Assignment 1 - Loading Data Into a Dataframe

William Jasmine

2022-09-01

Introduction

The data we used for this assignment this assignment can be found as the following github location, and is provided via FiveThiryEight's publically available databases. The data contains predictions of all Major League Baseball games from the latest season (2022), which are made via Elo ratings. The specifics of how FiveThiryEight calculates these Elo ratings and subsequent predictions can be found in this article.

Import Packages

The following chunk loads the libraries needed to run the subsequent R code for this assignment.

```r
library("RCurl")
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("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v purrr   0.3.4
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::complete() masks RCurl::complete()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
```

Import Data

The following chunk provides the code to import and take a first look at our data:

```r
csv_data <- getURL("https://projects.fivethirtyeight.com/mlb-api/mlb_elo_latest.csv")
df <- data.frame(read.csv(text=csv_data))
glimpse(df)
```

```
## Rows: 2,430
## Columns: 26
## $ date         <chr> "2022-10-05", "2022-10-05", "2022-10-05", "2022-10-05", "~
## $ season       <int> 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 202~
## $ neutral      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ playoff      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ team1        <chr> "LAD", "SEA", "SDP", "NYM", "MIL", "HOU", "FLA", "CLE", "~
## $ team2        <chr> "COL", "DET", "SFG", "WSN", "ARI", "PHI", "ATL", "KCR", "~
## $ elo1_pre     <dbl> 1619.258, 1538.452, 1511.278, 1559.803, 1519.420, 1576.95~
## $ elo2_pre     <dbl> 1458.059, 1449.094, 1500.692, 1427.806, 1486.195, 1529.62~
## $ elo_prob1    <dbl> 0.7438531, 0.6575839, 0.5496101, 0.7105367, 0.5816163, 0.~
## $ elo_prob2    <dbl> 0.2561469, 0.3424161, 0.4503899, 0.2894633, 0.4183837, 0.~
## $ elo1_post    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ elo2_post    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ rating1_pre  <dbl> 1619.903, 1525.827, 1522.537, 1563.657, 1528.464, 1568.70~
## $ rating2_pre  <dbl> 1453.174, 1450.817, 1494.614, 1432.806, 1478.508, 1535.71~
## $ pitcher1     <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "~
## $ pitcher2     <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "~
## $ pitcher1_rgs <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pitcher2_rgs <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pitcher1_adj <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ pitcher2_adj <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ rating_prob1 <dbl> 0.7505160, 0.6547150, 0.5815474, 0.7196434, 0.6075037, 0.~
## $ rating_prob2 <dbl> 0.2494840, 0.3452850, 0.4184526, 0.2803566, 0.3924963, 0.~
## $ rating1_post <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ rating2_post <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ score1       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ score2       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

The *glimpse()* function reveals that this data set has 26 features, and 2,430 observations (which in this case are all the MLB games in the 2022 regular season).

Data Dictionary

The following is a description of each of the column names listed above, which can also be found in the data set documentation.

Column Name	Description
date	Date of game
season	Year of season
neutral	Whether game was on a neutral site
playoff	Whether game was in playoffs, and the playoff round if so
team1	Abbreviation for home team
team2	Abbreviation for away team
elo1__pre	Home team's Elo rating before the game
elo2__pre	Away team's Elo rating before the game
elo__prob1	Home team's probability of winning according to Elo ratings

Column Name	Description
elo_prob2	Away team's probability of winning according to Elo ratings
elo1_post	Home team's Elo rating after the game
elo2_post	Away team's Elo rating after the game
rating1_pre	Home team's rating before the game
rating2_pre	Away team's rating before the game
pitcher1	Name of home starting pitcher
pitcher2	Name of away starting pitcher
pitcher1_rgs	Home starting pitcher's rolling game score before the game
pitcher2_rgs	Away starting pitcher's rolling game score before the game
pitcher1_adj	Home starting pitcher's adjustment to their team's rating
pitcher2_adj	Away starting pitcher's adjustment to their team's rating
rating_prob1	Home team's probability of winning according to team ratings and starting pitchers
rating_prob2	Away team's probability of winning according to team ratings and starting pitchers
rating1_post	Home team's rating after the game
rating2_post	Away team's rating after the game
score1	Home team's score
score2	Away team's score

Data Cleaning/Prepartion

Some of the columns in the above data set are redundant, since they can be easily derived from others. For example, we do not need two probability columns given that the probability of one team winning can be easily determined if we know the other team's probability of winning (if the probability of one team winning is *p*, the probability of the other team winning is *1-p*). Thus, we can subset our data frame to remove two of the redundant probability columns, **elo_prob2**, and **rating_prob2**. Additionally, there are two columns that provide no additional information, and can be removed. These fields, and explanation of why they can be removed is given below:

- **playoff**: This assignment was completed on 9/1, well before the 2022 MLB playoffs start, meaning that there is not yet any data available for playoff games.
- **season**: All games in this data set were played during the 2022 season.

The following R blocks confirm the above explanation by showing the one unique value in each of the **playoff** and **season** columns, respectively (NA and 2022).

```
print(unique(df$playoff))
```

```
## [1] NA
```

```
print(unique(df$season))
```

```
## [1] 2022
```

The following code block subsets our dataframe to remove the aforementioned redundant/useless columns. It then uses the *dim()* function to confirm that 4 of our original 26 columns have now been removed (22 remaining):

```
df <- subset(df, select = -c(rating_prob2, elo_prob2, playoff, season))
dim(df)
```

```
## [1] 2430   22
```

Additionally, there are some columns that would be useful to add to the data set in order to evaluate the effectiveness of FiveThiryEight's predictions. For games that have already been played, we can look at how both of their prediction metrics align with the results of the actual game. First, the following chunk subsets the data to include only those games that have been played, by checking for non NA values in the **score1** and **score2** columns.

```
df <- subset(df, is.na(score1)==FALSE & is.na(score2)==FALSE)
dim(df)
```

```
## [1] 1958   22
```

The *dim()* command in the code block above shows that after filtering there are 1,956 rows remaining, meaning that we have removed 2,430 - 1,956 = 474 unplayed (as of this point) games.

Next, the chunk below creates two additional columns, **elo_correct** and **rating_correct**, which checks to see if their respective **elo_prob1** and **rating_prob1** columns accurately predicted the results of the game.

```
df <- df %>%
  mutate(df, elo_correct = ifelse((elo_prob1 > 0.5 & score1 > score2) |
                                    (elo_prob1 < 0.5 & score1 < score2),
                                  TRUE, FALSE))

df <- df %>%
  mutate(df, rating_correct = ifelse((rating_prob1 > 0.5 & score1 > score2) |
                                       (rating_prob1 < 0.5 & score1 < score2),
                                     TRUE, FALSE))
dim(df)
```

```
## [1] 1958   24
```

The *dim* function above confirms that two new columns have been added to our dataframe.

Data Analysis

Now that the data has been cleaned and prepped, our final dataframe can be further analyzed to answer some interesting questions. First of all, we can check how often FiveThiryEight was able to accurately able to predict the result of MLB games. The chunk below prints the percentage of times the **elo_prob1** field accurately predicted the game results:

```
nrow(subset(df, elo_correct == TRUE)) / nrow(df) * 100
```

```
## [1] 58.78447
```

The chunk below does the same for the **rating_prob1** results

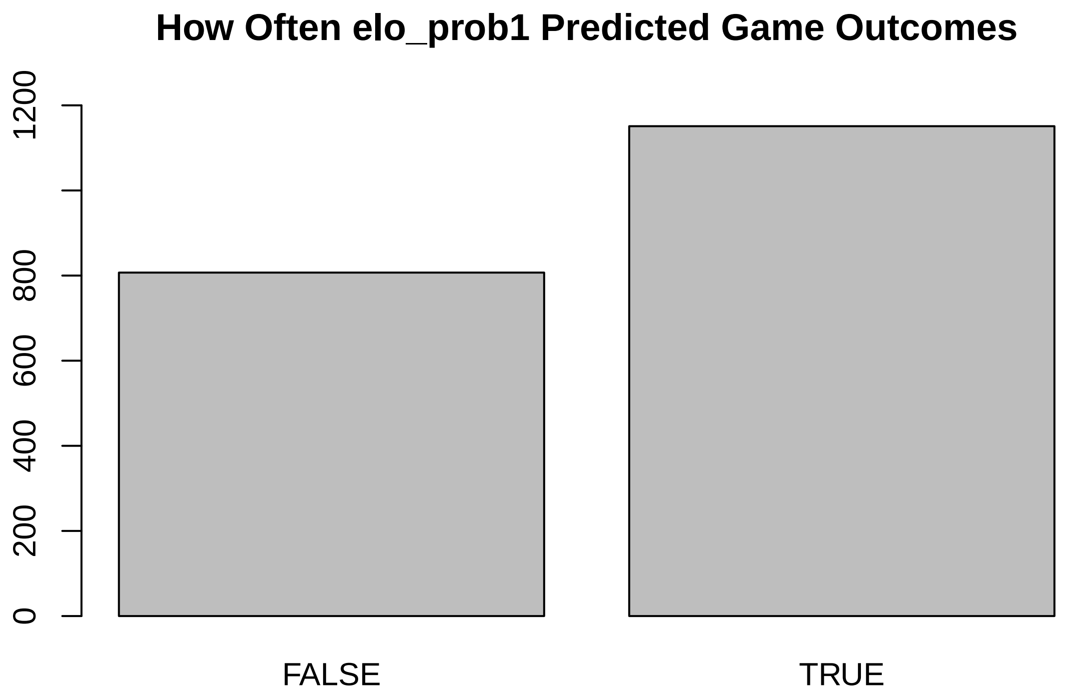
```
nrow(subset(df, rating_correct == TRUE)) / nrow(df) * 100
```

```
## [1] 59.75485
```

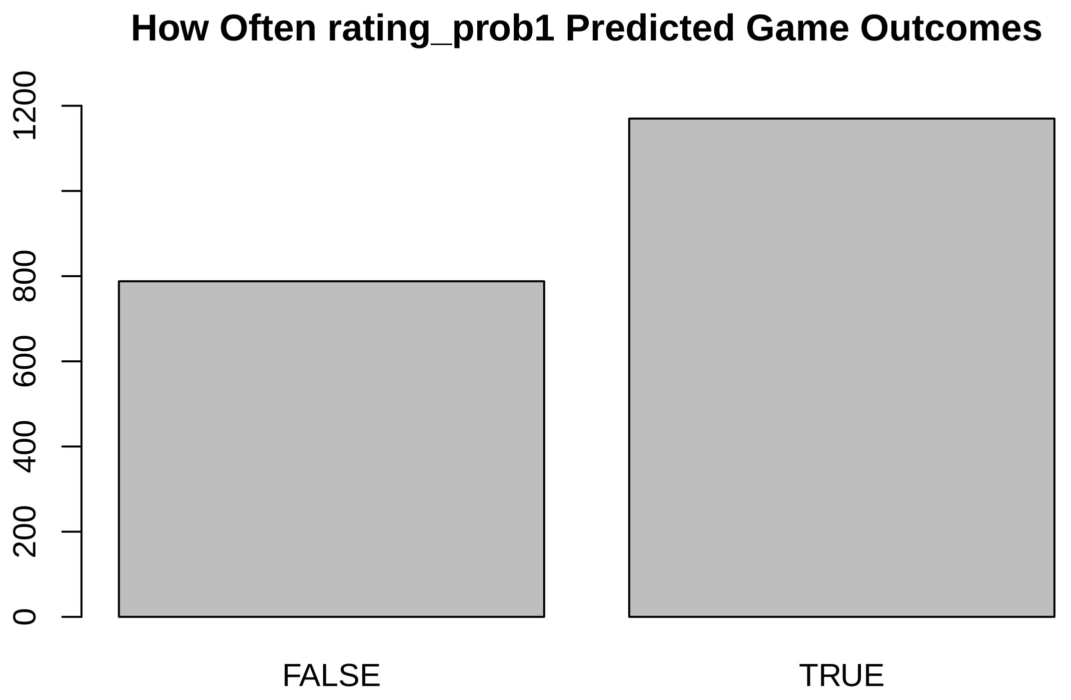
As you can see from the outputs above, both prediction methodologies do predict the correct outcome of MLB games better than chance. However, it will require additional analysis to prove whether or not this improvement is statistically significant.

The results seen above can also be visualized below in the following histograms:

```
barplot(table(df$elo_correct), ylim=c(0,1200),
        main="How Often elo_prob1 Predicted Game Outcomes")
```



```
barplot(table(df$rating_correct), ylim=c(0,1200),
        main="How Often rating_prob1 Predicted Game Outcomes")
```



Additionally, if we separate out the correct and incorrect predictions, we can see if there's a difference in how close the probabilities were to making the right guess. For this analysis, we will focus on using only the **rating_correct** column, since it was more accurate.

```r
df_correct <- subset(df, rating_correct == TRUE)
df_incorrect <- subset(df, rating_correct == FALSE)
```

Next, we can add a field to see the difference in win probability for each team. This can be written mathematically as *|p - (1 - p)| = |2p - 1|*, and is done in R in the following chunk:

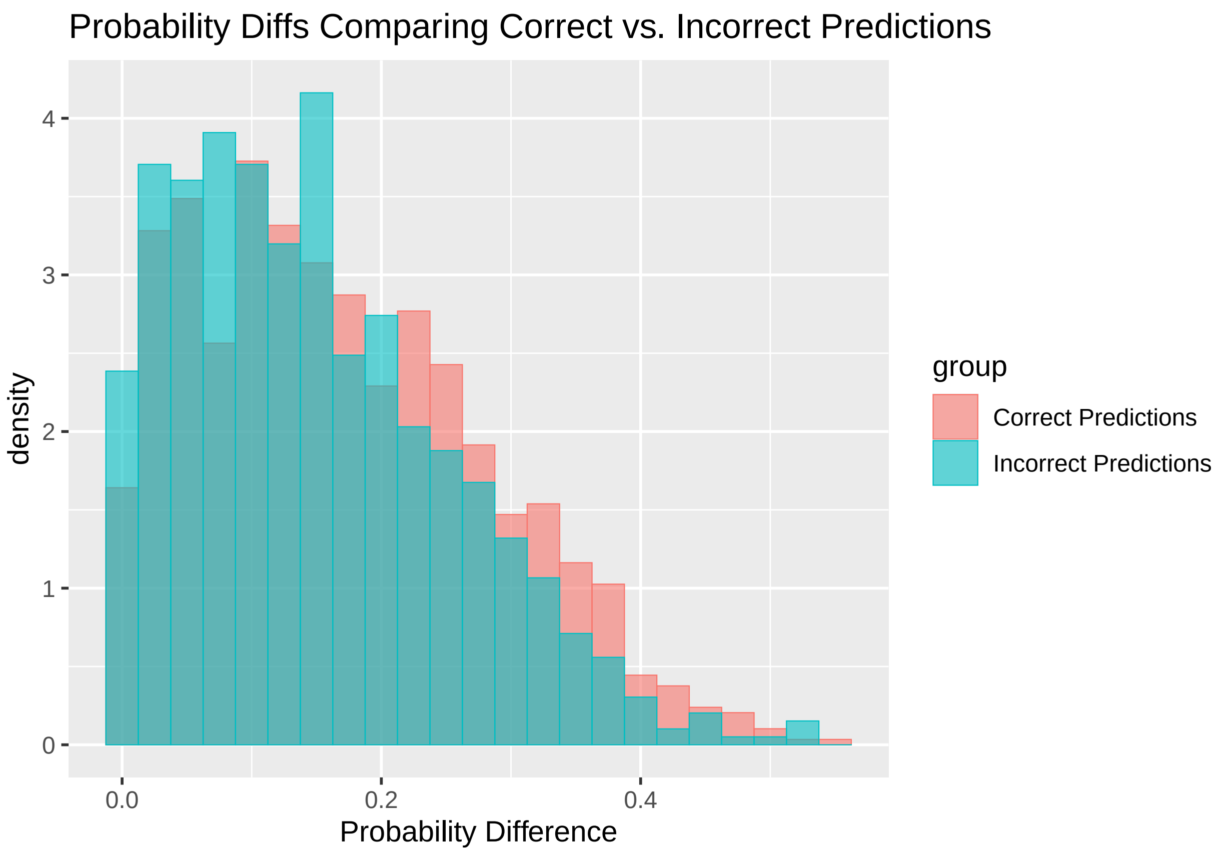
```r
df_correct <- df_correct %>%
  mutate(prob_diff = abs(2 * rating_prob1 - 1))

df_incorrect <- df_incorrect %>%
  mutate(prob_diff = abs(2 * rating_prob1 - 1))
```

Next, we can compare the differences in win probability for both the correct and incorrect predictions using a histogram (graphic inspired by this stack overflow post):

```r
dat1 = data.frame(x=df_correct$prob_diff, group="Correct Predictions")
dat2 = data.frame(x=df_incorrect$prob_diff, group="Incorrect Predictions")
dat = rbind(dat1, dat2)

ggplot(dat, aes(x, fill=group, colour=group)) +
  geom_histogram(aes(y=..density..), binwidth = 0.025,
                 alpha=0.6, position="identity", lwd=0.2) +
    ggtitle("Probability Diffs Comparing Correct vs. Incorrect Predictions") +
      xlab('Probability Difference')
```



The graphic above shows that the incorrect probability difference distribution is pushed slightly to the left of the correct probability distribution, possibly indicating that the FiveThirtyEight probability predictions are on average closer to being 50-50 when being incorrect, as opposed to when they get it right. Once again, further statistical testing would be needed to prove this.

Conclusion

The above code goes through the process of importing, cleaning, and preparing a data set, and shows a few data analyses that can be done as a result of completing these initial steps. It does seem as though FiveThirtyEight's prediction methodology does correctly predict the outcome of MLB games more likely than chance, but a next step in this case would be to verify statistically if that is indeed the case.