Predicting MLB Games

Francisco Robles

April 29, 2019

Load Packages, Load Data, and Clean Data

```
library(tidyverse)
library(teamcolors)
library(zoo)
library(xts)
library(forecast)
library(magrittr)
```

```
Teams = read_csv('Teams.csv')
schedule_2018 = read.table('schedule_2018.TXT', sep = ',')
```

Prediction Metric

To predict games, I made the conscious choice to only go back to 1998, which was the last time that the MLB expanded, going from 28 to 30 teams.

```
Teams %>%
  filter(yearID >= as.Date('1998-01-01')) -> Teams
```

There are many differenct current predictors to try and predict how a team should have done during a season. Using just the basic Win% (Wins / Total Games) isn't really the best way to show how good a team truly is. If a team is .500, but loses all its games by 10 runs and wins all of its games by only 1 run, then they are most likely just having a some luck to go their way. This is why a better metric is to use Runs & Runs allowed as a metric for success. There are 5 main public ways that people use to try and predict how good a team actually is, all of which use Runs and Runs allowed. I used the 2018 season to see which one was the best at predicting games.

I then found the RMSE of each predictor and compared it to what each team's actual winning percentage was and multiplied it by 162 to see how off the predictors were from how many games the teams actually won

```
Teams %>%
  summarize(pyt_rmse = sqrt(sum((pyt_resid)^2)/629),
            opt_rmse = sqrt(sum((opt_resid)^2)/629),
            port_rmse = sqrt(sum((port_resid)^2)/629),
            pat_rmse = sqrt(sum((pat_resid)^2)/629),
            lin_rmse = sqrt(sum((lin_resid)^2)/629)) -> rmse
rmse*162
```

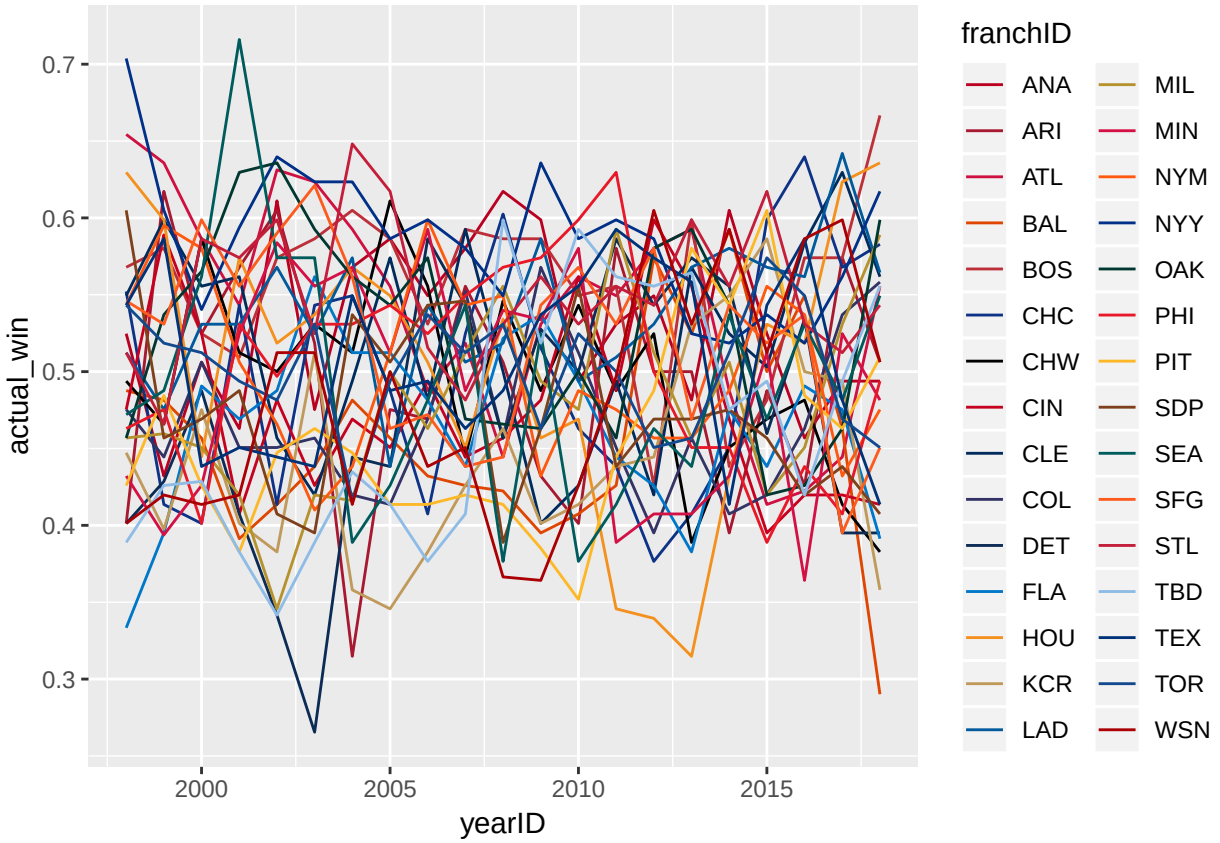
```
##   pyt_rmse opt_rmse port_rmse pat_rmse lin_rmse
## 1 4.122079  4.03252  4.011891 4.013318 4.203769
```

All of them were very close to each other, all around 4, so it doesn't seem like any of them are bad, so I decided to just go with the smallest one which was port_win predictor.

Visualization

To try and visualize how the data would look, I tried making a multicolored scatterplot to try and visualize the data, it did not end up too well

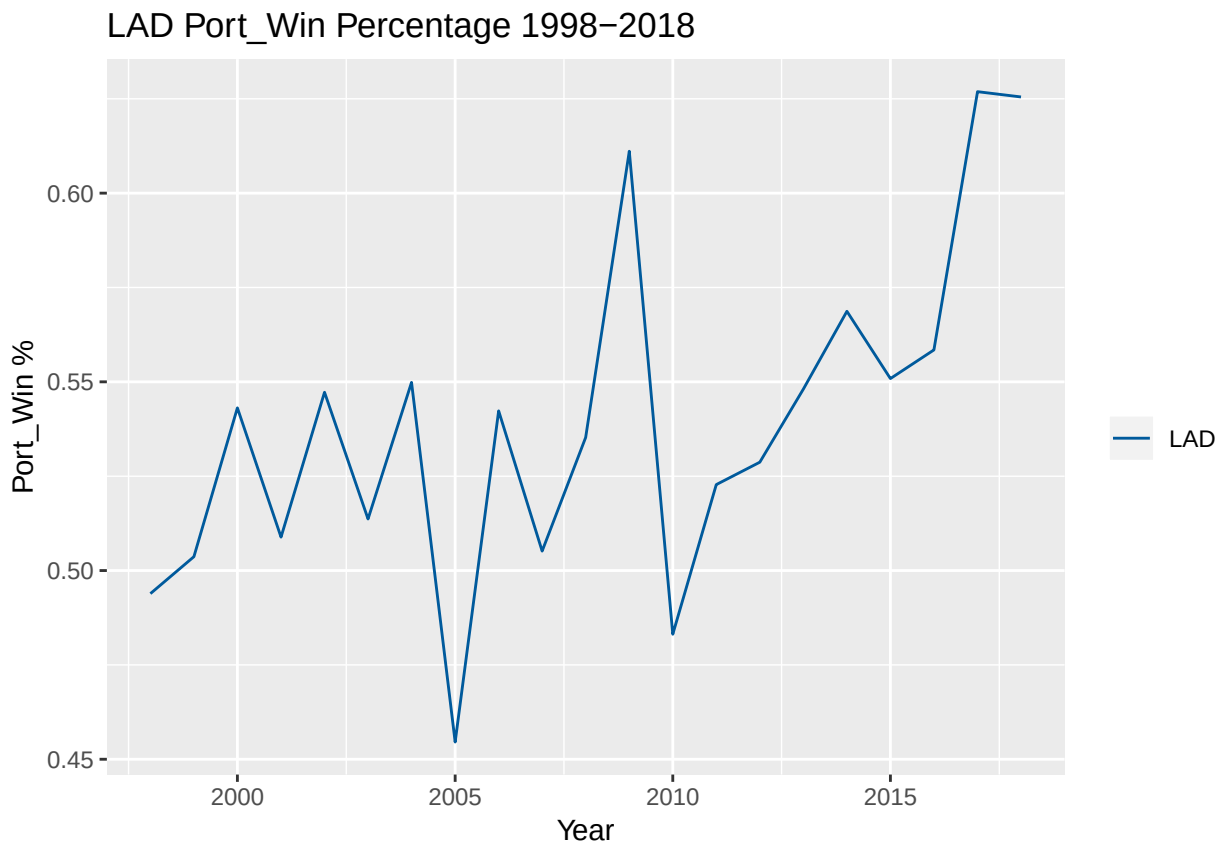
```
ggplot(Teams, aes(yearID, actual_win, color = franchID)) +
  geom_line() +
  scale_color_manual(values = mlb.palette)
```



Yeah, it didn't turn out too pretty. But once you split it apart and only have each individual team on their own plot, it looks somewhat nice

Here's one for the Dodgers for instance.

```
plot.list$LAD
```

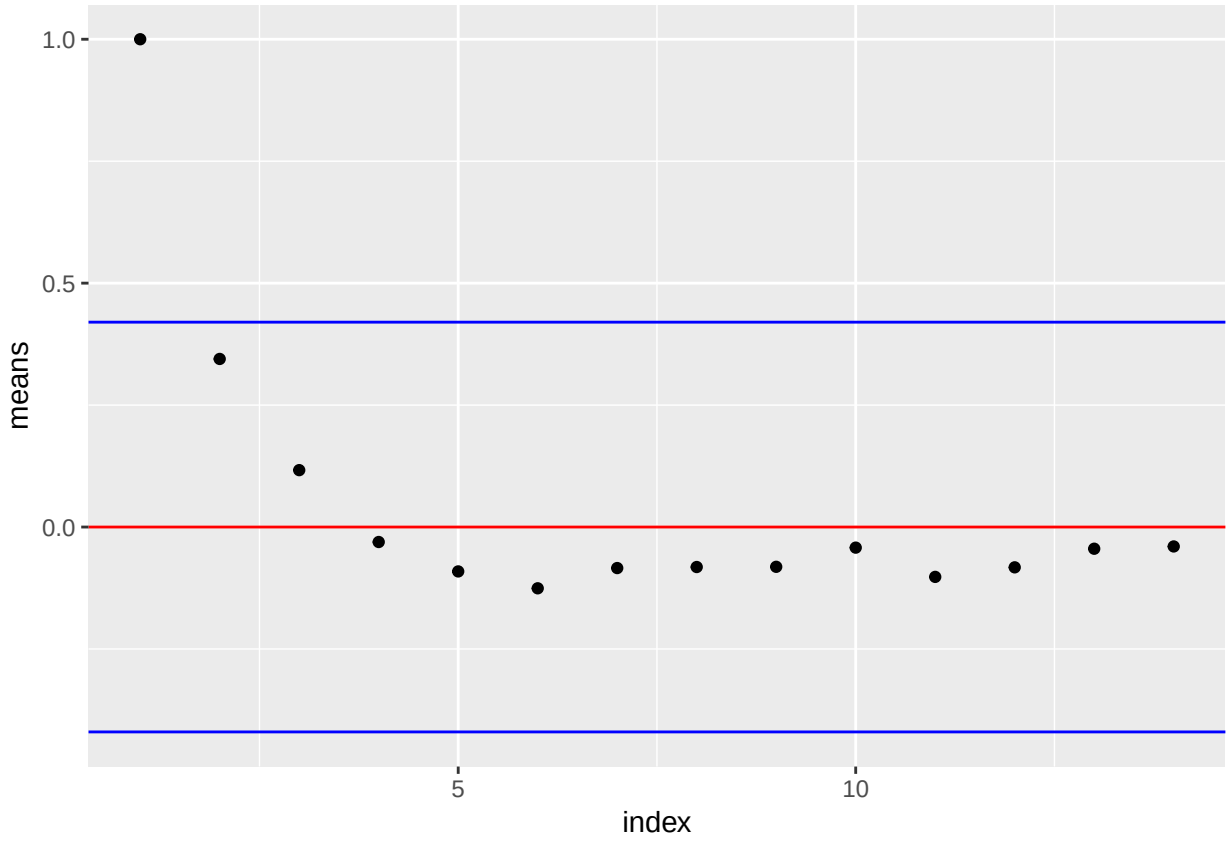


Time Series Paramaeter Estimation

From just looking at that one graph, it is kind of tough to see how we can use our previous data to try and predict. Luckily, there is a time series model called an ARIMA model, that can help us fit a model for us. They depend on three different parameters (AR, I, MA), hence ARIMA. For our case, the I=0. But to find out the AR and MA, there are different ways to try and find what they will be, with an ACF and PACF plot being the most common.

So, I ran an ACF for each of the teams and then took the average coefficients at each of the interval to come up with this plot.

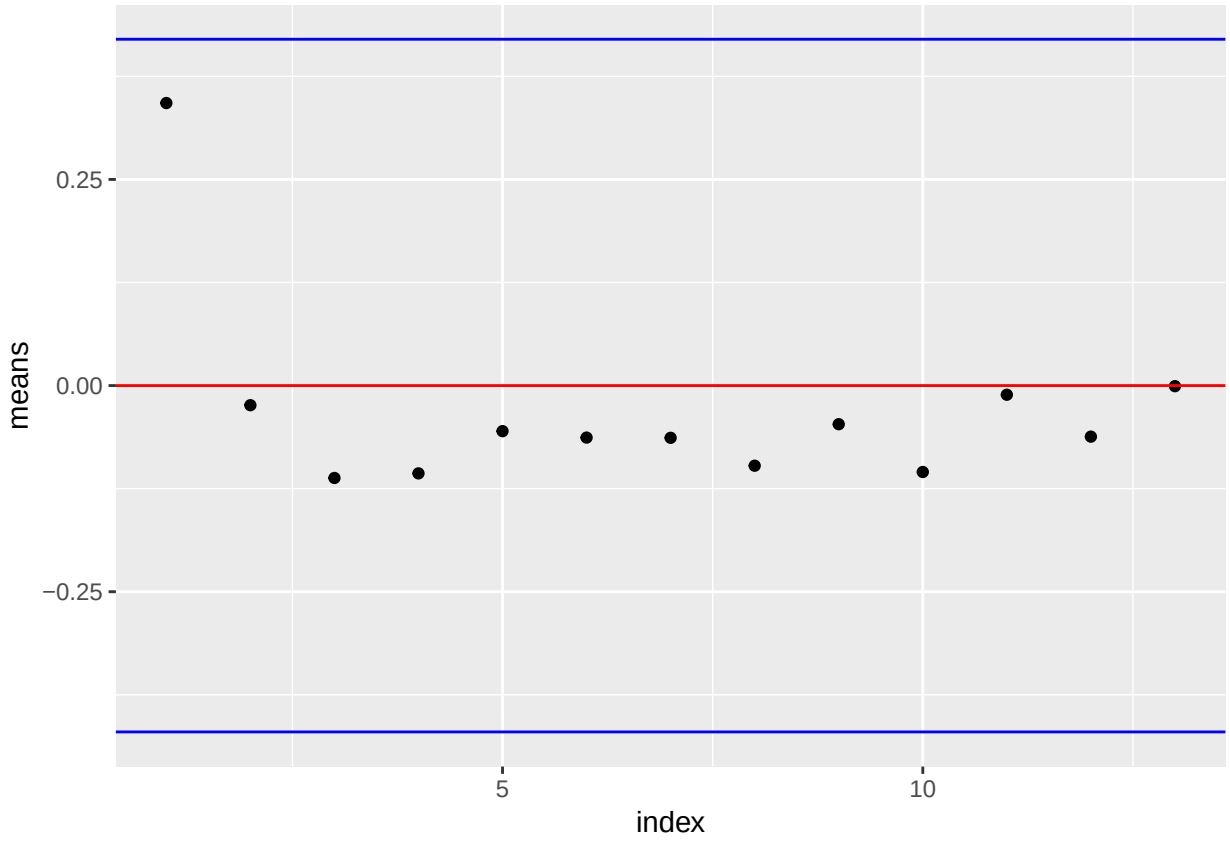
```
ggplot(acf.means, aes(index, means)) +
  geom_point() +
  geom_hline(color = 'red', yintercept = 0) +
  geom_hline(color = 'blue', yintercept = 0.42) +
  geom_hline(color = 'blue', yintercept = -0.42)
```



The first point doesn't mean much, but we are looking for points that fall outside of the blue bars. No other point does, which isn't a good sign. However, I am going to say that the first one seems close enough and say that it matters. This tells us that there is most likely an AR=1 model

The next thing to do is check the PACF. I ran through the same process that I mentioned above for this as well. Here is the result.

```
ggplot(pacf.means, aes(index, means)) +
  geom_point() +
  geom_hline(color = 'red', yintercept = 0) +
  geom_hline(color = 'blue', yintercept = 0.42) +
  geom_hline(color = 'blue', yintercept = -0.42)
```



For this case, the first point does matter to us. Again, no point falls outside the blue line range, but I am going to say that it is important, which would tell us that we have a MA=1 process.

Time Series Analysis

Since our data seems to suggest an ARIMA(1,0,1) model, I ran one for each of the 30 teams in the MLB from the data from 2017 and before. I then took the mean of each of our 3 coefficients (AR1, MA1, Intercept) and used that as the coefficients of my time series. I then used these values to predict the winning percentages of these teams in 2018. This was my result.

```
as.data.frame(pred.list.2018) %>%
  gather('Team', 'Pred_Win_Pct') %>%
  arrange(desc(Pred_Win_Pct))
```

```
##    Team Pred_Win_Pct
## 1   CLE    0.6239173
## 2   NYY    0.6056460
## 3   LAD    0.6017688
## 4   HOU    0.6000174
## 5   ARI    0.5965112
## 6   WSN    0.5777823
## 7   BOS    0.5664315
## 8   CHC    0.5641370
## 9   COL    0.5566416
## 10  MIN    0.5500887
## 11  STL    0.5491868
## 12  MIL    0.5487629
## 13  ANA    0.5325452
```

```
## 14  TBD    0.5300534
## 15  TEX    0.5250562
## 16  SEA    0.5199599
## 17  FLA    0.5173684
## 18  PIT    0.5084615
## 19  PHI    0.5073456
## 20  ATL    0.5064253
## 21  OAK    0.5060941
## 22  KCR    0.5002929
## 23  CIN    0.4977814
## 24  BAL    0.4945201
## 25  TOR    0.4917925
## 26  CHW    0.4911042
## 27  NYM    0.4823853
## 28  DET    0.4738442
## 29  SFG    0.4734755
## 30  SDP    0.4552266
```

Bradley-Terry Model

I will be using a model called the Bradley-Terry model to try and predict games. Essentially what it does is you assign each team a talent score and it uses those talent scores to try and predict if a team would win that game or not. The Bradley-Terry model suggests that you have your talents follow a N(0,.2) distribution (0 mean and .2 standard deviation). I converted my predicted winning percentages to have that and this is the result

```
talent.2018
```

```
##          ANA          ARI          ATL          BAL          BOS
##  0.003231628  0.288598681 -0.113295066 -0.166406953  0.154406156
##          CHW          CHC          CIN          CLE          COL
## -0.181646050  0.144169816 -0.151857648  0.410863578  0.110731434
##          DET          FLA          HOU          KCR          LAD
## -0.258646643 -0.064475292  0.304240603 -0.140653275  0.312054046
##          MIL          MIN          WSN          NYY          NYM
##  0.075582499  0.081497301  0.205045012  0.329351197 -0.220543087
##          OAK          PHI          PIT          SDP          SEA
## -0.114772828 -0.109189212 -0.104210986 -0.341704228 -0.052914155
##          SFG          STL          TBD          TEX          TOR
## -0.260291726  0.077473697 -0.007884587 -0.030178509 -0.178575400
```

I wrote a function that will simulate the entire regular season and playoffs for the season. I ran this simulation 10,000 times to make sure my results weren't flukey

```
results.list = readRDS('results_list')
```

Analysis of Prediction

I am going to use RMSE to see how well my prediction did for this as well. I will take the average amount of games each of the teams won over the 10,000 simulations and use that as my prediction for the 2018 season.

```
sim.rmse = sqrt(sum((resid.df$Actual_Pct - resid.df$Avg.Win)^2)/30)
sim.rmse * 162
```

```
## [1] 10.83675
```

That is really, really bad. I'm essentially 11 games off on average.

What went wrong?

While you can never say with 100% certainty what went wrong, I do have some ideas for why my predictions were so bad.

1. The ARIMA(1,0,1) model
    - Chose this from a big generalization of all 30 MLB teams
    - Each team could have had it's own unique model and using those might be better
2. New Players and Players that left
    - Players that were traded for, signed in FA, new prospects would help a team win more than they were predicted and vice versa.
3. Constant season talent
    - Teams talent goes up and down at different points of the season do to current injured players and other things. This model does not account for that.
4. Progression of Players that stay on roster
    - Older teams will more naturally go down, while younger teams will get better. Win % doesn't account for this.

World Series Winners

Just for an interesting look, I decided to see how many times each team won the World Series in the 10,000 simulations.

```
results.list %>%
  map(5) %>%
  map(2) %>%
  map(1) %>%
  unlist() %>%
  data.frame() -> ws.winners
colnames(ws.winners) = 'Winners'
ws.winners %>%
  group_by(Winners) %>%
  summarize(Titles = n()) %>%
  arrange(desc(Titles))
```

```
## # A tibble: 28 x 2
##    Winners Titles
##    <fct>    <int>
##  1 CLE       2125
##  2 NYY       1450
##  3 HOU       1245
##  4 LAD       1202
##  5 ARI       1047
##  6 WSN        901
##  7 CHC        494
##  8 BOS        361
##  9 STL        239
## 10 COL        232
## # ... with 18 more rows
```

The actual winners, The Boston Red Sox, were predicted to win 361 of the 10,000 times. There are however, 2 teams that did not win the World Series at all in the 10,000 simulations.

```
Teams.Split[!(Teams.Split$franchID %in% ws.winners$Winners),1]
```

```
## # A tibble: 2 x 1
##   franchID
```

```
##   <chr>
## 1 SDP
## 2 SFG
```