF(OPS) %>%
  autoplot()
```

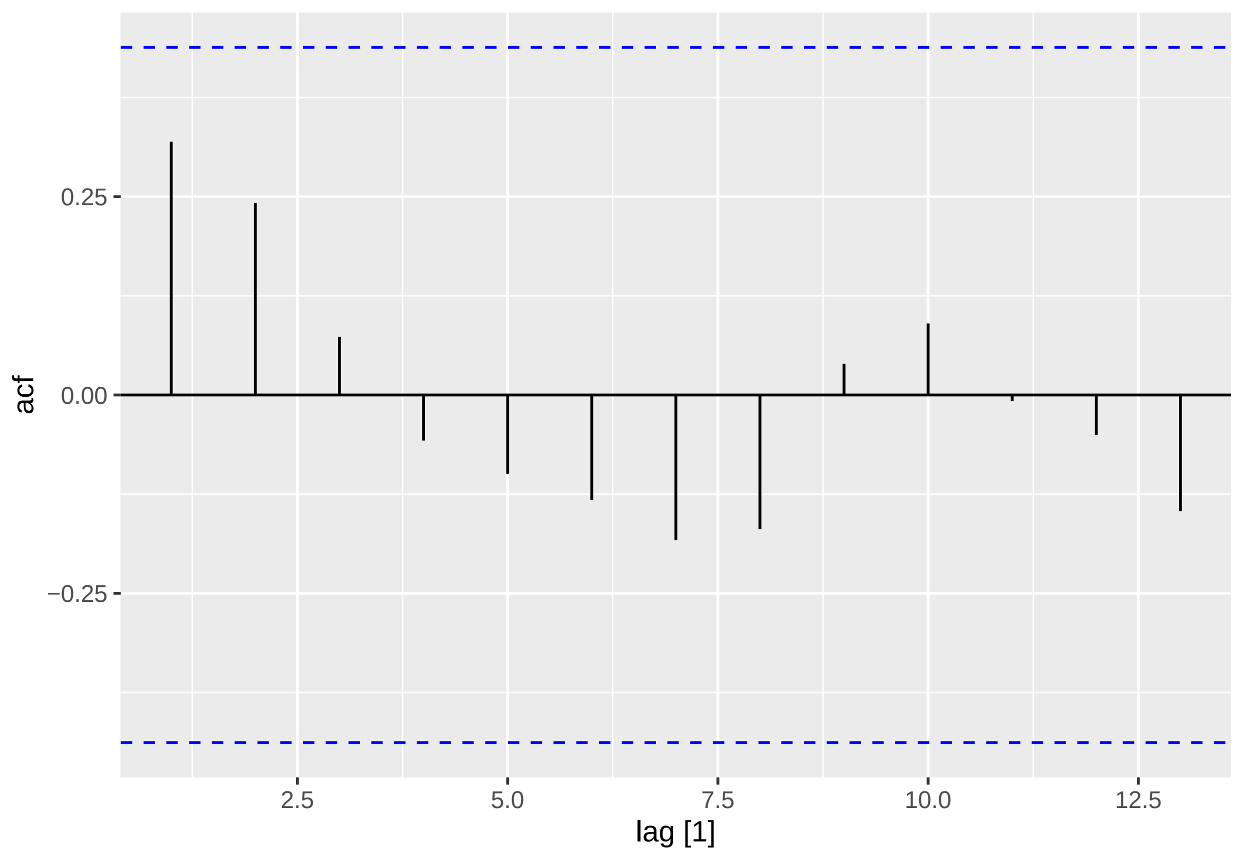


David Ortiz OPS

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

ortizda01_tsbl %>% autoplot(OPS)
```

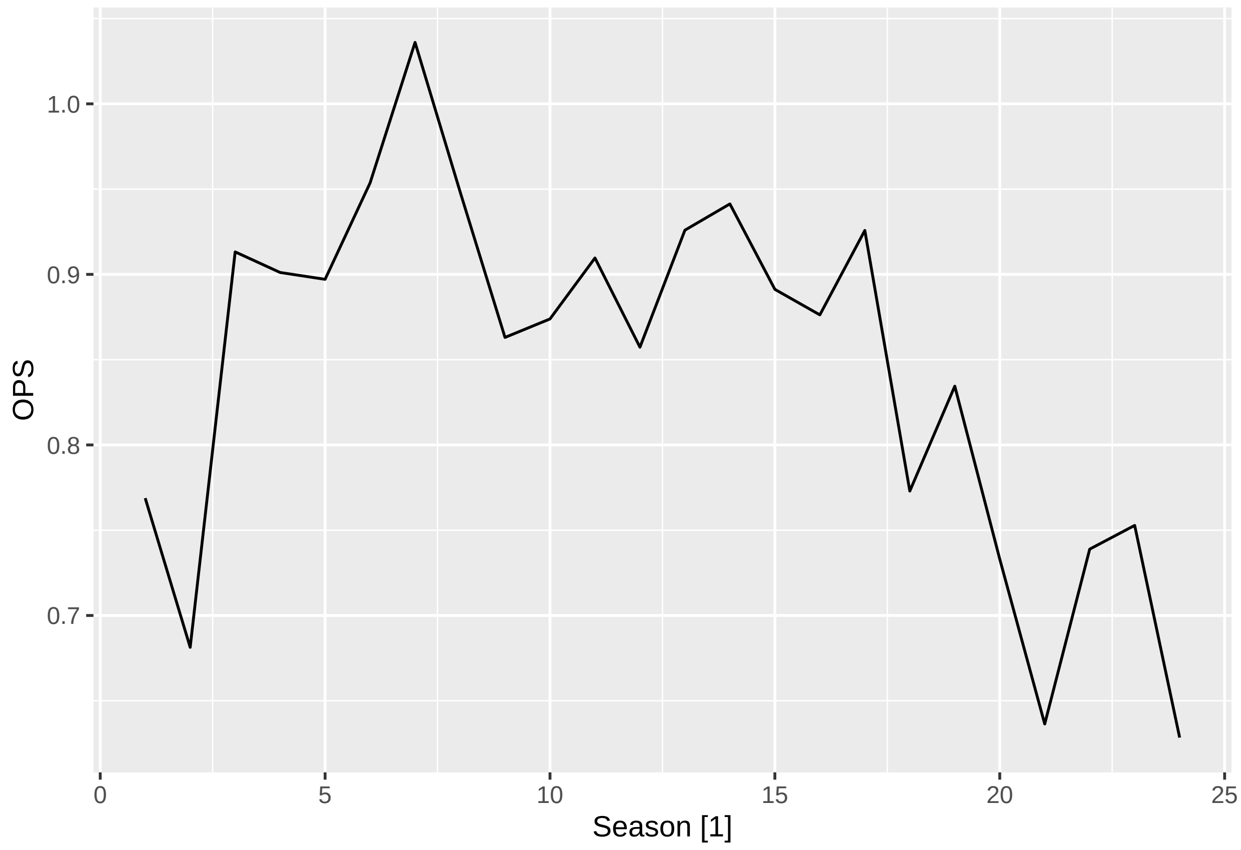


```
# David Ortiz's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "ortizda01") %>%
  ACF(OPS) %>%
  autoplot()
```

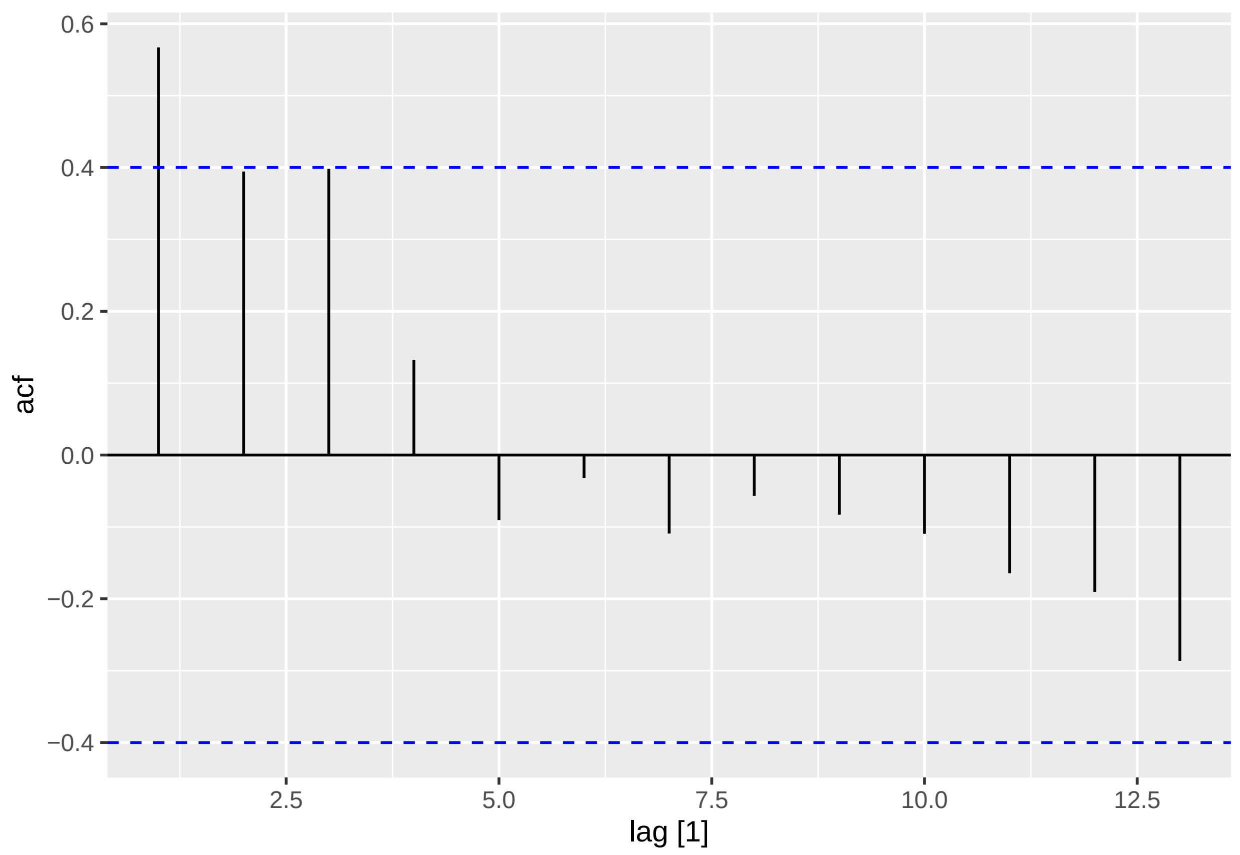


Pete Rose OPS

```
# Pete Rose's OPS plot
Batting_tsbl %>%
  filter(playerID == "rosepe01") -> rosepe01_tsbl

rosepe01_tsbl %>% autoplot(OPS)
```

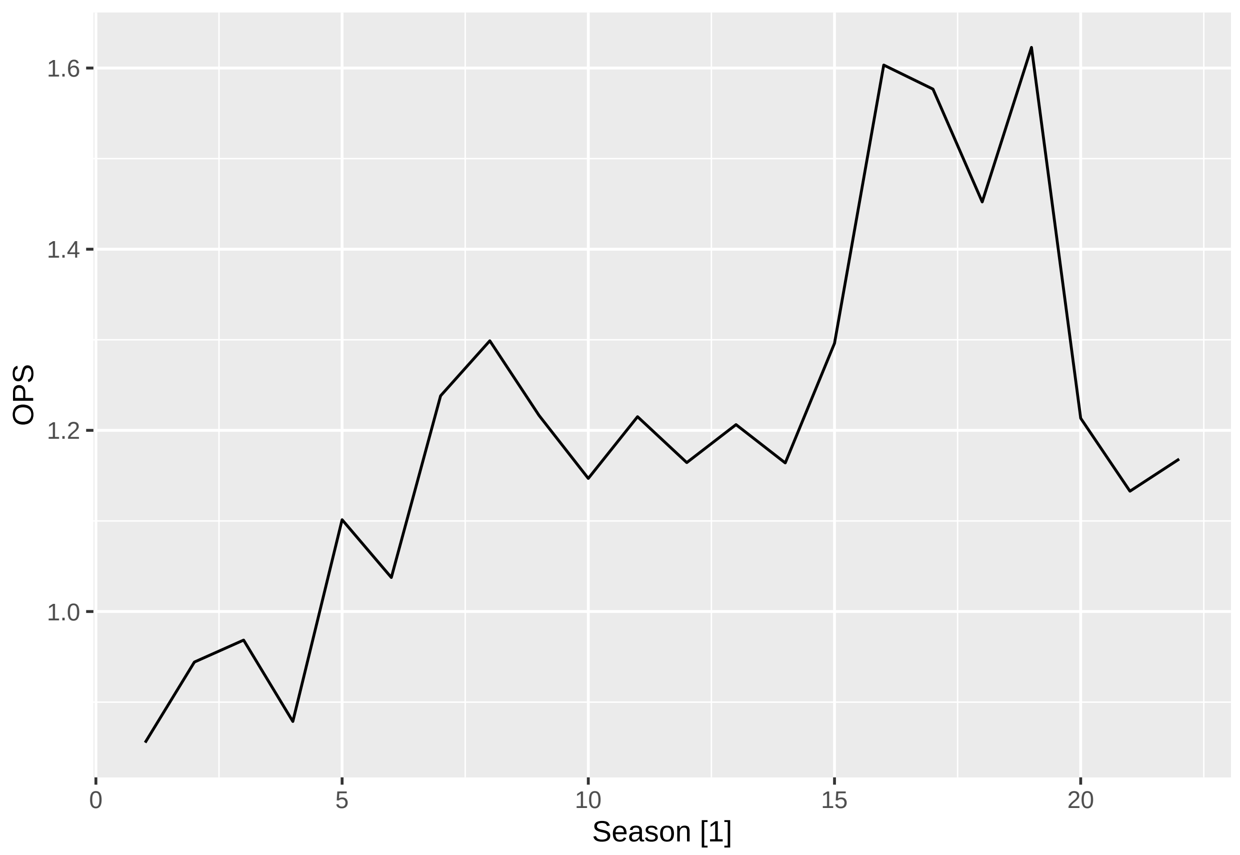


```
# Pete Rose's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "rosepe01") %>%
  ACF(OPS) %>%
  autoplot()
```

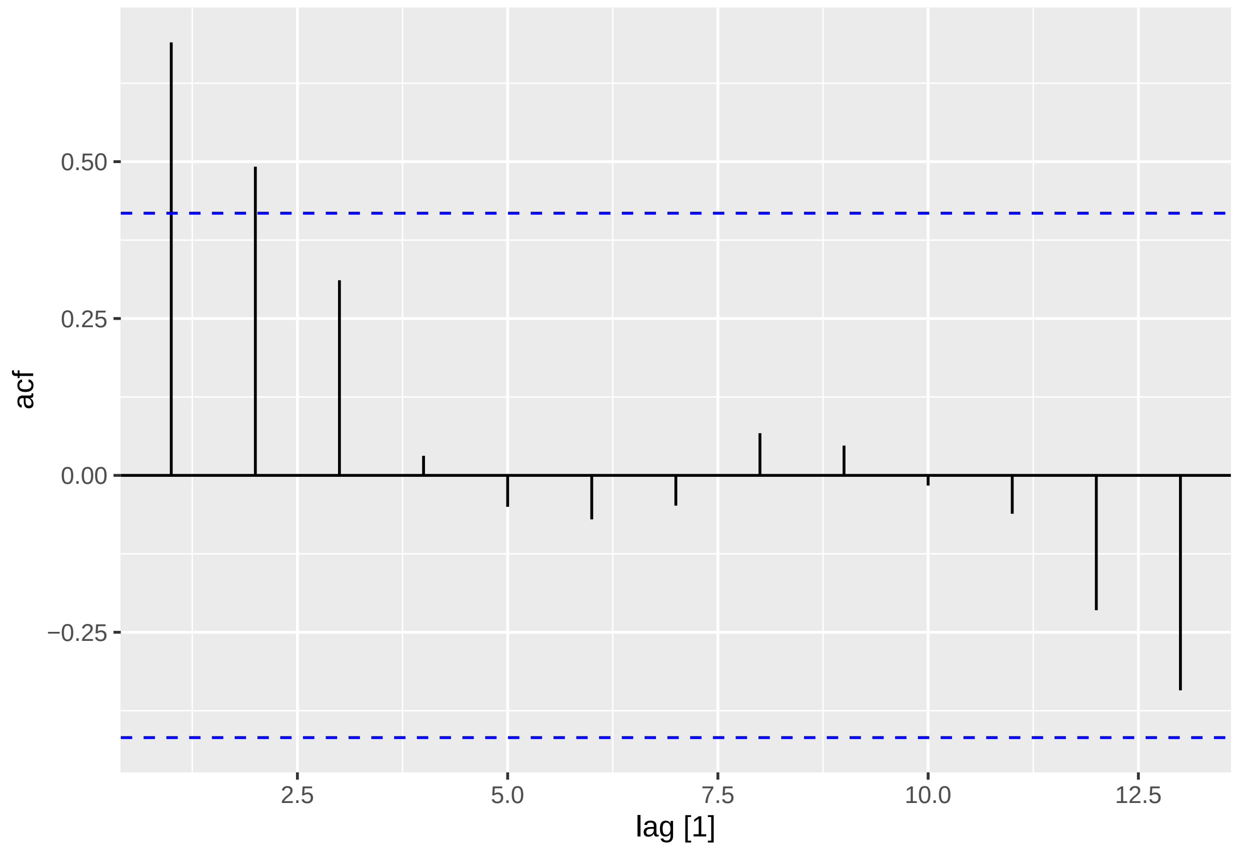


Barry Bonds OPS

```r
# Barry Bonds's OPS plot
Batting_tsbl %>%
  filter(playerID == "bondsba01") -> bondsba01_tsbl

bondsba01_tsbl %>% autoplot(OPS)
```



```
# Barry Bonds's OPS ACF plot
Batting_tsbl %>%
  filter(playerID == "bondsba01") %>%
  ACF(OPS) %>%
  autoplot()
```



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 are not correlated to each other (except the first and second lags in some cases) and stay well below the threshold levels so there is not any major amount of information that is not being captured.

AR(1) and Auto ARIMA Modeling with six different players.

In this section I will look at setting up the models for each of the six players from the last section. I will set up an AR(1) model as well as an auto.arima model and compare both the RMSE and the AIC of both models to evaluate the performance of each. To calculate the RMSE we will use 80% of the players career to train the model and 20% of the players career to calculate the RMSE between the predicted OPS and the True (testing) OPS values to evaluate how well their predictions were.

Performance table Setup

```r
ARIMA_perf_RMSE_table <- data.frame(matrix(ncol = 3, nrow = 6))
colnames(ARIMA_perf_RMSE_table) <- c("Player","p(1,0,0)", "AUTO")
ARIMA_perf_RMSE_table[1, 1] = "Hank Aaron"
ARIMA_perf_RMSE_table[2, 1] = "Ichiro Suzuki"
ARIMA_perf_RMSE_table[3, 1] = "Derek Jeter"
ARIMA_perf_RMSE_table[4, 1] = "David Ortiz"
ARIMA_perf_RMSE_table[5, 1] = "Pete Rose"
ARIMA_perf_RMSE_table[6, 1] = "Barry Bonds"
```

```
ARIMA_perf_AIC_table <- data.frame(matrix(ncol = 3, nrow = 6))
colnames(ARIMA_perf_AIC_table) <- c("Player","p(1,0,0)", "AUTO")
ARIMA_perf_AIC_table[1, 1] = "Hank Aaron"
ARIMA_perf_AIC_table[2, 1] = "Ichiro Suzuki"
ARIMA_perf_AIC_table[3, 1] = "Derek Jeter"
ARIMA_perf_AIC_table[4, 1] = "David Ortiz"
ARIMA_perf_AIC_table[5, 1] = "Pete Rose"
ARIMA_perf_AIC_table[6, 1] = "Barry Bonds"
```

Hank Aaron Ar(1) set up and predictions.

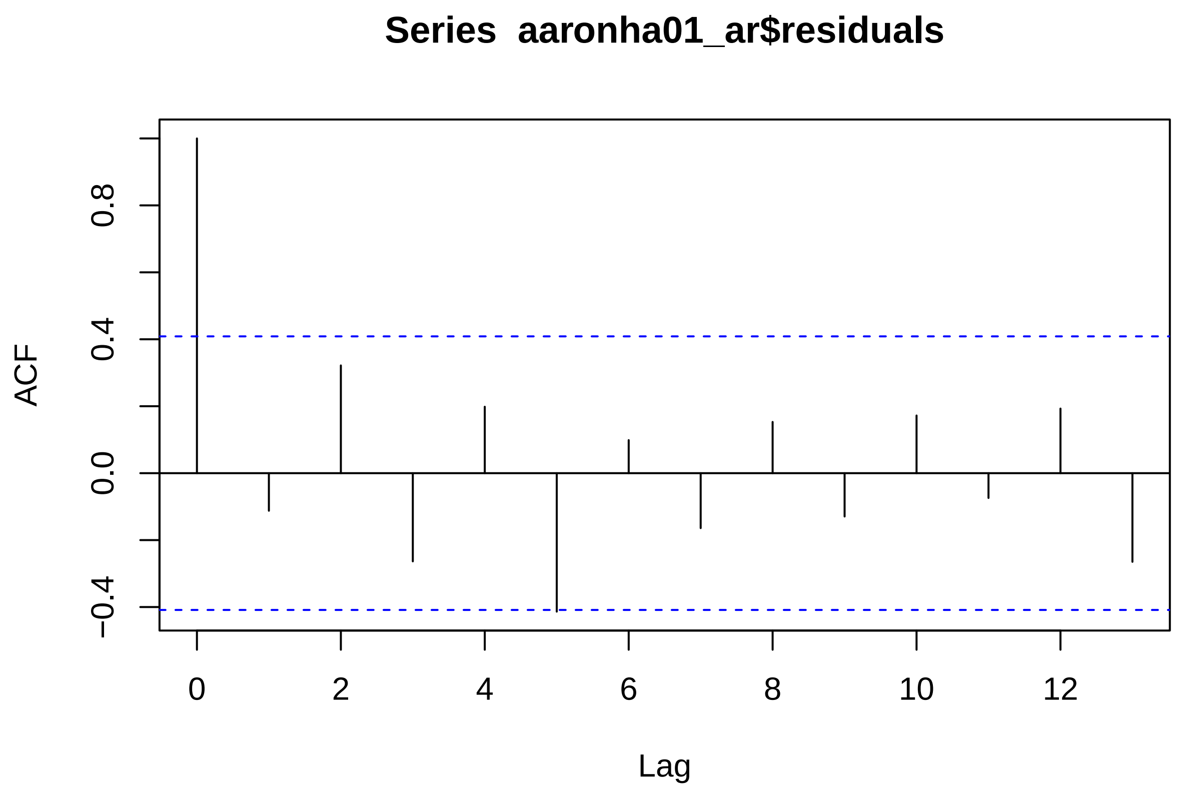
```
# Setting up ar100 model
aaronha01_ar <- arima(aaronha01_tsbl$OPS , order = c(1, 0, 0))
summary(aaronha01_ar)
```

```
##
## Call:
## arima(x = aaronha01_tsbl$OPS, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.4475     1.0219
## s.e.  0.2279     0.0423
##
## sigma^2 estimated as 0.01221:  log likelihood = 17.92,  aic = -29.84
##
## Training set error measures:
##                       ME      RMSE        MAE        MPE     MAPE      MASE
## Training set 0.003732302 0.1104885 0.09295328 -0.9133509 9.388076 0.8780043
##                    ACF1
## Training set -0.1123133
```

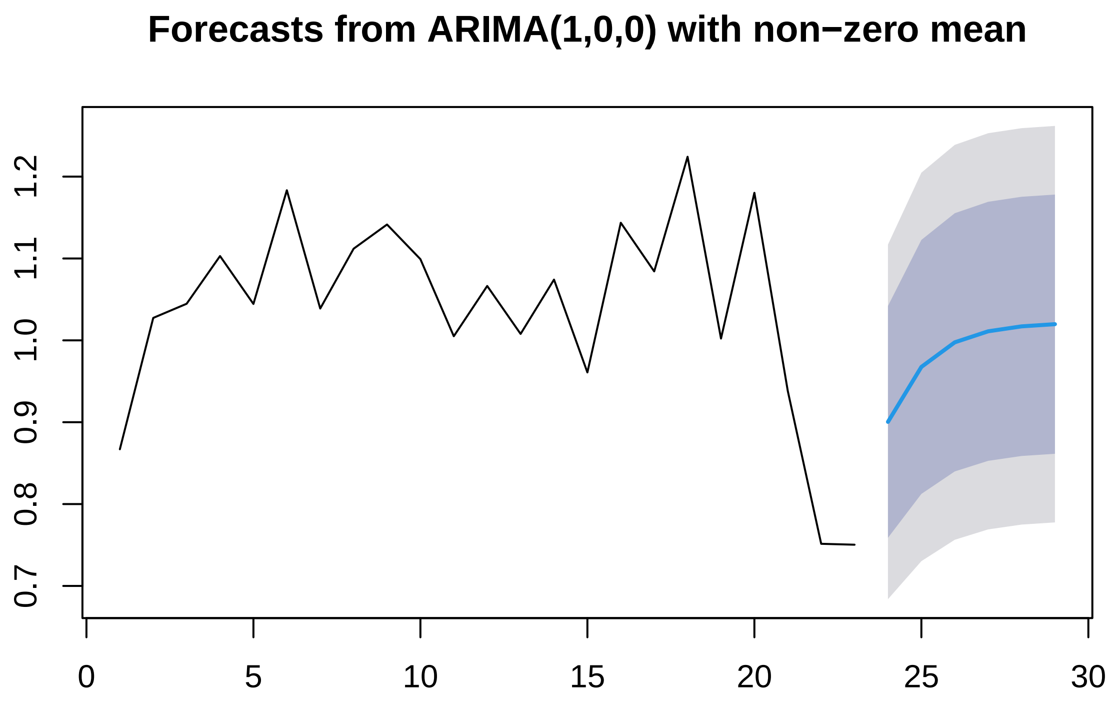
```
# Setting up auto.arima model
aaronha01_auto <- auto.arima(aaronha01_tsbl$OPS)
summary(aaronha01_auto)
```

```
## Series: aaronha01_tsbl$OPS
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.4475  1.0219
## s.e.  0.2279  0.0423
##
## sigma^2 = 0.01337:  log likelihood = 17.92
## AIC=-29.84   AICc=-28.57   BIC=-26.43
##
## Training set error measures:
##                       ME      RMSE        MAE        MPE     MAPE      MASE
## Training set 0.003732302 0.1104885 0.09295328 -0.9133509 9.388076 0.8780043
##                       ACF1
## Training set -0.1123133
```

```
# Storing AIC values for both models
ARIMA_perf_AIC_table[1,2] <- aaronha01_ar$aic
ARIMA_perf_AIC_table[1,3] <- aaronha01_auto$aic

# Looking at residuals for the model
acf(aaronha01_ar$residuals)
```



```
# Setting up 20% of career for Testing Data and 80% of career for Training Data
aaronha01_tsbl_test <- aaronha01_tsbl %>%
  filter(aaronha01_tsbl$yearID >= 1971)

aaronha01_tsbl_train <- aaronha01_tsbl %>%
  filter(aaronha01_tsbl$yearID < 1971)

# Fitting Model with Training Data
aaronha01_ar100 <- arima(aaronha01_tsbl_train$OPS , order = c(1, 0, 0))

# Using Model to predict on 20% of Players Career
aaronha01_fc_ar <- forecast(aaronha01_ar, h = 6)

# Plotting Prediction
plot(aaronha01_fc_ar)
```



```
# Calculating RMSE
aaronha01_fc_ar_mean <- aaronha01_fc_ar$mean[1:6]
aaronha01_fc_ar100_acc <- accuracy(aaronha01_fc_ar_mean, aaronha01_tsbl_test$OPS)

# Storing RMSE Values
ARIMA_perf_RMSE_table[1,2] <- aaronha01_fc_ar100_acc[1,2]
ARIMA_perf_RMSE_table[1,3] <- aaronha01_fc_ar100_acc[1,2]
```

Ichiro Suzuki Ar(1) set up and predictions.

```
# Setting up ar100 model
suzukic01_ar <- arima(suzukic01_tsbl$OPS , order = c(1, 0, 0))
summary(suzukic01_ar)
```

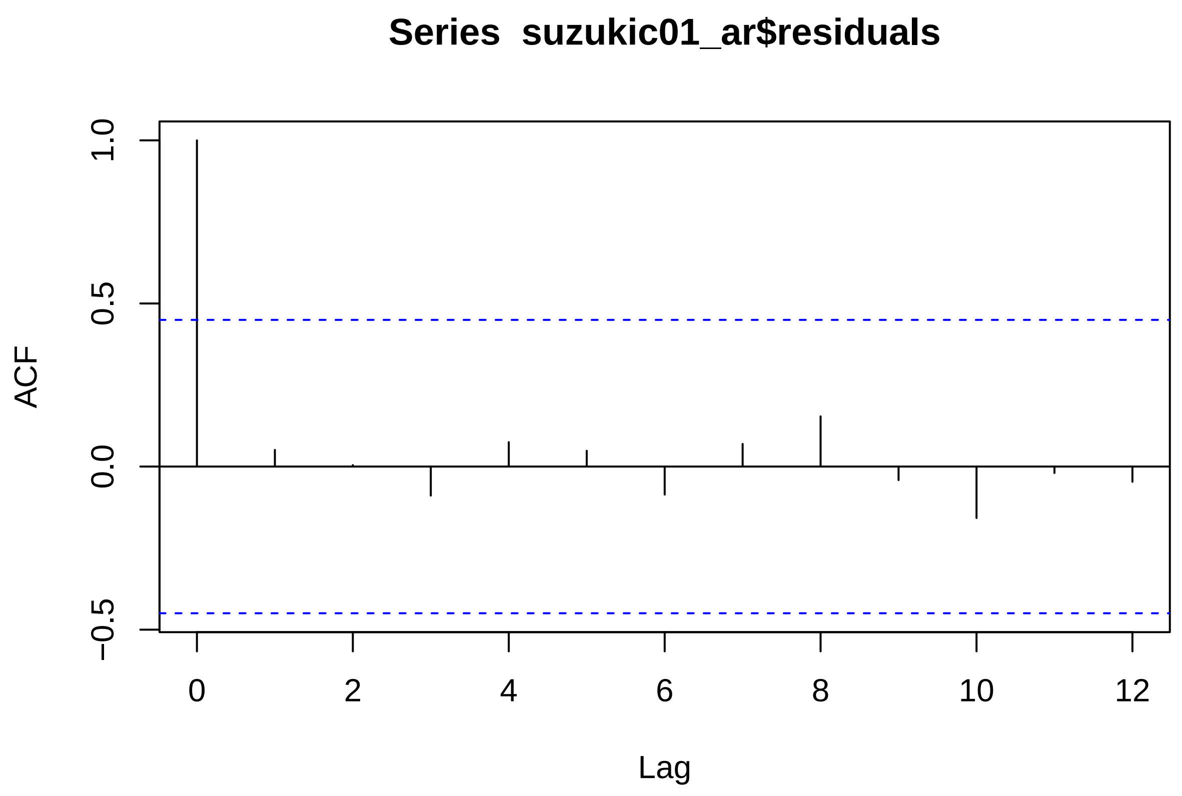
```
##
## Call:
## arima(x = suzukic01_tsbl$OPS, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.8910     0.6517
## s.e.  0.1238     0.2013
##
```

```
## sigma^2 estimated as 0.01515:  log likelihood = 12.05,  aic = -18.11
##
## Training set error measures:
##                      ME     RMSE        MAE       MPE     MAPE     MASE
## Training set -0.0195745 0.123086 0.09489539 -12.19648 21.06038 0.952944
##                    ACF1
## Training set 0.05114648
```

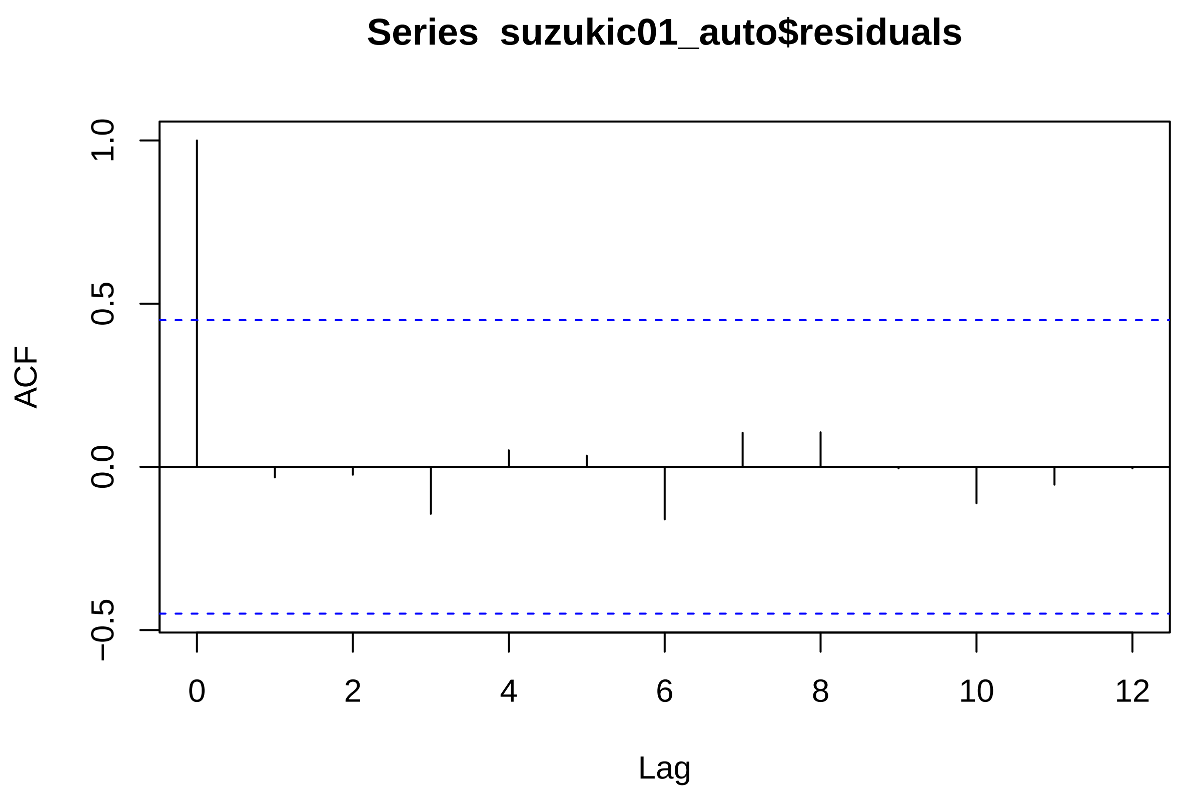
```
# Setting up auto.arima model
suzukic01_auto <- auto.arima(suzukic01_tsbl$OPS)
summary(suzukic01_auto)
```

```
## Series: suzukic01_tsbl$OPS
## ARIMA(0,1,0)
##
## sigma^2 = 0.01553:  log likelihood = 11.95
## AIC=-21.89   AICc=-21.64   BIC=-21
##
## Training set error measures:
##                       ME      RMSE        MAE      MPE     MAPE      MASE
## Training set -0.03909453 0.1212788 0.09438808 -14.0139 20.65358 0.9478496
##                     ACF1
## Training set -0.03241385
```

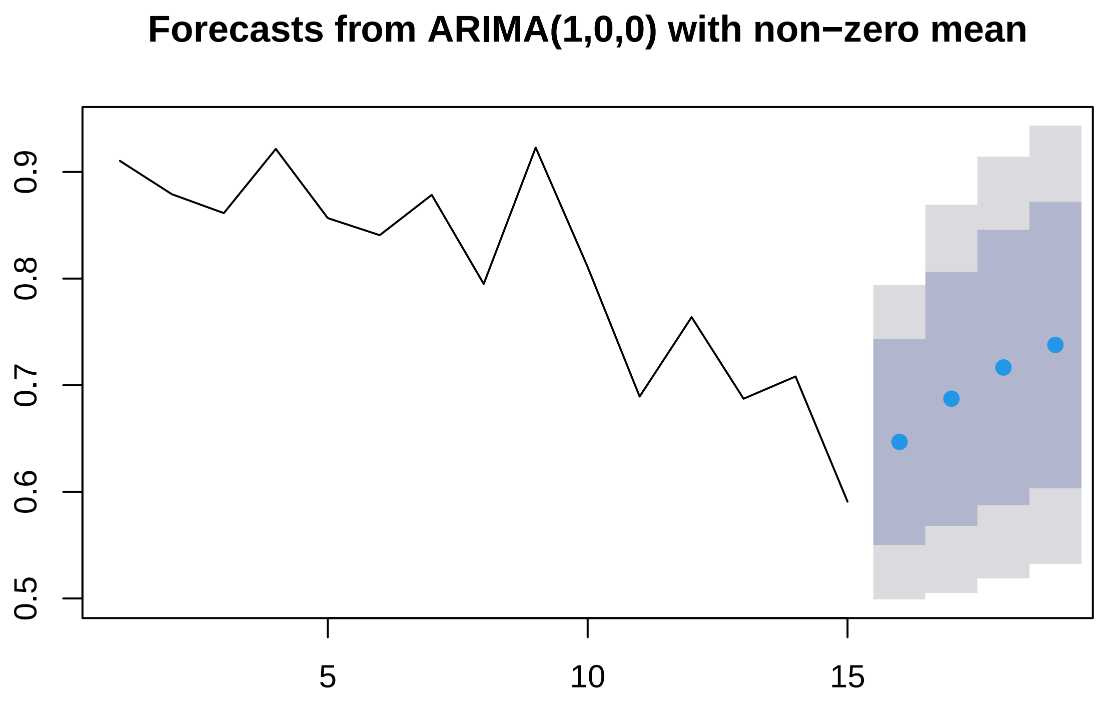
```
# Storing AIC values for both models
ARIMA_perf_AIC_table[2,2] <- suzukic01_ar$aic
ARIMA_perf_AIC_table[2,3] <- suzukic01_auto$aic

# Looking at residuals for the model
acf(suzukic01_ar$residuals)
```



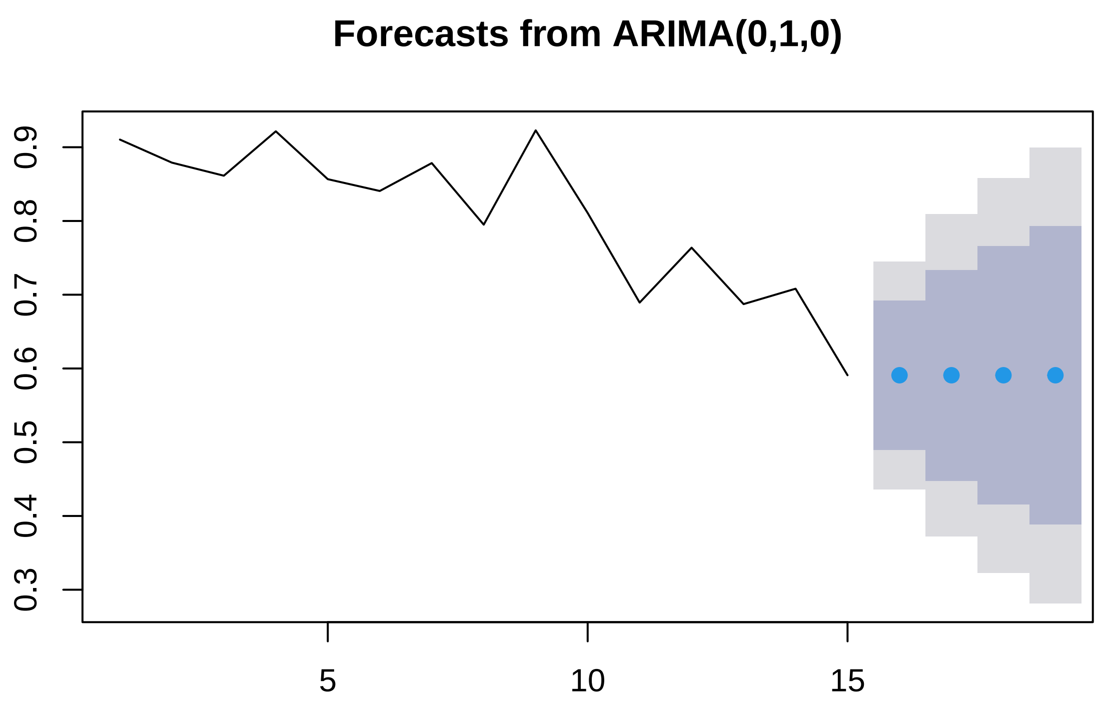
```
acf(suzukic01_auto$residuals)
```



```r
# Setting up 20% of career for Testing Data and 80% of career for Training Data
suzukic01_tsbl_test <- suzukic01_tsbl %>%
  filter(suzukic01_tsbl$yearID >= 2016)

suzukic01_tsbl_train <- suzukic01_tsbl %>%
  filter(suzukic01_tsbl$yearID < 2016)

# Fitting Models with Training Data
suzukic01_ar100 <- arima(suzukic01_tsbl_train$OPS , order = c(1, 0, 0))

suzukic01_ar010 <- arima(suzukic01_tsbl_train$OPS , order = c(0, 1, 0))

# Using Models to predict on 20% of Players Career
suzukic01_fc_ar100 <- forecast(suzukic01_ar100, h = 4)

suzukic01_fc_ar010 <- forecast(suzukic01_ar010, h = 4)

# Plotting Predictions
plot(suzukic01_fc_ar100)
```



```
plot(suzukic01_fc_ar010)
```



```r
# Calculating RMSE
suzukic01_fc_ar100_mean <- suzukic01_fc_ar100$mean[1:4]
suzukic01_fc_ar010_mean <- suzukic01_fc_ar010$mean[1:4]

suzukic01_fc_ar100_acc <- accuracy(suzukic01_fc_ar100_mean, suzukic01_tsbl_test$OPS)
suzukic01_fc_ar010_acc <- accuracy(suzukic01_fc_ar010_mean, suzukic01_tsbl_test$OPS)

# Storing RMSE Values
ARIMA_perf_RMSE_table[2,2] <- suzukic01_fc_ar100_acc[1,2]
ARIMA_perf_RMSE_table[2,3] <- suzukic01_fc_ar010_acc[1,2]
```

Derek Jeter Ar(1) set up and predictions.

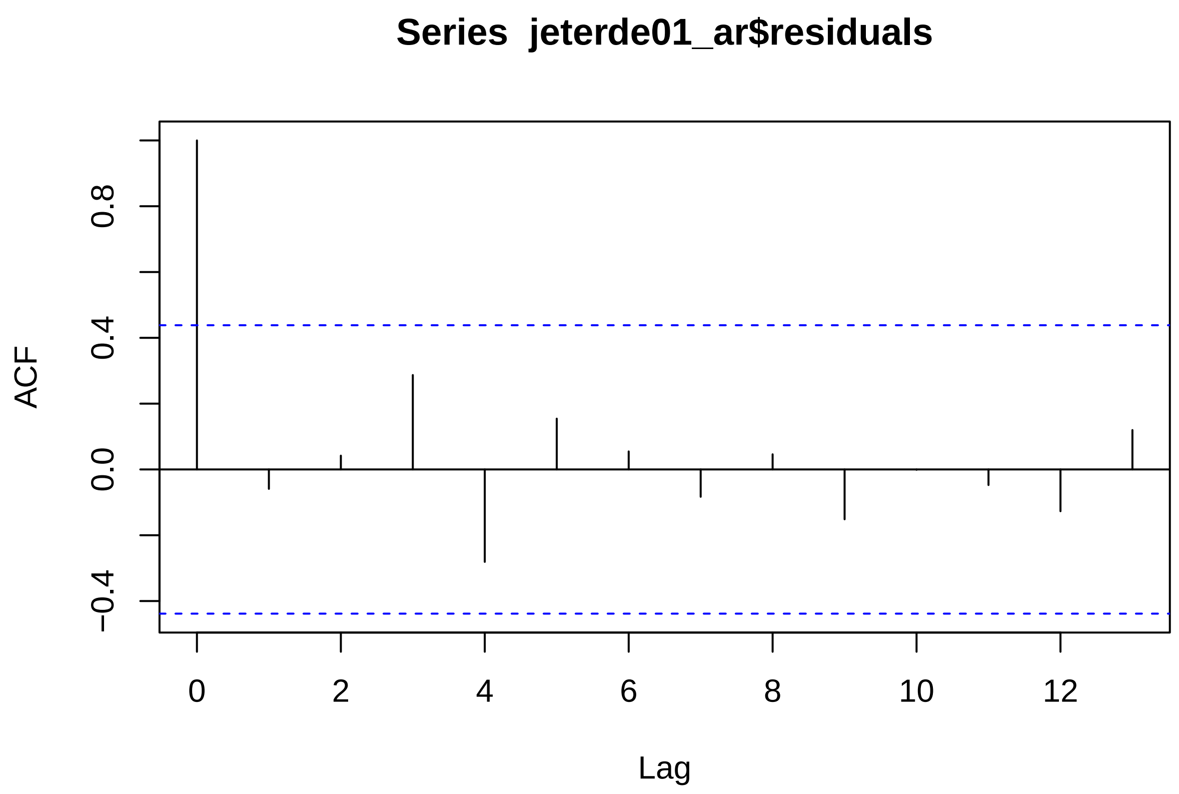
```r
# Setting up ar100 model
jeterde01_ar <- arima(jeterde01_tsbl$OPS , order = c(1, 0, 0))
summary(jeterde01_ar)
```

```
##
## Call:
## arima(x = jeterde01_tsbl$OPS, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.5772     0.8545
## s.e.  0.1991     0.0500
##
## sigma^2 estimated as 0.009415:  log likelihood = 18.07,  aic = -30.15
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 0.003089453 0.09703168 0.06971928 -1.144136 8.628886 0.8003585
##                    ACF1
## Training set -0.0590572
```

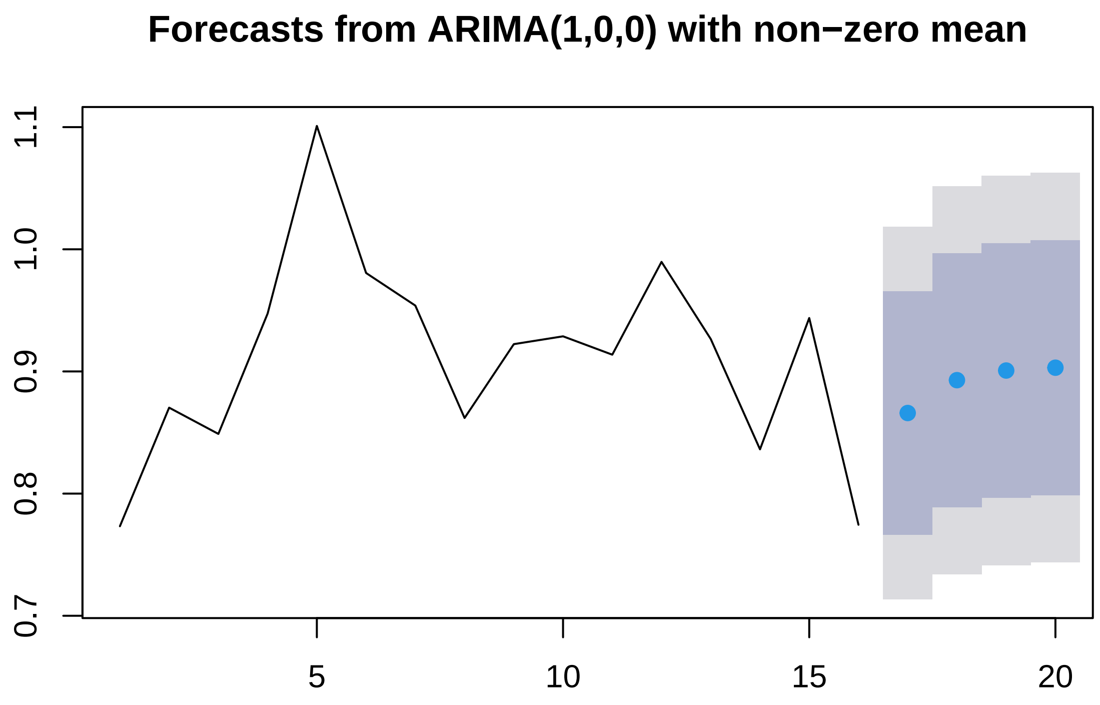
```
# Setting up auto.arima model
jeterde01_auto <- auto.arima(jeterde01_tsbl$OPS)
summary(jeterde01_auto)
```

```
## Series: jeterde01_tsbl$OPS
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.5772  0.8545
## s.e.  0.1991  0.0500
##
## sigma^2 = 0.01046:  log likelihood = 18.07
## AIC=-30.15   AICc=-28.65   BIC=-27.16
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 0.003089453 0.09703168 0.06971928 -1.144136 8.628886 0.8003585
##                    ACF1
## Training set -0.0590572
```

```
# Storing AIC values for both models
ARIMA_perf_AIC_table[3,2] <- jeterde01_ar$aic
ARIMA_perf_AIC_table[3,3] <- jeterde01_auto$aic

# Looking at residuals for the model
acf(jeterde01_ar$residuals)
```



```
# Setting up 20% of career for Testing Data and 80% of career for Training Data
jeterde01_tsbl_test <- jeterde01_tsbl %>%
  filter(jeterde01_tsbl$yearID >= 2011)

jeterde01_tsbl_train <- jeterde01_tsbl %>%
  filter(jeterde01_tsbl$yearID < 2011)

# Fitting Model with Training Data
jeterde01_ar <- arima(jeterde01_tsbl_train$OPS , order = c(1, 0, 0))

# Using Model to predict on 20% of Players Career
jeterde01_fc_ar <- forecast(jeterde01_ar, h = 4)

# Plotting Prediction
plot(jeterde01_fc_ar)
```



```
# Calculating RMSE
jeterde01_fc_ar_mean <- jeterde01_fc_ar$mean[1:4]

jeterde01_fc_ar100_acc <- accuracy(jeterde01_fc_ar_mean, jeterde01_tsbl_test$OPS)

# Storing RMSE Values
ARIMA_perf_RMSE_table[3,2] <- jeterde01_fc_ar100_acc[1,2]
ARIMA_perf_RMSE_table[3,3] <- jeterde01_fc_ar100_acc[1,2]
```

David Ortiz Ar(1) set up and predictions.

```
# Setting up ar100 model
ortizda01_ar <- arima(ortizda01_tsbl$OPS , order = c(1, 0, 0))
summary(ortizda01_ar)
```

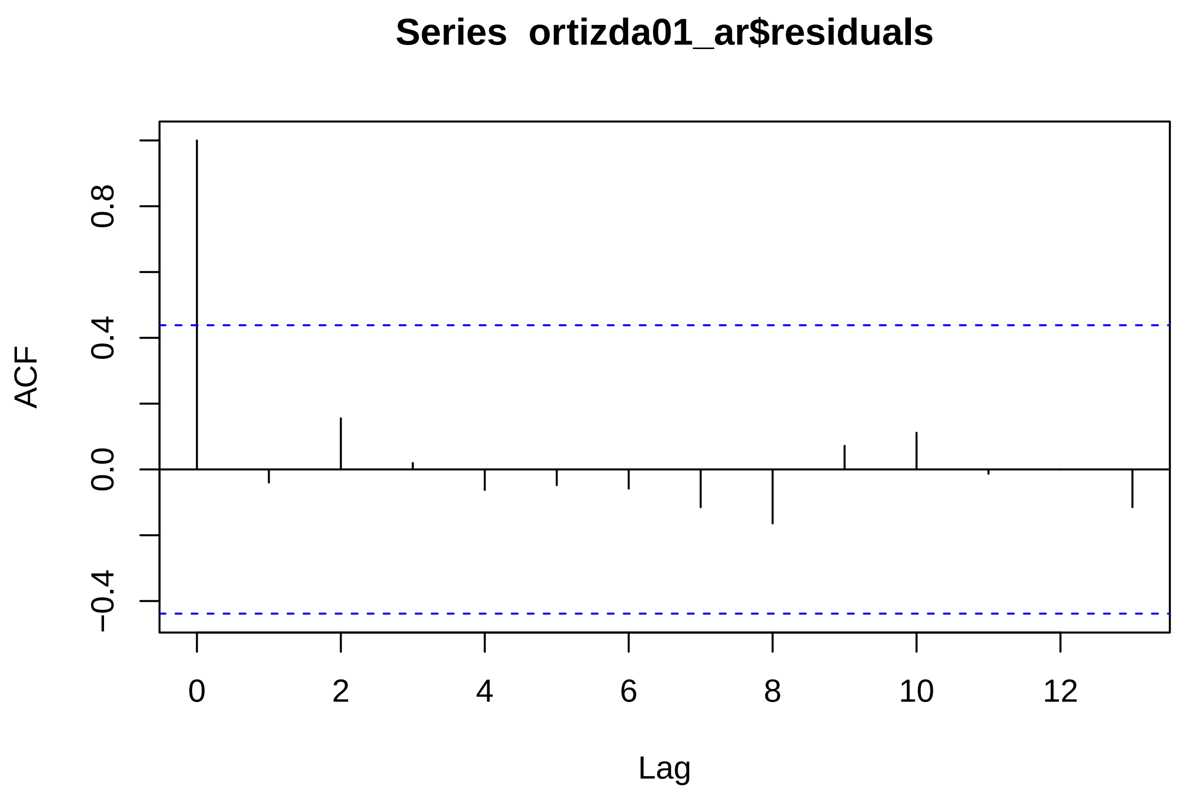
```
##
## Call:
## arima(x = ortizda01_tsbl$OPS, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.3196     1.0104
## s.e.  0.2102     0.0648
```

```
##
## sigma^2 estimated as 0.04073:  log likelihood = 3.58,  aic = -1.15
##
## Training set error measures:
##                       ME      RMSE       MAE       MPE     MAPE    MASE
## Training set 0.002359982 0.2018044 0.1266369 -16.25274 27.80769 0.87426
##                     ACF1
## Training set -0.03997079
```

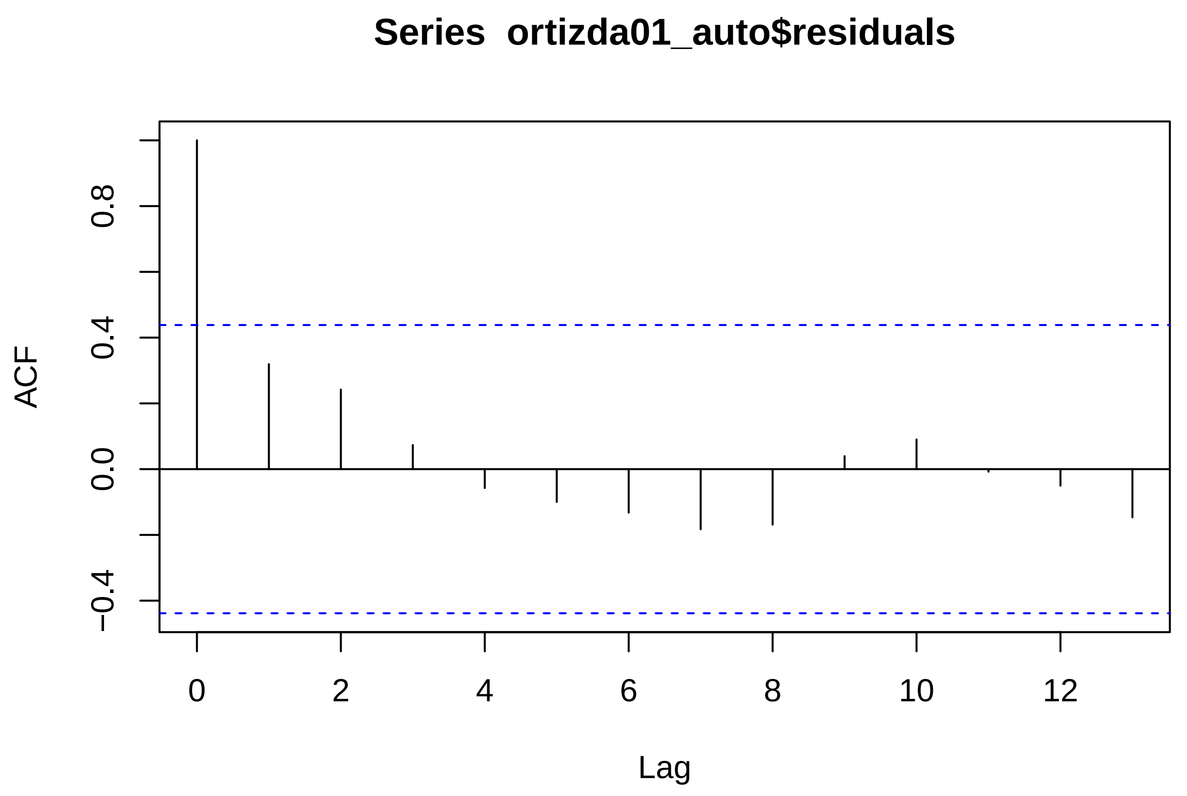
```
# Setting up auto.arima model
ortizda01_auto <- auto.arima(ortizda01_tsbl$OPS)
summary(ortizda01_auto)
```

```
## Series: ortizda01_tsbl$OPS
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##         mean
##       1.0093
## s.e.  0.0478
##
## sigma^2 = 0.04801:  log likelihood = 2.5
## AIC=-0.99   AICc=-0.29   BIC=1
##
## Training set error measures:
##                        ME      RMSE       MAE      MPE    MAPE      MASE
## Training set 4.342083e-14 0.2135707 0.1358915 -17.3698 29.0938 0.9381511
##                   ACF1
## Training set 0.3193526
```

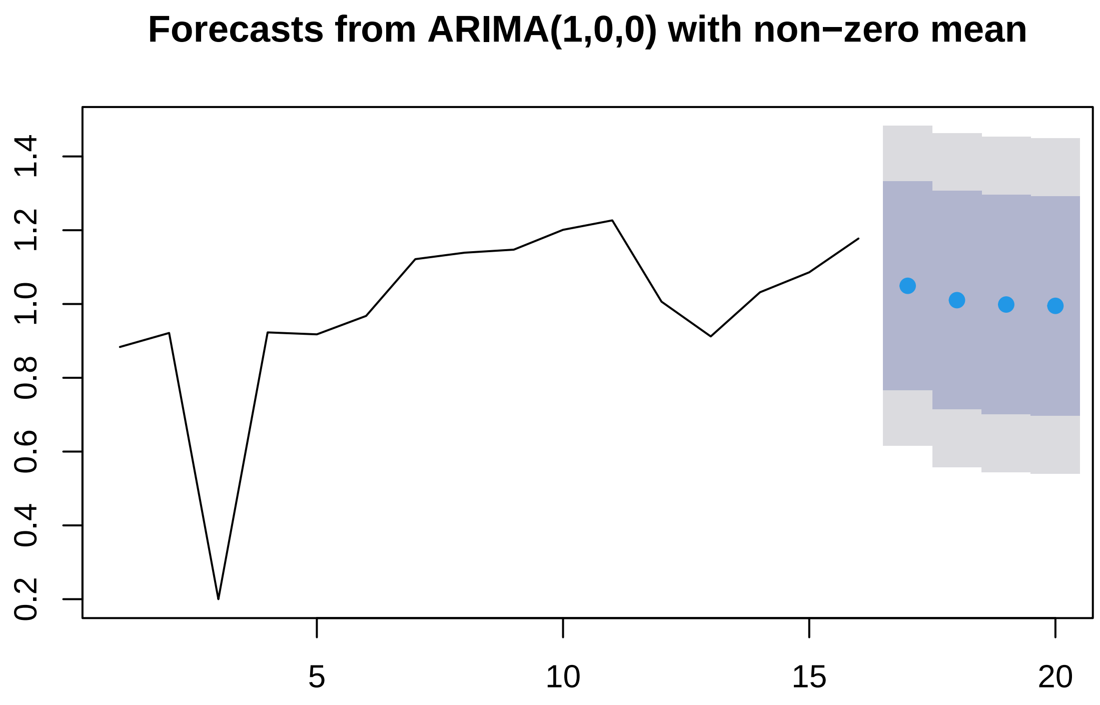
```
# Storing AIC values for both models
ARIMA_perf_AIC_table[4,2] <- ortizda01_ar$aic
ARIMA_perf_AIC_table[4,3] <- ortizda01_auto$aic

# Looking at residuals for the model
acf(ortizda01_ar$residuals)
```



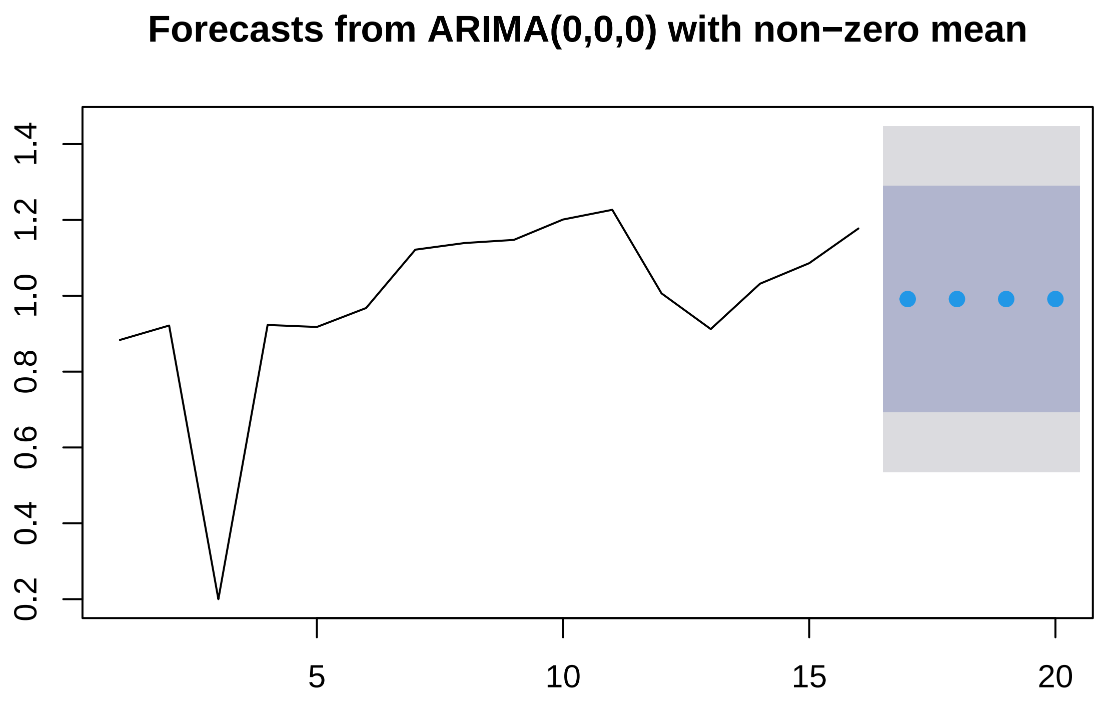
```
acf(ortizda01_auto$residuals)
```



```
# Setting up 20% of career for Testing Data and 80% of career for Training Data
ortizda01_tsbl_test <- ortizda01_tsbl %>%
  filter(ortizda01_tsbl$yearID >= 2013)

ortizda01_tsbl_train <- ortizda01_tsbl %>%
  filter(ortizda01_tsbl$yearID < 2013)

# Fitting Models with Training Data
ortizda01_ar100 <- arima(ortizda01_tsbl_train$OPS , order = c(1, 0, 0))

ortizda01_ar000 <- arima(ortizda01_tsbl_train$OPS , order = c(0, 0, 0))

# Using Models to predict on 20% of Players Career
ortizda01_fc_ar100 <- forecast(ortizda01_ar100, h = 4)

ortizda01_fc_ar000 <- forecast(ortizda01_ar000, h = 4)

# Plotting Predictions
plot(ortizda01_fc_ar100)
```



```
plot(ortizda01_fc_ar000)
```



```r
# Calculating RMSE
ortizda01_fc_ar100_mean <- ortizda01_fc_ar100$mean[1:4]
ortizda01_fc_ar000_mean <- ortizda01_fc_ar000$mean[1:4]

ortizda01_fc_ar100_acc <- accuracy(ortizda01_fc_ar100_mean, ortizda01_tsbl_test$OPS)
ortizda01_fc_ar000_acc <- accuracy(ortizda01_fc_ar000_mean, ortizda01_tsbl_test$OPS)

# Storing RMSE Values
ARIMA_perf_RMSE_table[4,2] <- ortizda01_fc_ar100_acc[1,2]
ARIMA_perf_RMSE_table[4,3] <- ortizda01_fc_ar000_acc[1,2]
```

Pete Rose Ar(1) set up and predictions.

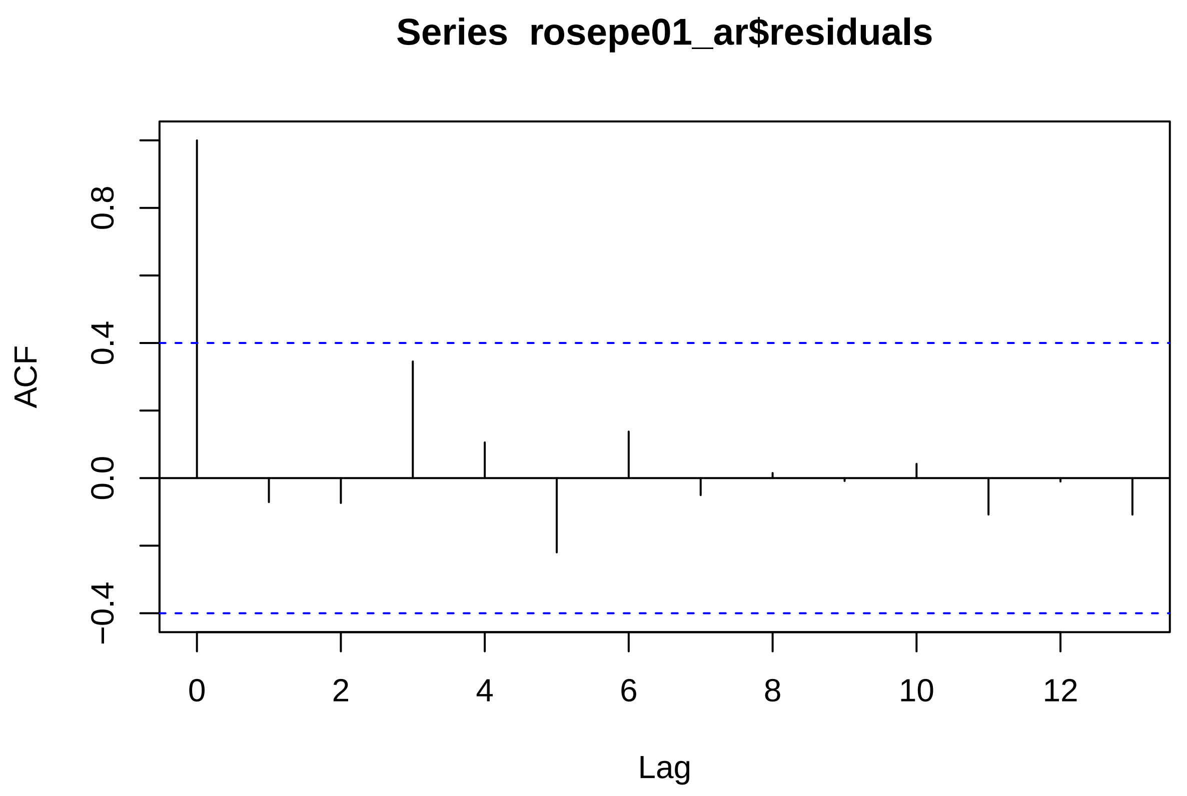
```r
# Setting up ar100 model
rosepe01_ar <- arima(rosepe01_tsbl$OPS , order = c(1, 0, 0))
summary(rosepe01_ar)
```

```
##
## Call:
## arima(x = rosepe01_tsbl$OPS, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.6779     0.8225
## s.e.  0.1631     0.0489
##
## sigma^2 estimated as 0.006454:  log likelihood = 26.15,  aic = -46.31
##
## Training set error measures:
##                      ME       RMSE        MAE        MPE     MAPE      MASE
## Training set 0.00210838 0.08033656 0.06430434 -0.7799731 8.005843 0.9254725
##                     ACF1
## Training set -0.07114259
```

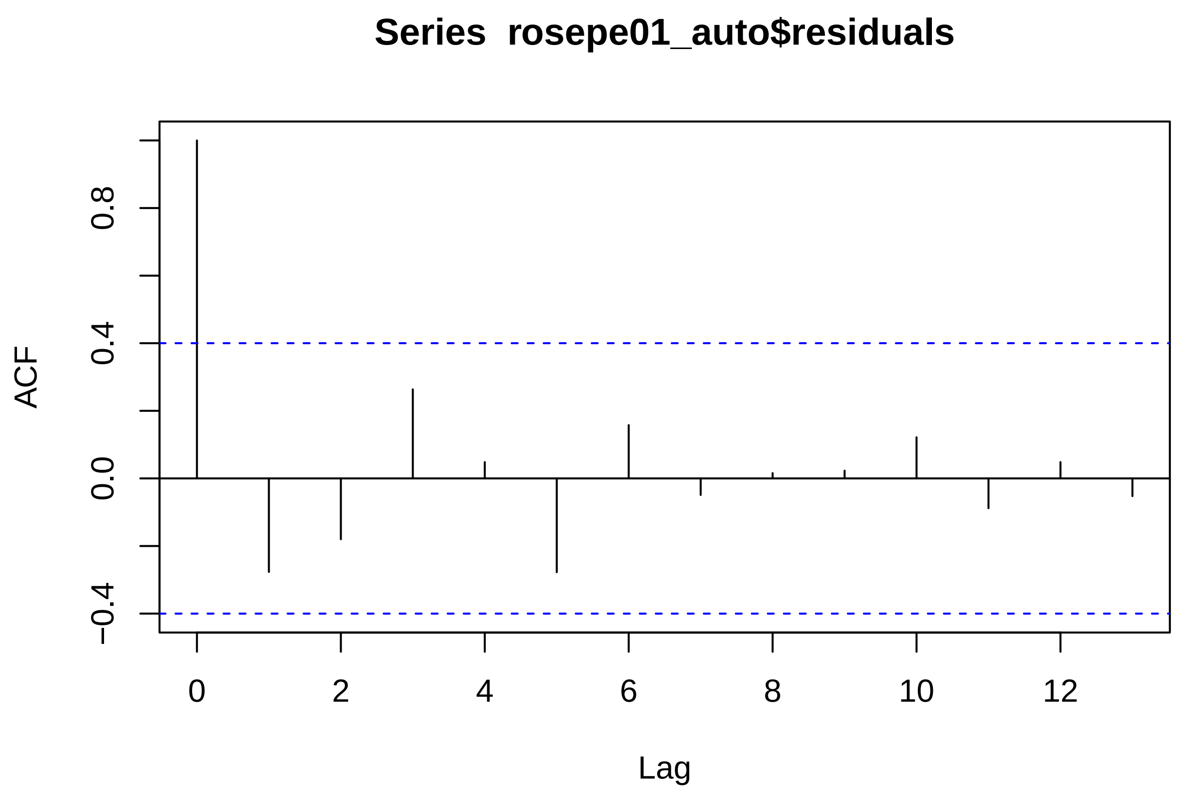
```
# Setting up auto.arima model
rosepe01_auto <- auto.arima(rosepe01_tsbl$OPS)
summary(rosepe01_auto)
```

```
## Series: rosepe01_tsbl$OPS
## ARIMA(0,1,0)
##
## sigma^2 = 0.007554:  log likelihood = 23.55
## AIC=-45.1   AICc=-44.91   BIC=-43.96
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE     MAPE      MASE
## Training set -0.005817814 0.08508395 0.06661963 -1.406319 8.242282 0.9587944
##                   ACF1
## Training set -0.2767143
```

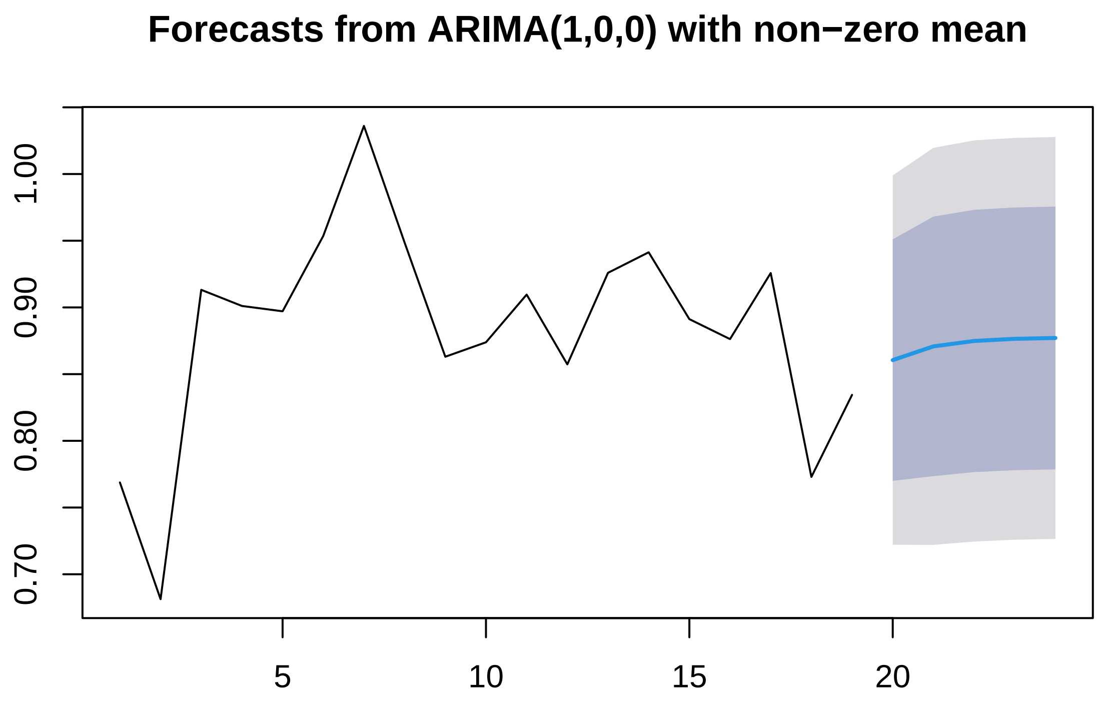
```
# Storing AIC values for both models
ARIMA_perf_AIC_table[5,2] <- rosepe01_ar$aic
ARIMA_perf_AIC_table[5,3] <- rosepe01_auto$aic

# Looking at residuals for the model
acf(rosepe01_ar$residuals)
```



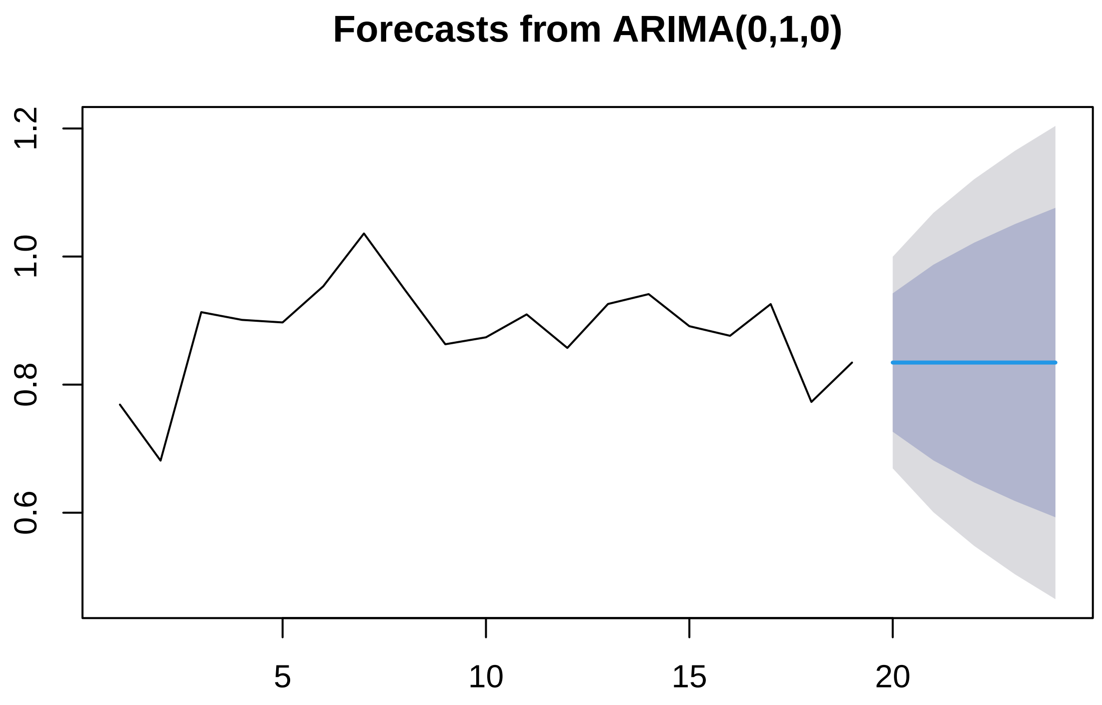
```
acf(rosepe01_auto$residuals)
```



```
# Setting up 20% of career for Testing Data and 80% of career for Training Data
rosepe01_tsbl_test <- rosepe01_tsbl %>%
  filter(rosepe01_tsbl$yearID >= 1982)

rosepe01_tsbl_train <- rosepe01_tsbl %>%
  filter(rosepe01_tsbl$yearID < 1982)

# Fitting Model with Training Data
rosepe01_ar100 <- arima(rosepe01_tsbl_train$OPS , order = c(1, 0, 0))

rosepe01_ar010 <- arima(rosepe01_tsbl_train$OPS , order = c(0, 1, 0))

# Using Model to predict on 20% of Players Career
rosepe01_fc_ar100 <- forecast(rosepe01_ar100, h = 5)

rosepe01_fc_ar010 <- forecast(rosepe01_ar010, h = 5)

# Plotting Predictions
plot(rosepe01_fc_ar100)
```



```
plot(rosepe01_fc_ar010)
```



```
# Calculating RMSE
rosepe01_fc_ar100_mean <- rosepe01_fc_ar100$mean[1:5]
rosepe01_fc_ar010_mean <- rosepe01_fc_ar010$mean[1:5]

rosepe01_fc_ar100_acc <- accuracy(rosepe01_fc_ar100_mean, rosepe01_tsbl_test$OPS)
rosepe01_fc_ar010_acc <- accuracy(rosepe01_fc_ar010_mean, rosepe01_tsbl_test$OPS)

# Storing RMSE Values
ARIMA_perf_RMSE_table[5,2] <- rosepe01_fc_ar100_acc[1,2]
ARIMA_perf_RMSE_table[5,3] <- rosepe01_fc_ar010_acc[1,2]
```

Barry Bonds Ar(1) set up and predictions.

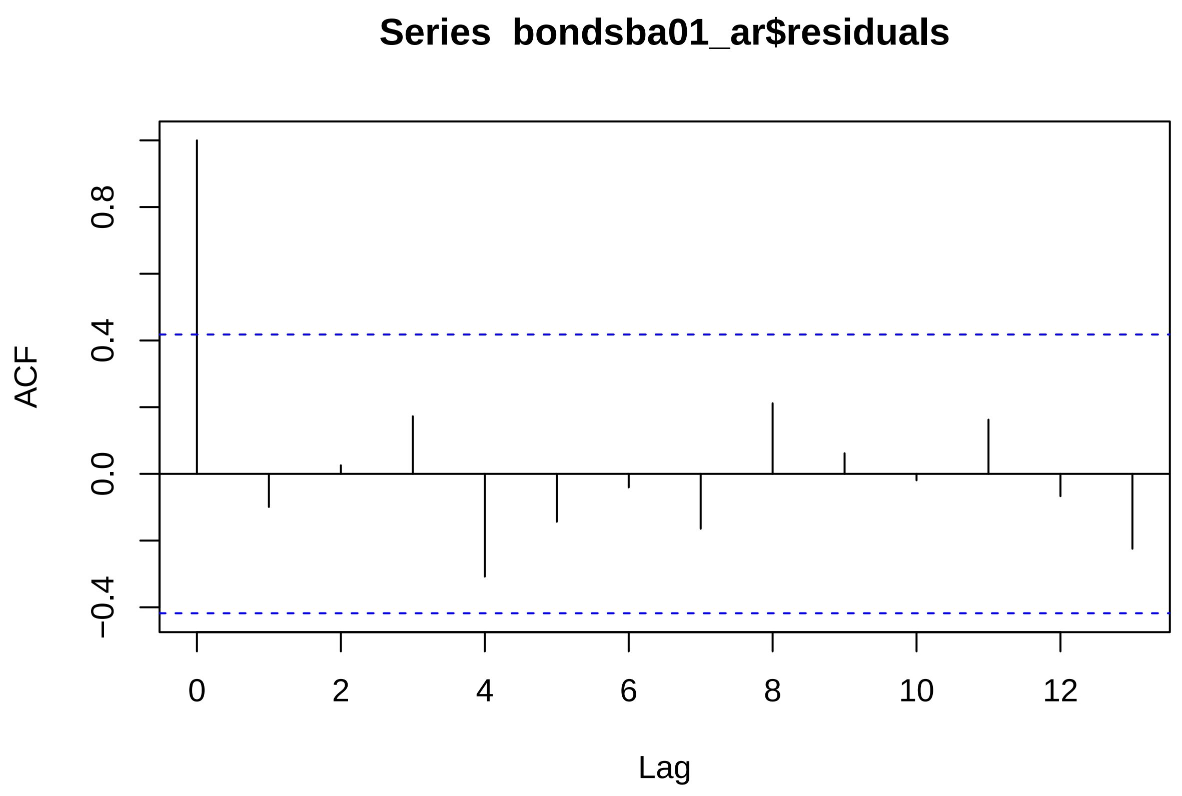
```
# Setting up ar100 model
bondsba01_ar <- arima(bondsba01_tsbl$OPS , order = c(1, 0, 0))
summary(bondsba01_ar)
```

```
##
## Call:
## arima(x = bondsba01_tsbl$OPS, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.7539     1.1626
## s.e.  0.1395     0.1101
##
## sigma^2 estimated as 0.01961:  log likelihood = 11.61,  aic = -17.22
##
## Training set error measures:
##                      ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.01531994 0.1400356 0.1070755 0.04852601 8.846623 0.9398822
##                     ACF1
## Training set -0.09908485
```

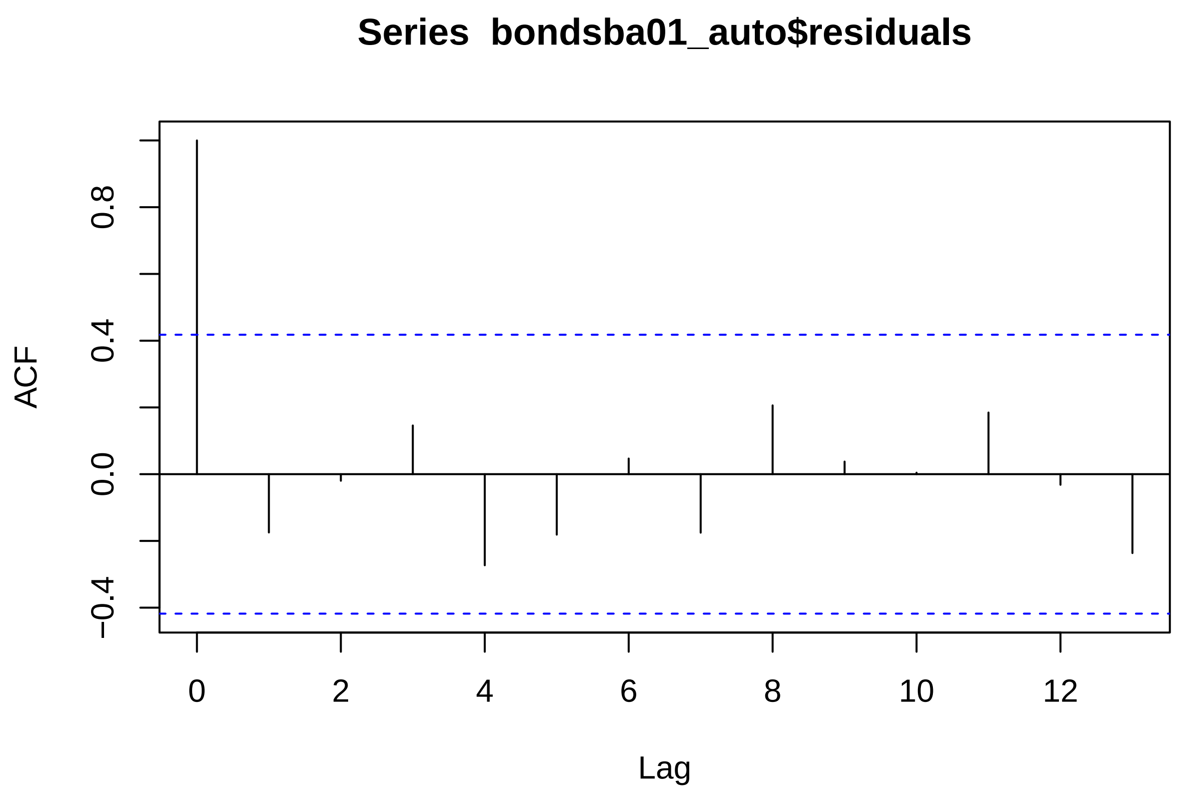
```
# Setting up auto.arima model
bondsba01_auto <- auto.arima(bondsba01_tsbl$OPS)
summary(bondsba01_auto)
```

```
## Series: bondsba01_tsbl$OPS
## ARIMA(0,1,0)
##
## sigma^2 = 0.02234:  log likelihood = 10.12
## AIC=-18.24   AICc=-18.02   BIC=-17.19
##
## Training set error measures:
##                      ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.01426705 0.1460224 0.1087849 0.763356 8.792082 0.9548867
##                    ACF1
## Training set -0.1747355
```

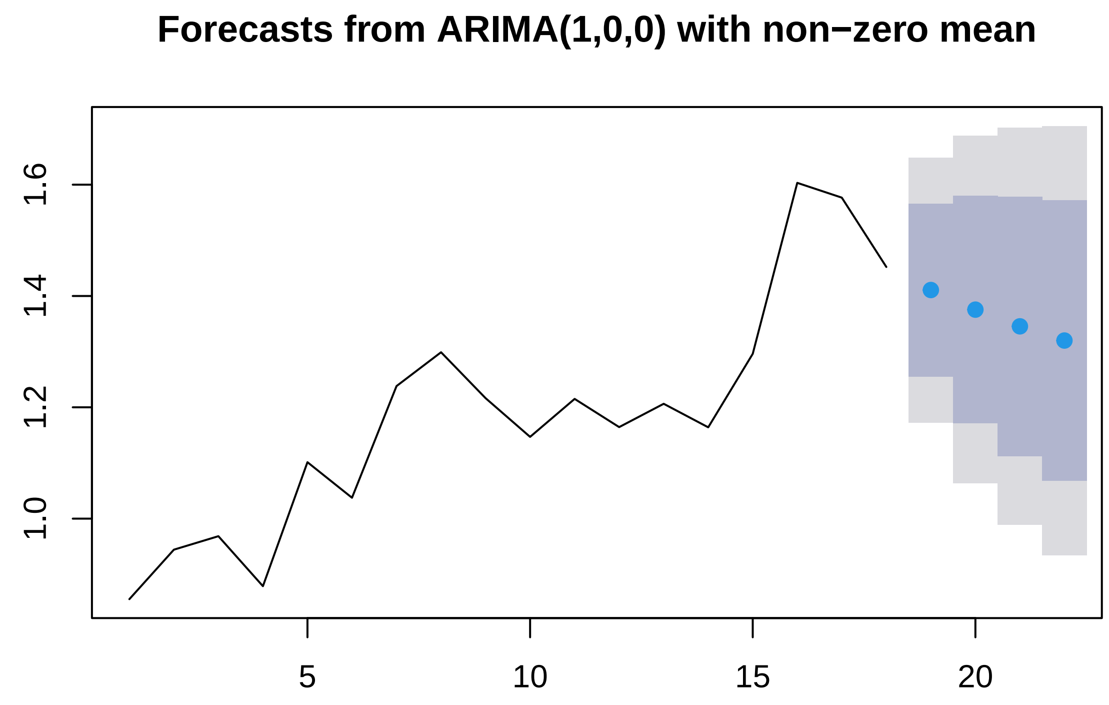
```
# Storing AIC values for both models
ARIMA_perf_AIC_table[6,2] <- bondsba01_ar$aic
ARIMA_perf_AIC_table[6,3] <- bondsba01_auto$aic

# Looking at residuals for the models
acf(bondsba01_ar$residuals)
```



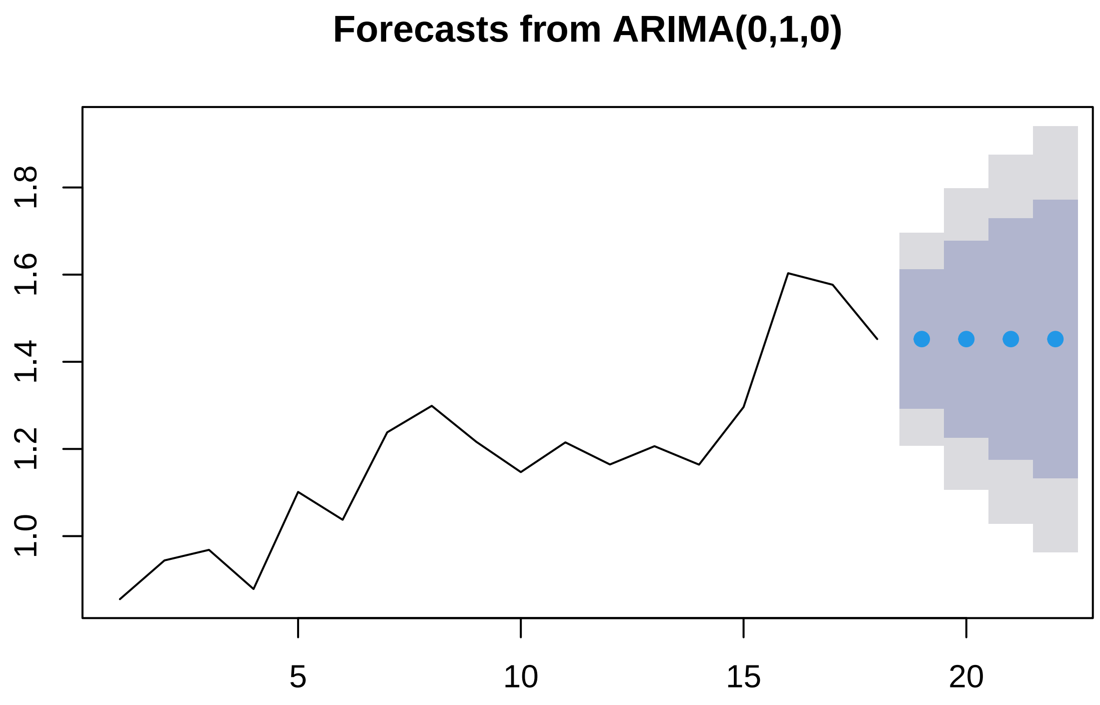
```
acf(bondsba01_auto$residuals)
```



```
# Setting up 20% of career for Testing Data and 80% of career for Training Data
bondsba01_tsbl_test <- bondsba01_tsbl %>%
  filter(bondsba01_tsbl$yearID >= 2004)

bondsba01_tsbl_train <- bondsba01_tsbl %>%
  filter(bondsba01_tsbl$yearID < 2004)

# Fitting Models with Training Data
bondsba01_ar100 <- arima(bondsba01_tsbl_train$OPS , order = c(1, 0, 0))

bondsba01_ar010 <- arima(bondsba01_tsbl_train$OPS , order = c(0, 1, 0))

# Using Model to predict on 20% of Players Career
bondsba01_fc_ar100 <- forecast(bondsba01_ar100, h = 4)

bondsba01_fc_ar010 <- forecast(bondsba01_ar010, h = 4)

# Plotting Predictions
plot(bondsba01_fc_ar100)
```



```
plot(bondsba01_fc_ar010)
```



```
# Calculating RMSE
bondsba01_fc_ar100_mean <- bondsba01_fc_ar100$mean[1:4]
bondsba01_fc_ar010_mean <- bondsba01_fc_ar010$mean[1:4]

bondsba01_fc_ar100_acc <- accuracy(bondsba01_fc_ar100_mean, bondsba01_tsbl_test$OPS)
bondsba01_fc_ar010_acc <- accuracy(bondsba01_fc_ar010_mean, bondsba01_tsbl_test$OPS)

# Storing RMSE Values
ARIMA_perf_RMSE_table[6,2] <- bondsba01_fc_ar100_acc[1,2]
ARIMA_perf_RMSE_table[6,3] <- bondsba01_fc_ar010_acc[1,2]
```

RMSE Performance Table

```
ARIMA_perf_RMSE_table
```

```
##           Player  p(1,0,0)      AUTO
## 1     Hank Aaron 0.2190332 0.2190332
## 2 Ichiro Suzuki 0.3216339 0.2496140
## 3   Derek Jeter 0.2073190 0.2073190
## 4   David Ortiz 0.1007935 0.1129279
## 5     Pete Rose 0.1826573 0.1468129
## 6   Barry Bonds 0.1867246 0.2591741
```

```
summary(ARIMA_perf_RMSE_table)
```

```
##     Player             p(1,0,0)          AUTO       
##  Length:6           Min.   :0.1008   Min.   :0.1129  
##  Class :character   1st Qu.:0.1837   1st Qu.:0.1619  
##  Mode  :character   Median :0.1970   Median :0.2132  
##                     Mean   :0.2030   Mean   :0.1991  
##                     3rd Qu.:0.2161   3rd Qu.:0.2420  
##                     Max.   :0.3216   Max.   :0.2592
```

AIC Model Performance Table

```
ARIMA_perf_AIC_table
```

```
##          Player   p(1,0,0)        AUTO
## 1    Hank Aaron -29.836156 -29.8361558
## 2 Ichiro Suzuki -18.106034 -21.8929371
## 3   Derek Jeter -30.145943 -30.1459428
## 4   David Ortiz  -1.152944  -0.9939457
## 5     Pete Rose -46.309186 -45.0994101
## 6   Barry Bonds -17.224376 -18.2355040
```

```
summary(ARIMA_perf_AIC_table)
```

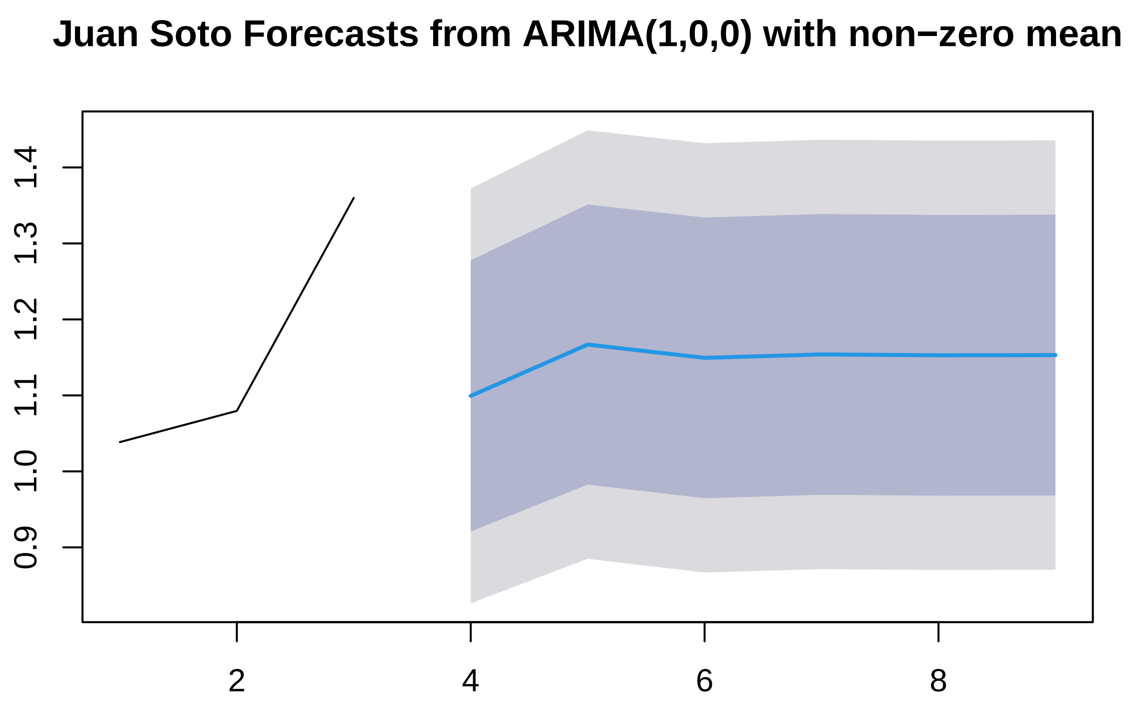
```
##     Player             p(1,0,0)           AUTO        
##  Length:6           Min.   :-46.309   Min.   :-45.099  
##  Class :character   1st Qu.:-30.069   1st Qu.:-30.069  
##  Mode  :character   Median :-23.971   Median :-25.865  
##                     Mean   :-23.796   Mean   :-24.367  
##                     3rd Qu.:-17.445   3rd Qu.:-19.150  
##                     Max.   : -1.153   Max.   : -0.994
```

Predictions on Current Players

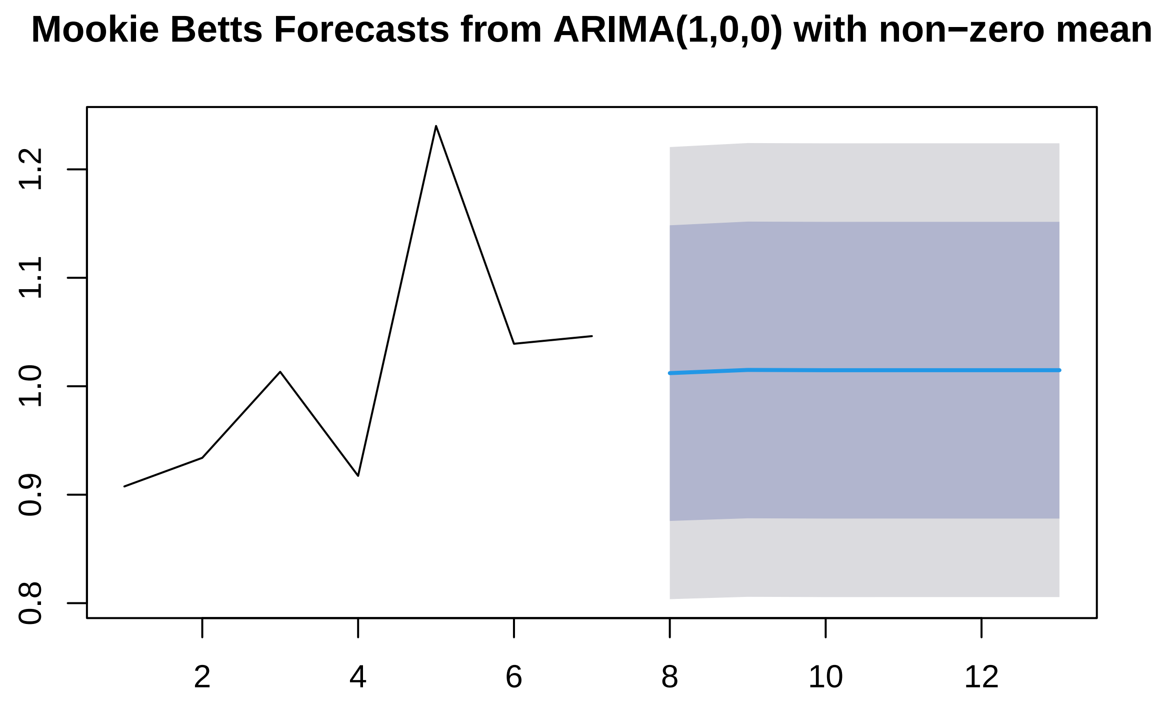
Using the Ar(1) model I will make predictions on six current players; Juan Soto, Mookie Betts, Anthony Rendon, Bryce Harper, Mike Trout, and Nolan Arenado. I will be using the Ar(1) models to predict the OPS of each player for the next six seasons.

```
Batting_tsbl %>%
  filter(playerID == "sotoju01") -> sotoju01_tsbl

Batting_tsbl %>%
  filter(playerID == "bettsmo01") -> bettsmo01_tsbl

Batting_tsbl %>%
  filter(playerID == "rendoan01") -> rendoan01_tsbl
```

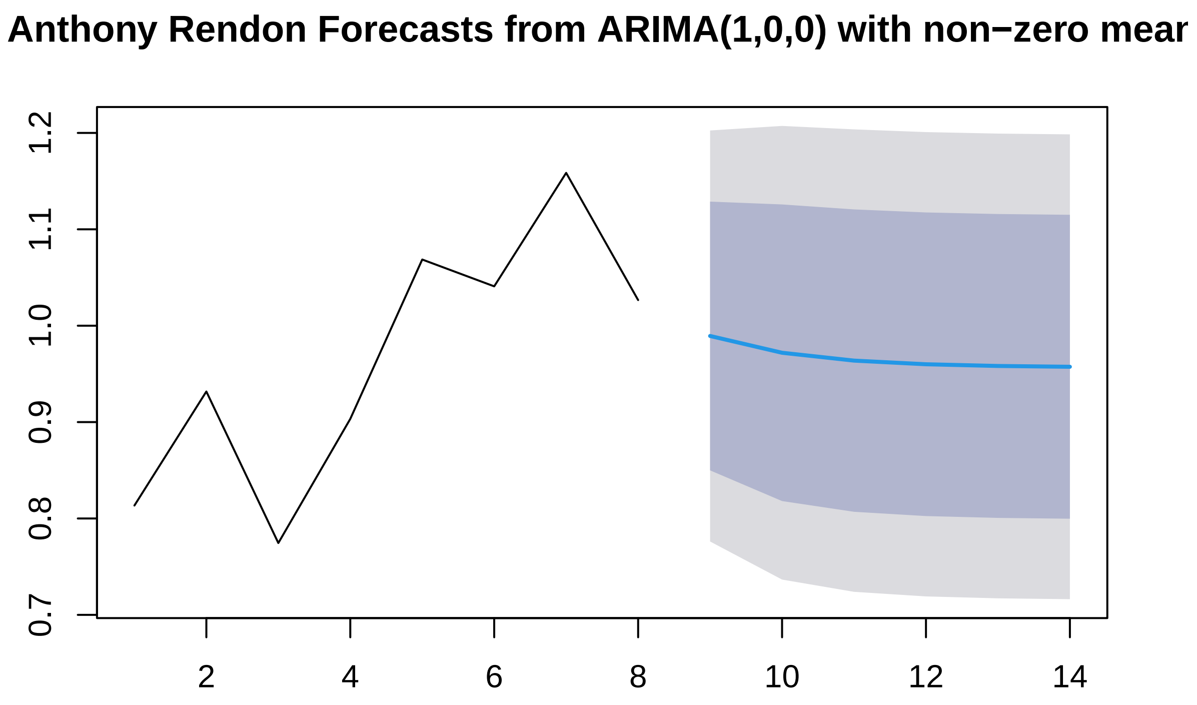
```r
Batting_tsbl %>%
  filter(playerID == "harpebr03") -> harpebr03_tsbl

Batting_tsbl %>%
  filter(playerID == "troutmi01") -> troutmi01_tsbl

Batting_tsbl %>%
  filter(playerID == "arenano01") -> arenano01_tsbl

# Building AR(1) models for each player
sotoju01_ar <- arima(sotoju01_tsbl$OPS , order = c(1, 0, 0))
bettsmo01_ar <- arima(bettsmo01_tsbl$OPS , order = c(1, 0, 0))
rendoan01_ar <- arima(rendoan01_tsbl$OPS , order = c(1, 0, 0))
harpebr03_ar <- arima(harpebr03_tsbl$OPS , order = c(1, 0, 0))
troutmi01_ar <- arima(troutmi01_tsbl$OPS , order = c(1, 0, 0))
arenano01_ar <- arima(arenano01_tsbl$OPS , order = c(1, 0, 0))

# Forecasting next six seasons
sotoju0_fc_ar <- forecast(sotoju01_ar, h = 6)
bettsmo01_fc_ar <- forecast(bettsmo01_ar, h = 6)
rendoan01_fc_ar <- forecast(rendoan01_ar, h = 6)
harpebr03_fc_ar <- forecast(harpebr03_ar, h = 6)
troutmi0_fc_ar <- forecast(troutmi01_ar, h = 6)
arenano01_fc_ar <- forecast(arenano01_ar, h = 6)

plot(sotoju0_fc_ar, main = "Juan Soto Forecasts from ARIMA(1,0,0) with non-zero mean")
```



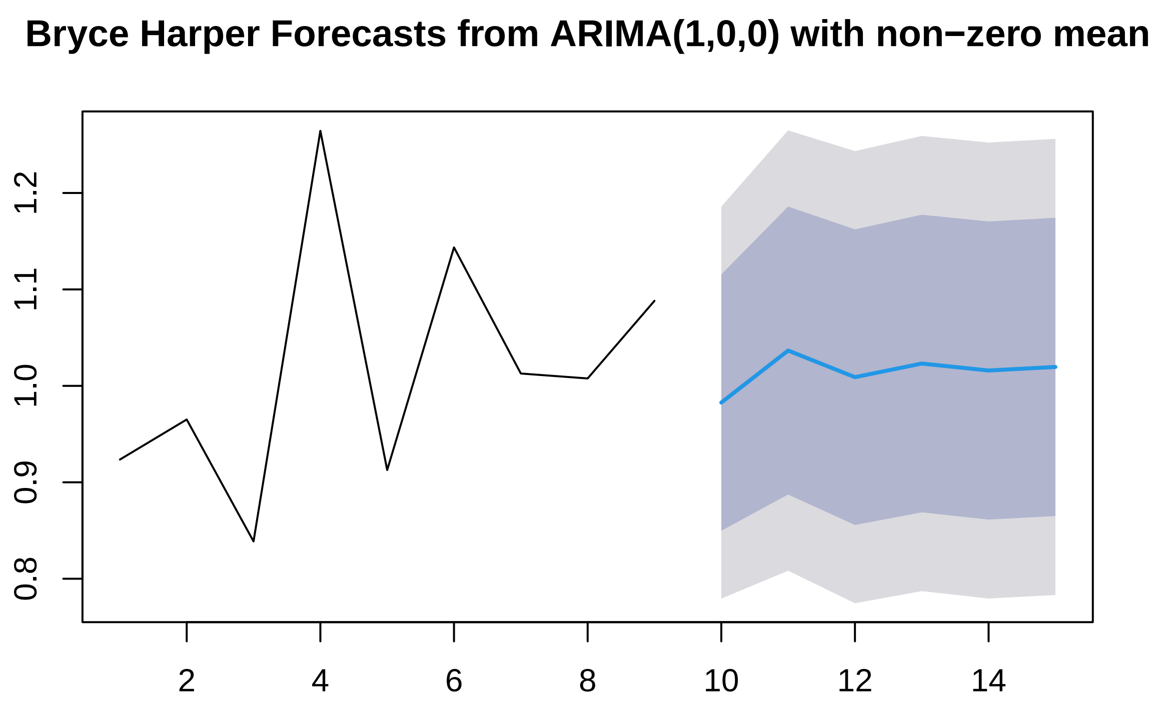
```
plot(bettsmo01_fc_ar, main = "Mookie Betts Forecasts from ARIMA(1,0,0) with non-zero mean")
```



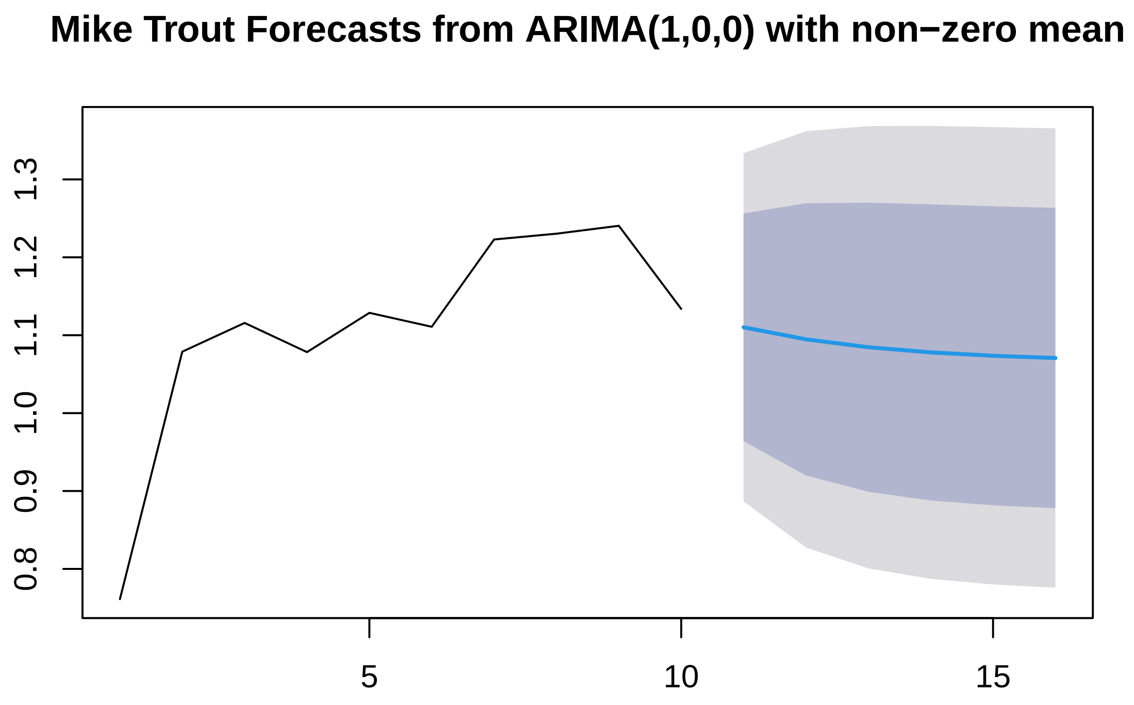
```r
plot(rendoan01_fc_ar, main = "Anthony Rendon Forecasts from ARIMA(1,0,0) with non-zero mean")
```



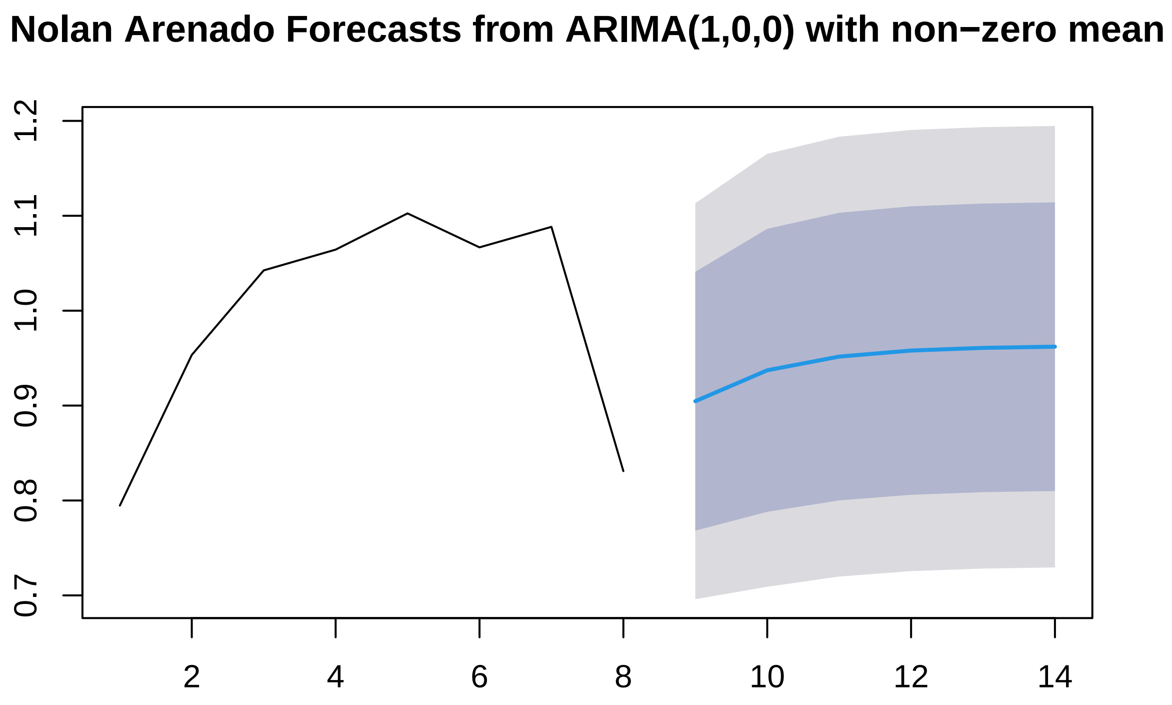
```
plot(harpebr03_fc_ar, main = "Bryce Harper Forecasts from ARIMA(1,0,0) with non-zero mean")
```



```r
plot(troutmi0_fc_ar, main = "Mike Trout Forecasts from ARIMA(1,0,0) with non-zero mean")
```



```r
plot(arenano01_fc_ar, main = "Nolan Arenado Forecasts from ARIMA(1,0,0) with non-zero mean")
```



Conclusion and Future Work

Looking at the results of the Ar(1) and auto.arima models we see that each of them have acf plots that have residuals that are not significantly correlated with each other and are ready to be used to forecast. Using the auto.arima model I found that the models for Hank Aaron and Derek Jeter were indeed Ar(1) models as confirmed by auto.arima while the four others were a variety of other ARIMA models. Using both models I found the RMSE and the AIC of each players Ar(1) and auto.arima model to compare their performance in predicting the last 20% of the players career. Both models for each player performed well in predicting as the RMSE for every prediction was lower than 0.4 and the AIC for each model was below 0 and negative. While the auto.arima models do make better predictions when it comes to their RMSE score the AR(1) models do a better job at explaining the most variation in the data.

Although this work yielded a model that can predict the OPS statistic for there is still a lot of future work to be done to further improve this model. Future work includes to further look at the dataset and try to develop a better model that can take more factors into consideration to make better predictions. The Ar(1) models was also developed using former/retired players and using the last 20% of their careers to test the models predictions when usually players start to decline in their performance. Future work would look at developing maybe separate models for players at the beginning, middle, and end of their careers to try and capture unique patterns that arise in those different stages.

References

Lahman, Sean. 2020. Baseball Database. https://www.seanlahman.com/baseball-archive/statistics/. This database is copyright 1996-2022 by Sean Lahman.