MTH 369 Final Project

Christopher Bussen

2023-12-15

```
library(pacman)
p_load(dplyr, tidyverse, viridis, plotly, DT, fastDummies, leaps, corrplot, car, MASS, lmtest, glmnet, nortest)

mlb_full <- read_csv("/Users/christopherbussen/Documents/School/UDF2023/MTH369/finalProject/qualified_trimmed_wit
h_salary.csv")
```

```
## Rows: 134 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): Player, Pos, Team
## dbl (9): Age, RBI, SB, AVG, K%, wRC+, Barrel%, Fielding, Salary
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# sort based on name A-Z
mlb_full <- mlb_full[order(mlb_full$Player), ]

# originally wanted to look at effect of team but it complicated things with 29 dummy variables for 30 teams
mlb_full <- mlb_full %>%
   dplyr::select(-Team)

# typeof(Team)
# typeof(Pos)
# ^ type character so dummy_cols will use first in alphabetical order for reference (1B)
# convert categorical team and pos to dummy
mlb <- dummy_cols(mlb_full, select_columns = "Pos", remove_first_dummy = TRUE, remove_selected_columns = TRUE)

# get rid of player name for dataset used for model
mlb <- mlb %>%
   dplyr::select(-Player)

# move salary column back to end
mlb <- mlb %>%
   dplyr::select(-Salary, everything(), Salary)

attach(mlb)
```

Research Question

How do certain statistics for MLB players correlate with salary? For example, how well does a player's batting average relate to salary? Additionally, which statistics have the biggest impact on a player's salary? I believe that some of the most important variables for predicting a player's salary will be the player's age, batting average, and wRC+. There are other variables that I think could have an impact, but they are not in the dataset for reasons mentioned in the next section. The primary objective of my model is inference, meaning that I want to focus on some of the relationships between the predictors and salary and which statistics are most important when front offices are determining how much money to offer a player. However, I do think it would be very interesting to expand on this model in the future to turn it into a predictive tool that can look at a player's statistics at the end of a season and predict how much money they will make next season if they are a free agent and negotiating new contracts with teams.

Dataset Transformations

This dataset originally comes from this link (https://www.kaggle.com/datasets/m000sey/major-league-baseball-hitting-data/data). It contained the statistics for every player with at least 100 plate appearances and had 105 columns (including some of the more basic hitting statistics as well as many of the more advanced statistics). In order to make this dataset more manageable, I decided to trim it to players who are considered “qualified hitters”. This simply means they average at least 3.1 plate appearances per team game (or 502 plate appearances throughout the entire season). This took the number of players in the dataset from 461 to 134. The csv file was also not properly handling accents in player names so I removed them and replaced them with regular letters. Additionally, I decided to add in columns for each player’s age and position because I believed these would be important variables when looking at salary. I got these columns from a downloadable csv file from RotoWire (https://www.rotowire.com/baseball/stats.php) and used the Python library Pandas to merge the age and position columns into the original dataset (could have used R but am more familiar with Python so it was quicker). Once merged, I realized that one dataset did not include “Jr.” for applicable names so I manually added the age and position for those few entries. I was also unable to find a downloadable dataset that contained the salaries of players for the 2023 season so I just manually added them in for each of the 134 players. I got this information from both USA Today (https://databases.usatoday.com/major-league-baseball-salaries-2023/) and Spotrac (https://www.spotrac.com/mlb/).

Lastly, as recommended, I decided to cut out many of the 105 variables in order to make the analysis less complicated. I kept in what I believed to be the 9 most important (or at least interesting - a few aren’t typically considered to be the most important but I wanted to see how they would impact the data) statistics. I originally had a few more but decided to cut out more variables after my initial research due to realizing that many of the statistics I had were sometimes based off of each other (e.g., took out exit velocity, launch angle, and hard hit % because barrel % includes all of these in some extent in a more encompassing statistic, took out HR because RBI tells more, took out OPS which is used to calculate wRC+, etc.). I also wanted to look at the potential impact of certain teams (for example, teams like the Dodgers tend to pay players more), but this would’ve involved creating 29 dummy variables for the 30 teams which would have made things far more complicated. Overall, this process of balancing what variables I wanted to keep and what made sense to keep in the end took very long as I had to be very selective and go back several times to cut more variables out to make the dataset manageable. I also got rid of WAR, which is an advanced statistic used to measure a player’s overall contributions because I realized it would likely be highly correlated with many statistics because it takes many of them into account in its calculation (it is also an accumulative statistic and I thought wRC+ which is a rate statistic would be more valuable). In the future, I would be interested to expand this model with more variables or look at how something like team winning percentage relates to a given player’s salary. I will describe each of the variables that I am using below.

Variables and Summary Statistics

```
glimpse(mlb_full)
```

```
## Rows: 134
## Columns: 11
## $ Player    <chr> "Adley Rutschman", "Adolis Garcia", "Alec Bohm", "Alex Bregm…
## $ Pos       <chr> "C", "OF", "3B", "3B", "OF", "SS", "2B", "OF", "1B", "OF", "…
## $ Age       <dbl> 25, 30, 27, 29, 27, 28, 25, 29, 25, 29, 22, 28, 26, 25, 23, …
## $ RBI       <dbl> 80, 107, 97, 98, 54, 58, 62, 45, 80, 95, 60, 67, 97, 73, 96,…
## $ SB        <dbl> 1, 9, 4, 3, 5, 15, 30, 13, 0, 5, 24, 5, 3, 5, 49, 0, 3, 4, 4…
## $ AVG       <dbl> 0.277, 0.245, 0.274, 0.262, 0.264, 0.263, 0.251, 0.262, 0.25…
## $ `K%`      <dbl> 0.147, 0.277, 0.154, 0.120, 0.154, 0.182, 0.182, 0.143, 0.21…
## $ `wRC+`    <dbl> 127, 124, 105, 125, 98, 88, 97, 87, 103, 119, 84, 112, 127, …
## $ `Barrel%` <dbl> 0.075, 0.161, 0.057, 0.054, 0.050, 0.034, 0.055, 0.029, 0.08…
## $ Fielding  <dbl> 5.1, 11.5, -2.9, -0.2, 9.5, -10.9, 12.7, -7.8, -3.5, -1.3, -…
## $ Salary    <dbl> 733900, 747760, 748000, 30500000, 6300000, 7800000, 1571429,…
```

```
head(mlb_full, 5)
```

```
## # A tibble: 5 × 11
##   Player       Pos     Age   RBI    SB   AVG  `K%` `wRC+` Barre…¹ Field…² Salary
##   <chr>        <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>   <dbl>  <dbl>
## 1 Adley Rutsc… C        25    80     1 0.277 0.147    127   0.075     5.1 7.34e5
## 2 Adolis Garc… OF       30   107     9 0.245 0.277    124   0.161    11.5 7.48e5
## 3 Alec Bohm    3B       27    97     4 0.274 0.154    105   0.057    −2.9 7.48e5
## 4 Alex Bregman 3B       29    98     3 0.262 0.12     125   0.054    −0.2 3.05e7
## 5 Alex Verdugo OF       27    54     5 0.264 0.154     98   0.05      9.5 6.3 e6
## # … with abbreviated variable names ¹`Barrel%`, ²Fielding
```

```
# get number of observations and variables
nrow(mlb_full)
```

```
## [1] 134
```

```
ncol(mlb_full)
```

```
## [1] 11
```

As you can see above, this dataset has 134 observations (players), each of which has 11 (16 if you count position dummy variables instead of just position as one) variables (the dataset above shows player name for context even though I do not include it in the dataset used with the model). I will briefly describe each of these variables below. Most of these descriptions either come the dataset description on Kaggle (linked above) or on FanGraphs (https://tht.fangraphs.com/tools/glossary/).

- **Player:** player's name
- **Pos:** player's position - converted to binary dummy variables (1 if player is that position) - 1B is reference variable so all 0's implies 1B
- **Age:** player's age
- **RBI:** runs batted in
- **SB:** stolen bases
- **AVG:** batting average (hits / at bats)
- **K%:** percent of plate appearances resulting in a strikeout
- **wRC+:** weighted runs created plus - estimate of the number of runs created by a player, compared to the league average and asjusted for ballpark (100 is average, 120 is 20% above average, 80 is 20% below average) - good all encompassing statistic for a player's offensive contribution
- **Barrel%:** percentage of batted balls classified as barrels
- **Fielding:** fielding runs above average
- **Salary:** player's annual salary in USD

```
summary(mlb_full)
```

```
##     Player              Pos                 Age             RBI        
##  Length:134         Length:134         Min.   :22.00   Min.   : 25.00  
##  Class :character   Class :character   1st Qu.:26.00   1st Qu.: 61.25  
##  Mode  :character   Mode  :character   Median :28.00   Median : 74.00  
##                                        Mean   :28.43   Mean   : 75.46  
##                                        3rd Qu.:31.00   3rd Qu.: 90.75  
##                                        Max.   :39.00   Max.   :139.00  
##        SB             AVG               K%              wRC+       
##  Min.   : 0.00   Min.   :0.1970   Min.   :0.0550   Min.   : 60.00  
##  1st Qu.: 2.25   1st Qu.:0.2480   1st Qu.:0.1715   1st Qu.: 98.25  
##  Median : 8.00   Median :0.2620   Median :0.2105   Median :111.00  
##  Mean   :11.66   Mean   :0.2617   Mean   :0.2082   Mean   :111.51  
##  3rd Qu.:16.00   3rd Qu.:0.2747   3rd Qu.:0.2390   3rd Qu.:123.75  
##  Max.   :73.00   Max.   :0.3540   Max.   :0.3270   Max.   :180.00  
##     Barrel%          Fielding            Salary        
##  Min.   :0.0050   Min.   :-15.9000   Min.   :  720000  
##  1st Qu.:0.0610   1st Qu.: -3.6500   1st Qu.:  913750  
##  Median :0.0875   Median :  0.4500   Median : 6737500  
##  Mean   :0.0896   Mean   :  0.5381   Mean   : 9600255  
##  3rd Qu.:0.1140   3rd Qu.:  4.3750   3rd Qu.:14875000  
##  Max.   :0.1930   Max.   : 15.5000   Max.   :35500000  
```

Above, we can see the summary statistics of each of the variables previously mentioned. When looking at these statistics, keep in mind that every player in this dataset must have at least 502 at bats. This means that players who are often injured or who are not good enough to be everyday players will not be taken into account. This makes sense when you look at a statistic like wRC+, where the mean is 111.51 but as mentioned above an “average” player will be at 100 (meaning that the average player in this dataset is an above-average player). Another thing worth noting from these summary statistics is that there is a very wide range of values for player salary. I will also show below what proportion of the total dataset plays each position.

```
num_players <- nrow(mlb)

c_prop <- sum(Pos_C / num_players)
second_prop <- sum(Pos_2B / num_players)
ss_prop <- sum(Pos_SS / num_players)
third_prop <- sum(Pos_3B / num_players)
of_prop <- sum(Pos_OF / num_players)
dh_prop <- sum(Pos_DH / num_players)
# first is dummy column so need to use other columns subtracted from 1 to get 1b prop
first_prop <- 1 - (c_prop+second_prop+ss_prop+third_prop+of_prop+dh_prop)

pos_props <- c(c_prop, first_prop, second_prop, ss_prop, third_prop, of_prop, dh_prop)
pos_props <- matrix(pos_props, nrow = 1, ncol = 7)
colnames(pos_props) <- c("C", "1B", "2B", "SS", "3B", "OF", "DH")

print(pos_props)
```

```
##               C        1B       2B        SS        3B        OF         DH
## [1,] 0.06716418 0.1343284 0.119403 0.1567164 0.1492537 0.3507463 0.02238806
```

Above we can see the proportion of the total observations that are categorized under each position. We can see that outfield accounts for about 35% of our data, which makes sense considering that this is classifying 3 positions (LF, CF, RF) under one category.

Data Table

```
DT::datatable(mlb_full)
```

Show 10 entries Search:

	Player	Pos	Age	RBI	SB	AVG	K%	wRC+	Barrel%	Fielding	Salary
1	Adley Rutschman	C	25	80	1	0.277	0.147	127	0.075	5.1	733900
2	Adolis Garcia	OF	30	107	9	0.245	0.277	124	0.161	11.5	747760
3	Alec Bohm	3B	27	97	4	0.274	0.154	105	0.057	-2.9	748000
4	Alex Bregman	3B	29	98	3	0.262	0.12	125	0.054	-0.2	30500000
5	Alex Verdugo	OF	27	54	5	0.264	0.154	98	0.05	9.5	6300000
6	Amed Rosario	SS	28	58	15	0.263	0.182	88	0.034	-10.9	7800000
7	Andres Gimenez	2B	25	62	30	0.251	0.182	97	0.055	12.7	1571429
8	Andrew Benintendi	OF	29	45	13	0.262	0.143	87	0.029	-7.8	8600000
9	Andrew Vaughn	1B	25	80	0	0.258	0.21	103	0.084	-3.5	760000
10	Anthony Santander	OF	29	95	5	0.257	0.232	119	0.102	-1.3	7400000

Showing 1 to 10 of 134 entries Previous 1 2 3 4 5 … 14 Next

To get a better idea of the dataset that will actually be used with the model, I will show the first few rows of the updated dataset that has the position dummy variables and player name excluded. Overall, the full model includes 14 predictors:

```
glimpse(mlb)
```

```
## Rows: 134
## Columns: 15
## $ Age       <dbl> 25, 30, 27, 29, 27, 28, 25, 29, 25, 29, 22, 28, 26, 25, 23, …
## $ RBI       <dbl> 80, 107, 97, 98, 54, 58, 62, 45, 80, 95, 60, 67, 97, 73, 96,…
## $ SB        <dbl> 1, 9, 4, 3, 5, 15, 30, 13, 0, 5, 24, 5, 3, 5, 49, 0, 3, 4, 4…
## $ AVG       <dbl> 0.277, 0.245, 0.274, 0.262, 0.264, 0.263, 0.251, 0.262, 0.25…
## $ `K%`      <dbl> 0.147, 0.277, 0.154, 0.120, 0.154, 0.182, 0.182, 0.143, 0.21…
## $ `wRC+`    <dbl> 127, 124, 105, 125, 98, 88, 97, 87, 103, 119, 84, 112, 127, …
## $ `Barrel%` <dbl> 0.075, 0.161, 0.057, 0.054, 0.050, 0.034, 0.055, 0.029, 0.08…
## $ Fielding  <dbl> 5.1, 11.5, -2.9, -0.2, 9.5, -10.9, 12.7, -7.8, -3.5, -1.3, -…
## $ Pos_2B    <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, …
## $ Pos_3B    <int> 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, …
## $ Pos_C     <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ Pos_DH    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ Pos_OF    <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, …
## $ Pos_SS    <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, …
## $ Salary    <dbl> 733900, 747760, 748000, 30500000, 6300000, 7800000, 1571429,…
```

```
head(mlb, 5)
```

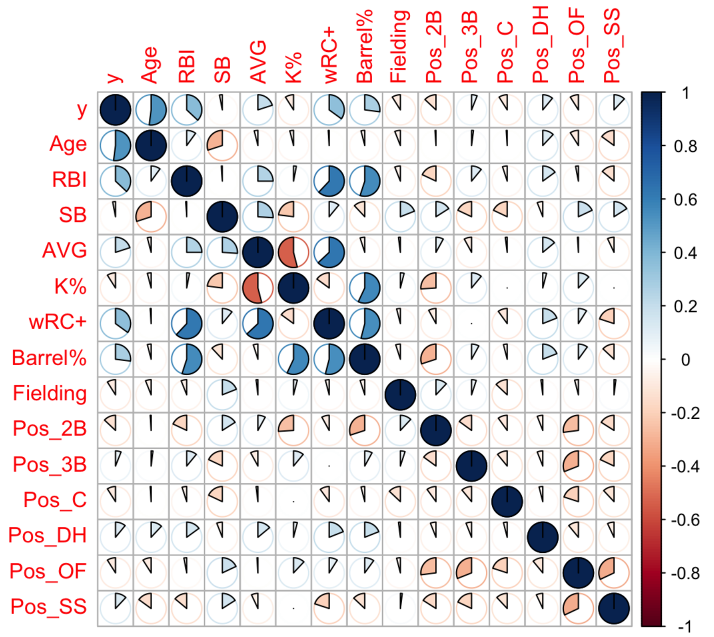
```
## # A tibble: 5 × 15
##     Age   RBI    SB   AVG  `K%` `wRC+` `Barrel%` Fielding Pos_2B Pos_3B Pos_C
##   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>     <dbl>    <dbl>  <int>  <int> <int>
## 1    25    80     1 0.277 0.147    127     0.075      5.1      0      0     1
## 2    30   107     9 0.245 0.277    124     0.161     11.5      0      0     0
## 3    27    97     4 0.274 0.154    105     0.057     -2.9      0      1     0
## 4    29    98     3 0.262 0.12     125     0.054     -0.2      0      1     0
## 5    27    54     5 0.264 0.154     98     0.05       9.5      0      0     0
## # … with 4 more variables: Pos_DH <int>, Pos_OF <int>, Pos_SS <int>,
## #   Salary <dbl>
```

Exploratory Data Analysis

In this section I will explore different relationships throughout the data with different visualizations in order to better understand the dataset prior to the creation of the model.

Correlation between Variables

```
# correlation plot between variables
x <- as.matrix(mlb[,1:14])
y <- mlb$Salary

M <- round(cor(cbind(y,x)), 2)
corrplot(M, method = "pie")
```



As we can see from the correlation plot above, there are some variables that are somewhat correlated. However, this is to be expected when looking at a dataset with MLB statistics because many of them are based on each other in some way. The following are some examples of this with likely explanations:

1. **RBI and wRC+** - RBI measures total runs batted in and wRC+ is a measure of total offensive output related to an average player and adjusted for environmental factors. Clearly, they both depend on how good a player's offensive output is.

2. **AVG and wRC+** - this is a similar case as above but AVG measures how often players get hits, which clearly still has a large impact on overall offensive impact.
3. **Barrel% and K%** - a higher barrel% (meaning higher exit velocity and optimal launch angle) typically means that a player is swinging than normal which can often lead to decreased bat control, therefore the player might miss the ball more.
4. **Barrel% and RBI/wRC+** - a higher exit velocity and more optimal launch angle typically leads to more extra base hits or home runs, therefore a good barrel% likely leads to greater offensive output from a player.

Outside of these cases, we can see that most of the predictors are not strongly correlated to each other which is ideal for the purpose of our model. Part of this result is likely due to the fact that I greatly decreased the number of predictors and dropped variables that were directly related to each other by definition (and how they are actually calculated).

```
# test for multicollinearity with full model
fit.full <- lm(Salary~., data = mlb)
(vif_full <- vif(fit.full))
```

```
##       Age       RBI        SB       AVG      `K%`    `wRC+` `Barrel%`  Fielding
##  1.207002  2.008995  1.473702  2.775300  3.100597  4.570464  4.613949  1.089890
##    Pos_2B    Pos_3B     Pos_C    Pos_DH    Pos_OF    Pos_SS
##  2.034249  1.902668  1.510781  1.186782  2.816193  2.280006
```

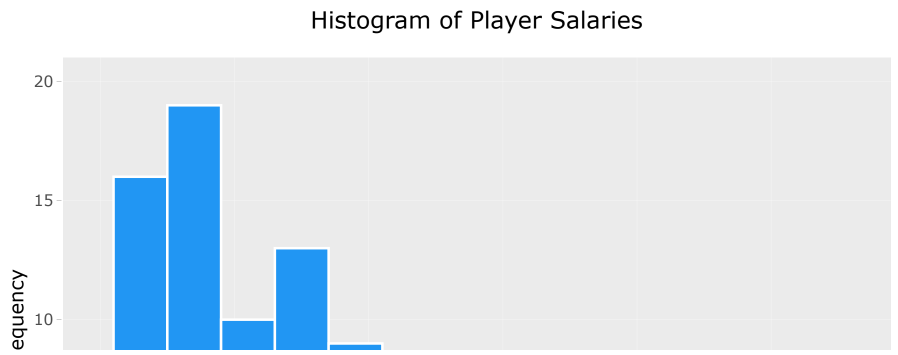
```
(avg_vif_full <- mean(vif_full))
```

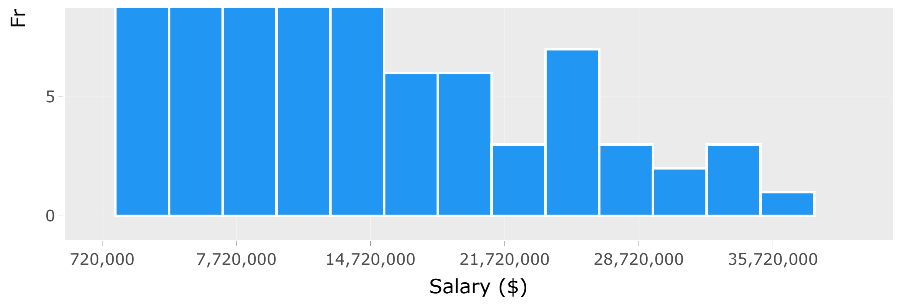
```
## [1] 2.32647
```

Here, we use the Variance Inflation Factor (VIF) to build off of the correlation plot above and determine whether or not multicollinearity is an issue. As we can see above, the highest VIF scores come from wRC+ and Barrel%. A VIF score greater than 10 typically indicates severe multicollinearity issues, and a score from 5-10 indicates significant multicollinearity that could distort regression and impact results negatively. However, as we see above, the highest scores are less than 5 and the average VIF score of all the predictors is 2.32647. This indicates that some multicollinearity is present in the model (which is to be expected), but it is not an issue that will impact our data.

General Salary Information

```
salary_hist <- ggplot(mlb_full, aes(x = Salary)) +
   geom_histogram(bins = 15, fill = "#2196f3", color = "white") +
   scale_x_continuous(breaks = seq(720000, 40000000, by=7000000), limits = c(720000, 40000000), labels = scales::
comma) +
   scale_y_continuous(breaks = seq(0, 20, by=5), limits = c(0, 20)) +
   labs(title = "Histogram of Player Salaries", x = "Salary ($)", y = "Frequency") +
   theme(plot.title = element_text(hjust = 0.5))

ggplotly(salary_hist)
```





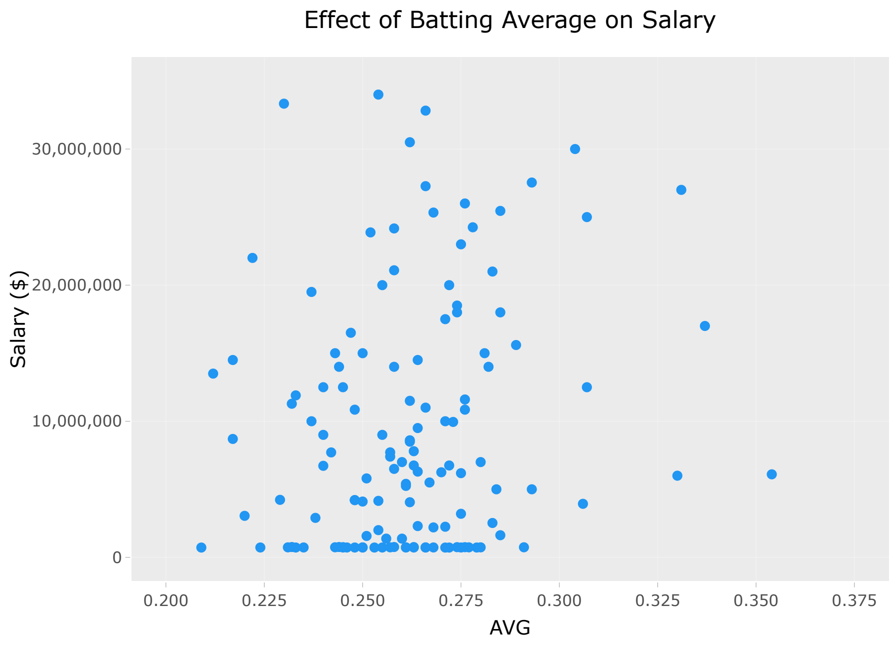
As we can see from the histogram above of player salaries, the majority of the league does not make more than $15,000,000 annually. However, some of the top players make upwards of $30,000,000. For more information, we can refer back to the summary statistics for salary shown above (Min = 720000, 1st Qu. = 913750, Median = 6737500, Mean = 9600255, 3rd Qu. = 14875000, Max. = 35500000).

AVG vs Salary

```
player_salaries <- ggplot(mlb_full, aes(x = AVG, y = Salary, label = Player)) +
  geom_point(col = "#2196f3", aes(text = paste0("Player: ", Player, "\nAVG: ", AVG, "\nAge: ", Age, "\nSalary
($): ", Salary))) +
  scale_x_continuous(breaks = seq(0.200, 0.375, by=0.025), limits = c(0.200, 0.375)) +
  scale_y_continuous(breaks = seq(0, 35000000, by=10000000), limits = c(0, 35000000), labels = scales::comma) +
  labs(title="Effect of Batting Average on Salary", x="AVG", y="Salary ($)") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning in geom_point(col = "#2196f3", aes(text = paste0("Player: ", Player, :
## Ignoring unknown aesthetics: text
```

```
ggplotly(player_salaries, tooltip = "text")
```



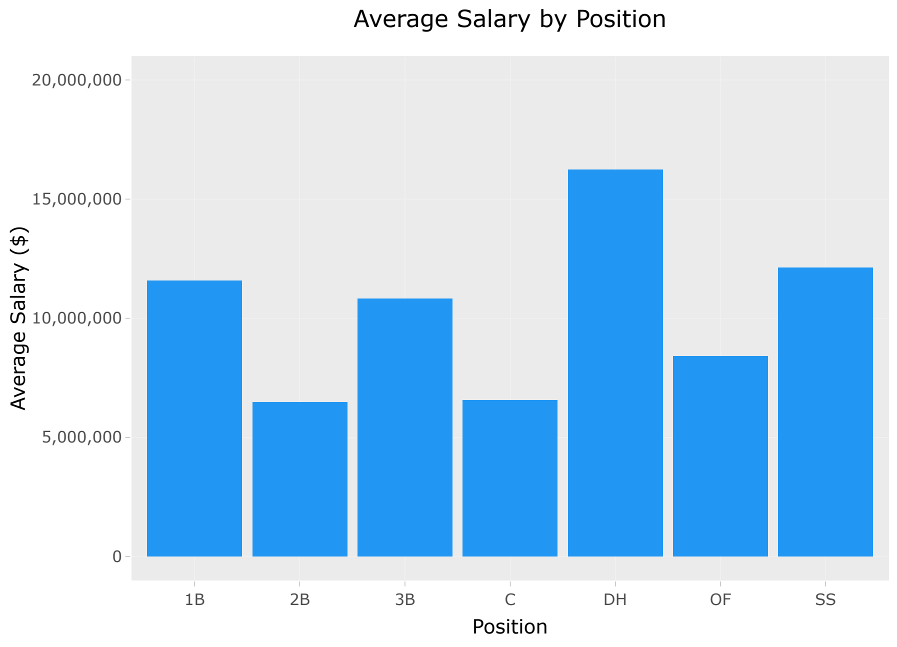
While the scatterplot above does not give us a ton of information, it at least shows us that players who make smaller salaries can have a great range in batting avergae. This is because many of the players with higher batting averages who make less money are rather young, meaning that they are still on their first Major League contract. Additionally we can see a great variance in batting average for players who have higher salaries. This is because the players with higher salaries tend to be players who have had good years in the past, but it is not uncommon in baseball for even the best players to struggle on offense throughout a season every now and then. Additionally, some of them are starting to regress due to age but are still on older contracts from many years ago.

Salary by Position

```
avgPosSalary <- mlb_full %>%
  group_by(Pos) %>%
  summarise(
    avgSalary = mean(Salary, na.rm = T)
  )

avgSalaries <- ggplot(avgPosSalary, aes(x = Pos, y = avgSalary)) +
  geom_col(fill = "#2196f3", aes(text = paste0("Position: ", Pos, "\nAverage Salary: $", round(avgSalary,0)))) +
  labs(title="Average Salary by Position", x="Position", y="Average Salary ($)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(breaks = seq(0, 20000000, by=5000000), limits = c(0, 20000000), labels = scales::comma)
```

```
## Warning in geom_col(fill = "#2196f3", aes(text = paste0("Position: ", Pos, :
## Ignoring unknown aesthetics: text
```

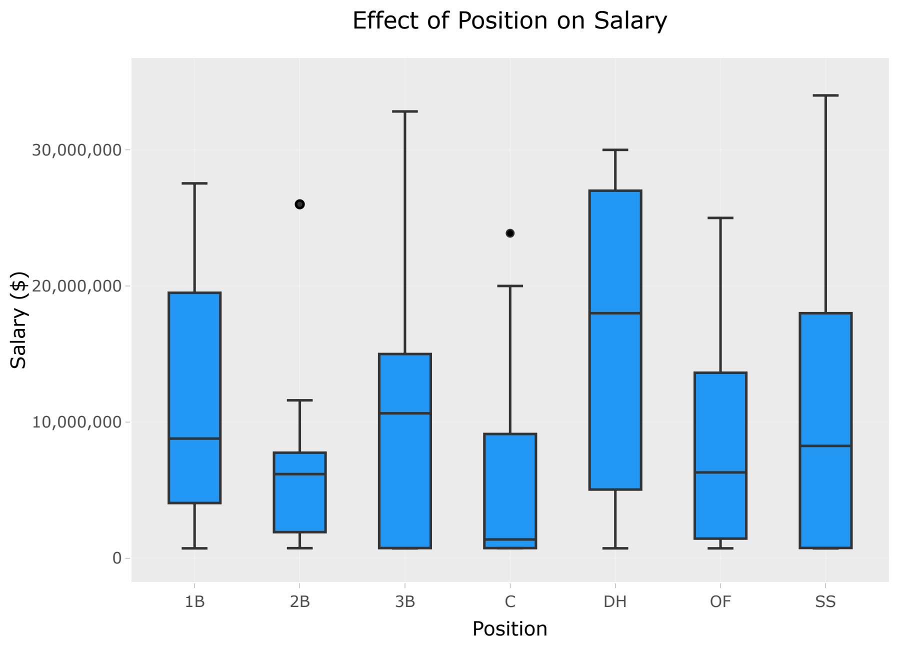
```
ggplotly(avgSalaries, tooltip = "text")
```



The plot above gives a good visualization of the average salary by position. We can see that DH, SS, and 1B tend to make the most on average, whereas 2B and C make the least.

```
pos_salary_box <- ggplot(mlb_full, aes(x = Pos, y = Salary)) +
  geom_boxplot(fill = "#2196f3")  +
  scale_y_continuous(breaks = seq(0, 35000000, by=10000000), limits = c(0, 35000000), labels = scales::comma) +
  labs(title="Effect of Position on Salary", x="Position", y="Salary ($)") +
  theme(plot.title = element_text(hjust = 0.5))

ggplotly(pos_salary_box, tooltip = "text")
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```



Above we can see a more detailed visualization of how salary can differ from position to position. We can see that DH has by far the highest median salary whereas catcher easily has the lowest. We also see that 3B and SS have the biggest ranges.

Salary vs Age

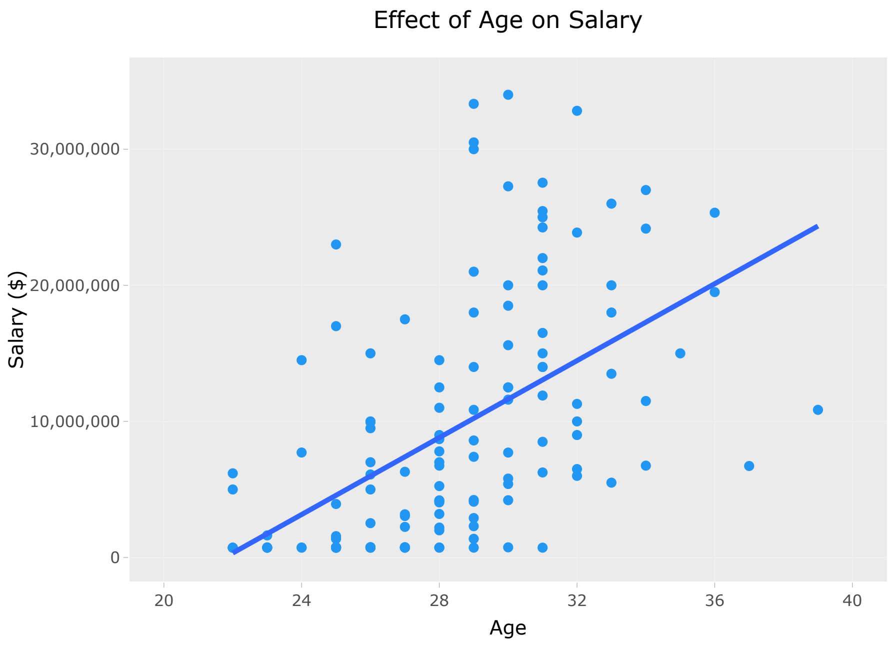
```
age_salary <- ggplot(mlb_full, aes(x = Age, y = Salary, label = Player)) +
  geom_point(col = "#2196f3", aes(text = paste0("Player: ", Player, "\nAge: ", Age, "\nSalary ($): ", Salary))) +
   geom_smooth(method = "lm", se = FALSE) +
  scale_x_continuous(breaks = seq(20, 40, by=4), limits = c(20, 40)) +
  scale_y_continuous(breaks = seq(0, 35000000, by=10000000), limits = c(0, 35000000), labels = scales::comma) +
  labs(title="Effect of Age on Salary", x="Age", y="Salary ($)") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning in geom_point(col = "#2196f3", aes(text = paste0("Player: ", Player, :
## Ignoring unknown aesthetics: text
```

```
ggplotly(age_salary, tooltip = "text")
```

```
## `geom_smooth()` using formula = 'y ~ x'
## Warning: Removed 1 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: The following aesthetics were dropped during statistical transformation: label
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



As we can see above, there is generally a positive correlation between age and an MLB player's salary. This is because players generally need some time to develop and adjust to the league which happens over their first few years. So, players are more likely to hit their baseball peaks closer to their 30s than early 20s. Additionally, when many players first make it to the MLB, they are often still on a more team-friendly (cheaper contract for a few years). However, after their first few years in the league most players are able to either enter free agency or sign an extension worth more money annually.

Barrel% vs wRC+

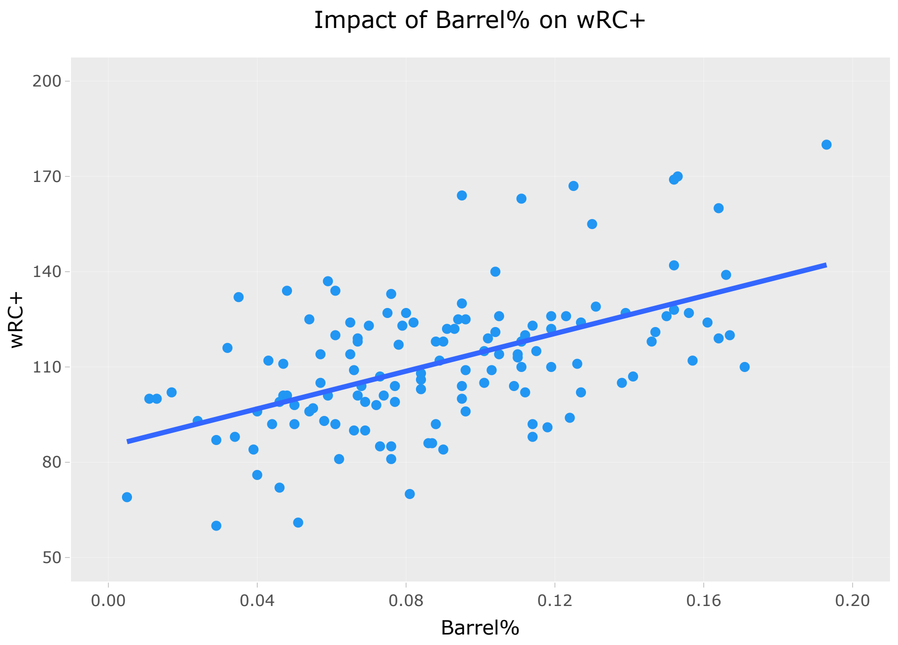
```
barrel_wrc <- ggplot(mlb_full, aes(x = `Barrel%`, y = `wRC+`, label = Player)) +
  geom_point(col = "#2196f3", aes(text = paste0("Player: ", Player, "\nBarrel%: ", `Barrel%`, "\nwRC+: ", `wRC+`,
"\nSalary ($): ", Salary))) +
   geom_smooth(method = "lm", se = FALSE) +
  scale_x_continuous(breaks = seq(0, 0.2, by=0.04), limits = c(0, 0.2)) +
  scale_y_continuous(breaks = seq(50, 200, by=30), limits = c(50, 200)) +
  labs(title="Impact of Barrel% on wRC+", x="Barrel%", y="wRC+") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning in geom_point(col = "#2196f3", aes(text = paste0("Player: ", Player, :
## Ignoring unknown aesthetics: text
```

```
ggplotly(barrel_wrc, tooltip = "text")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: The following aesthetics were dropped during statistical transformation: label
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



While this plot above does not have to do with salary, I thought it would be a helpful visualization to include in my exploratory data analysis for people who are not particularly familiar with either Barrel% or wRC+. As you can see above, there is a generally positive correlation between barrel% and wRC+. This makes sense because a higher barrel% means a player is hitting the ball harder and making better overall contact. While a player can obviously get unlucky with good contact, good contact will typically result in better hits, and therefore better offensive production (which wRC+ measures).

Model

Approach

For starters, I knew that this model would not be a poisson regression or a logistic regression model because the response variable (salary) was neither a count or a binary variable, but rather a continuous numeric variable. This leaves multiple linear regression and polynomial regression. Based on the statistics I am using, my first thought is that this dataset would work better with a multiple linear regression model with a reduced number of predictors. I also do not think polynomial regression would work great because of how complex it would quickly get with the large number of predictors that I have. I also only have 134 observations which might not be enough to create accurate estimates with an increased number of terms and I think it would likely lead to overfitting. I will start by using the subset algorithms we discussed in class (forward, backward, and stepwise selection) to look at potential reduced models. I will then look to see if any transformations are needed. I will also look at models using shrinkage/regularization methods we discussed in class, such as LASSO, Ridge, and Elastic-net regression to see which model is best.

Creating/Analyzing the Model(s)

In this section, I create several different models and check how well they perform with the given dataset based on error and values for adjusted R^2. I start with a full model (all predictors), then I attempt to reduce it using stepwise selection. I also perform transformations on the response variable in both the full and reduced models (log, square root, Box-Cox) and check model performance after each one. I also attempt to use polynomial regression with the dataset. After this I try both LASSO and Ridge Regression, finally followed by Elastic-net Regression. I will display the code I used but hide the results from everything but the best model in order to shorten the report.

```r
model <- regsubsets(Salary ~ ., data = mlb, nvmax = ncol(mlb)-1)
summary(model)

null <- lm(Salary ~ 1, data = mlb)
full <- lm(Salary ~ ., data = mlb)
# step(null, scope = list(lower=null, upper=full), direction = "forward")
# step(full, direction = "backward")

# both provides best results
step(null, scope = list(lower=null, upper=full), direction = "both")
```

```r
# creation of several different models for full and reduced

# looking for higher R^2 and lower error
summary(fit.full)
(sse <- sum(fit.full$residuals ^ 2))

# slightly improves R^2
fit.full_log <- lm(log(Salary) ~ ., data = mlb)
summary(fit.full_log)


# improves R^2
fit.full_sqrt <- lm(sqrt(Salary) ~ ., data = mlb)
summary(fit.full_sqrt)

# improves R^2 and RSE slightly over full, worse R^2 than full with sqrt(y)
fit.reduced <- lm(Salary ~ Age + `wRC+` + Pos_SS + `Barrel%` + `K%` +
    Pos_3B, data = mlb)
summary(fit.reduced)
(sse <- sum(fit.reduced$residuals^2))

# improves R^2 and RSE over full with sqrt(y)
fit.reduced_sqrt <- lm(sqrt(Salary) ~ Age + `wRC+` + Pos_SS + `Barrel%` + `K%` +
    Pos_3B, data = mlb)
summary(fit.reduced_sqrt)

# try removing statistically insignificant variables according to summary - worse
fit.reduced2 <- lm(Salary ~ Age + Pos_SS + `Barrel%` + `K%`, data = mlb)
summary(fit.reduced2)

# worse than full log(y) model in both RSE and R^2
fit.reduced_log <- lm(log(Salary) ~ Age + `wRC+` + Pos_SS + `Barrel%` + `K%` +
    Pos_3B, data = mlb)
summary(fit.reduced_log)

# try box cox
(bc <- boxcox(fit.full))
```

```r
# which.max() gives index of max y value
(lambda <- bc$x[which.max(bc$y)])

fit.full.bc <- lm(Salary^lambda~., data = mlb)
summary(fit.full.bc)

(bc <- boxcox(fit.reduced))
```

```r
# which.max() gives index of max y value
(lambda <- bc$x[which.max(bc$y)])

fit.reduced.bc <- lm(Salary^lambda ~ Age + `wRC+` + Pos_SS + `Barrel%` + `K%` +
    Pos_3B, data = mlb)
summary(fit.reduced.bc)
```

```r
centered_mlb <- as.data.frame(scale(mlb, center = TRUE, scale = FALSE))
fit.poly <- lm(Salary ~ Age + I(Age^2) + `wRC+` + I(`wRC+`^2) + `Barrel%` + I(`Barrel%`^2) + `K%` + I(`K%`^2) + A
ge:`wRC+` + Age:`Barrel%` + Age:`K%` + `wRC+`:`Barrel%`+`wRC+`:`K%` + `Barrel%`:`K%`, data = centered_mlb)

summary(fit.poly)
```

```r
# lasso/ridge
x <- as.matrix(subset(mlb, select = -c(Salary)))
y <- Salary

#10-fold cross-validation to get the lambda (tuning param)
# alpha=0 gives ridge solution
# alpha=1 gives the lasso solution
cv.lasso <- cv.glmnet(x, y, alpha =  1)
cv.lasso$lambda.min
plot(cv.lasso)
```

```r
model <- glmnet(x, y, alpha = 1, lambda = cv.lasso$lambda.min)
(coef <- coef(model))

# to compute SSE from lasso regression
y_predicted <- predict(model, s=cv.lasso$lambda.min, newx=x)
(SSE <- sum((y_predicted-y)^2))

# computing r^2
SSTO <- sum((y-mean(y))^2)
(R2 <- 1 - SSE/SSTO)

# ridge has higher R^2 and lower SSE
cv.ridge <- cv.glmnet(x, y, alpha =  0)
cv.ridge$lambda.min
plot(cv.ridge)
```

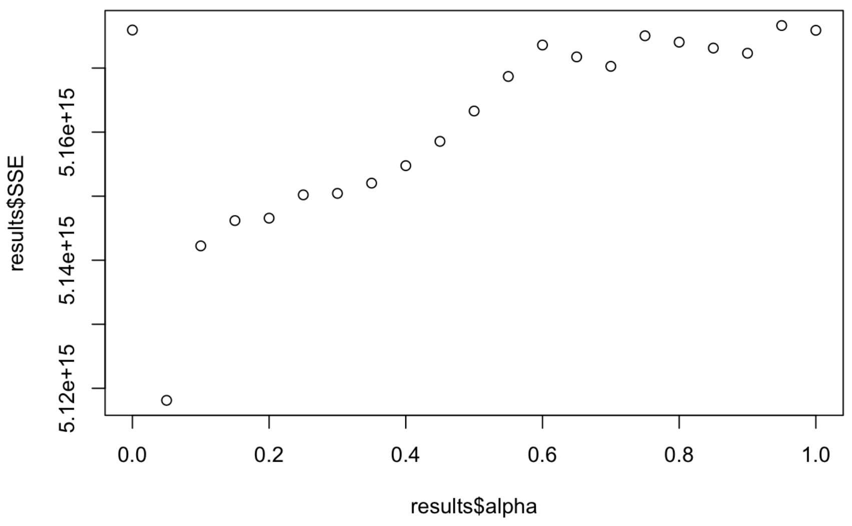
```
model <- glmnet(x, y, alpha = 0, lambda = cv.ridge$lambda.min)
(coef <- coef(model))

# to compute SSE from ridge regression
y_predicted <- predict(model, s=cv.ridge$lambda.min, newx=x)
(SSE <- sum((y_predicted-y)^2))

# computing r^2
SSTO <- sum((y-mean(y))^2)
(R2 <- 1 - SSE/SSTO)
```

Best Model - Elastic-net Regression

```
# ELASTIC NET - has lowest RSE and highest R^2
models <- list()
for (i in 0:20){
   name<- paste0("alpha", i/20)
   set.seed(2000)
   models[[name]] <- cv.glmnet(x, y,alpha=i/20)
}

results <- data.frame()
for (i in 0:20){
   name <- paste0("alpha", i/20)

   # use each model to predict y given the testing data set
   y_predicted <- predict(models[[name]], s=models[[name]]$lambda.min, newx=x)

   # calculate the SSE for testing data
   SSE <- sum((y_predicted-y)^2)

   # store the results
   temp <- data.frame(alpha=i/20, SSE=SSE, name=name)
   results <- rbind(results, temp)
}

print(results)
```

```
##    alpha          SSE      name
## 1   0.00 5.175937e+15    alpha0
## 2   0.05 5.118119e+15 alpha0.05
## 3   0.10 5.142235e+15  alpha0.1
## 4   0.15 5.146186e+15 alpha0.15
## 5   0.20 5.146567e+15  alpha0.2
## 6   0.25 5.150203e+15 alpha0.25
## 7   0.30 5.150433e+15  alpha0.3
## 8   0.35 5.152033e+15 alpha0.35
## 9   0.40 5.154761e+15  alpha0.4
## 10  0.45 5.158553e+15 alpha0.45
## 11  0.50 5.163292e+15  alpha0.5
## 12  0.55 5.168688e+15 alpha0.55
## 13  0.60 5.173597e+15  alpha0.6
## 14  0.65 5.171744e+15 alpha0.65
## 15  0.70 5.170270e+15  alpha0.7
## 16  0.75 5.175040e+15 alpha0.75
## 17  0.80 5.174042e+15  alpha0.8
## 18  0.85 5.173121e+15 alpha0.85
## 19  0.90 5.172315e+15  alpha0.9
## 20  0.95 5.176640e+15 alpha0.95
## 21  1.00 5.175894e+15    alpha1
```

```
# consider one after 0 (which is ridge) - minimize SSE
plot(results$alpha, results$SSE)
```



```
y_predicted <- predict(models[["alpha0.05"]], s = models[["alpha0.05"]]$lambda.min, newx = x)

(SSE <- sum((y_predicted-y)^2))
```

```
## [1] 5.118119e+15
```

```
# computing r^2
SSTO <- sum((y-mean(y))^2)
(R2 <- 1 - SSE/SSTO)
```

```
## [1] 0.5475004
```

```
elastic_residuals <- y - y_predicted

# RSE = sqrt(SSE / n-p-1) - n=134, p = 14
(rse <- sqrt(SSE / 134 - 15))
```

```
## [1] 6180204
```

```
(coefs <- predict(models[["alpha0.05"]], s = models[["alpha0.05"]]$lambda.min, type="coef"))
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -50385558.36
## Age           1641150.05
## RBI             30644.34
## SB              50973.21
## AVG          14591262.93
## K%          -47287503.29
## wRC+            41967.46
## Barrel%      96420148.97
## Fielding       -59488.91
## Pos_2B        1666018.12
## Pos_3B        4323029.64
## Pos_C         1479575.51
## Pos_DH        -282090.24
## Pos_OF        2269918.75
## Pos_SS        9140451.73
```

In the end, I decided on the Elastic-net Regression model because it gave the highest adjusted R^2 value of any of the other models with 0.5475004, meaning that right around 55% of the variation in player salary can be explained by the fitted regression line (or by the predictors). This model also had the lowest residual standard error (when considering scale - obviously some of the models where salary was transformed were lower because of the transformation), meaning that the Elastic-net model minimized the average size of the residuals (if you want to look at minimum MSE just square RSE - Elastic-net will still have the smallest MSE too since MSE is directly derived from RSE). In other words, it was the most accurate because it had the smallest error terms on average. While a residual standard error of 6,180,204 might seem large, it is not that notable in the grand scheme of things when you consider how the majority of players' salaries are also in the millions. So overall, I decided on the Elastic-net Regression model because the regression line fit the data best and was the most accurate by minimizing its errors compared to the other models.

As we can see from the coefficients resulting from the Elastic-net process above, this model did not eliminate any predictors as most of the other models did. The final model is shown below:

y=-50385558.36+1641150.05x1+30644.34x2+50973.21x3+14591262.93x4-47287503.29x5+41967.46x6+96420148.97x7-59488.91x8+1666018.12x9+4323029.64x10+1479575.51x11-282090.24x12+2269918.75x13+9140451.73x14

or, to be more specific:

Salary=-50385558.36+1641150.05(Age)+30644.34(RBI)+50973.21(SB)+14591262.93(AVG)-47287503.29(K%)+41967.46(wRC+)+96420148.97(Barrel%)-59488.91(Fielding)+1666018.12(Po

Based on the coefficients, we can see that some of the more important variables within the model for predicting salary are a player's batting average, strikeout percentage, and barrel percentage. This makes sense because these statistics measure how often a player gets a hit (which has always been one of the more widely looked at offensive statistics) as well as how often they make good contact. Clearly, it would make sense that the coefficient for K% would be a large negative number because a player who strikes out more will typically be paid less since teams want to limit strikeouts which are inefficient at bats.

Model Assumptions

This section will discuss the model assumptions, such as the regression function being linear, the error terms having a constant variance, error terms being independent of each other, and the error terms being normally distributed. While I show the code for several of the models, I will only display the output for the Elastic-net Regression model.

```
# assumptions for other models (not elastic net)
plot(fit.reduced_sqrt$fitted.values, fit.reduced_sqrt$residuals)
abline(h=0)
```

```
# funnel out variance - constant error variance assumption violated

dwtest(formula = fit.reduced_sqrt, alternative = "two.sided")

qqnorm(fit.reduced_sqrt$residuals)
qqline(fit.reduced_sqrt$residuals)
```

```
ad.test(fit.reduced_sqrt$residuals)

plot(fit.reduced_log$fitted.values, fit.reduced_log$residuals)
abline(h=0)
```

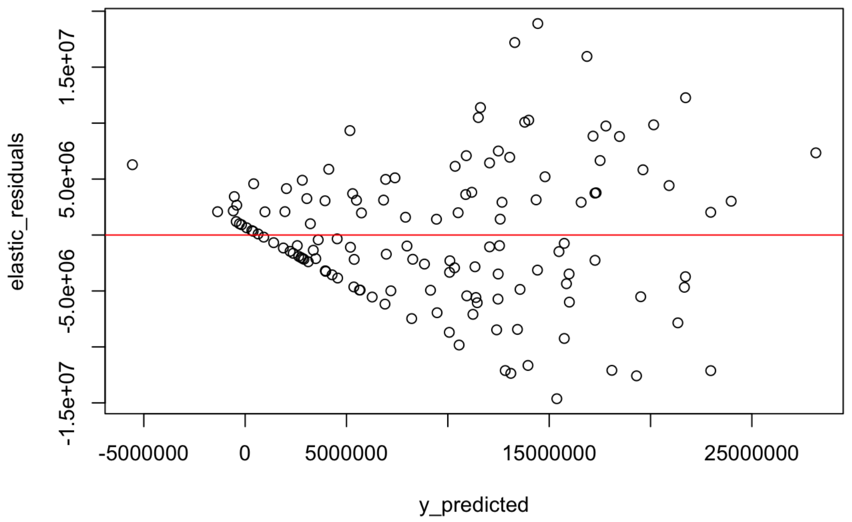
```
# funnel out variance - constant error variance assumption violated

dwtest(formula = fit.reduced_log, alternative = "two.sided")

qqnorm(fit.reduced_log$residuals)
qqline(fit.reduced_log$residuals)
```

```
ad.test(fit.reduced_log$residuals)

plot(fit.full$fitted.values, fit.full$residuals)
abline(h=0)
```

```
# funnel out variance - constant error variance assumption violated

dwtest(formula = fit.full, alternative = "two.sided")

qqnorm(fit.full$residuals)
qqline(fit.full$residuals)
```

```
ad.test(fit.full$residuals)

# ASSUMPTION OF CONSTANT VARIANCE OF ERROR TERMS ALWAYS VIOLATED
```

```
# check assumptions for elastic net
plot(y_predicted, elastic_residuals)
abline(h=0, col = "red")
```



```
# funnel out variance - constant error variance assumption violated
```

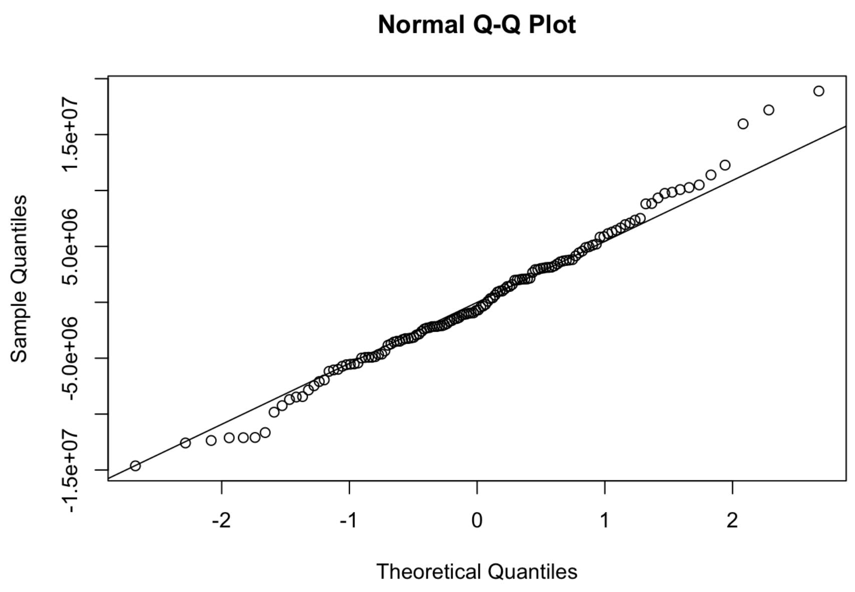
As we can see from the first plot, the assumptions of linearity of the regression model/constant variance of the error terms appears to be violated as the plot of fitted values versus residuals seems to fan out rather than form a horizontal band around 0. However, it is worth noting that this is the case for all of the models that were created. One possible explanation for this is that there are tons of factors that go into determining a player's salary that cannot necessarily be contained within a dataset, which makes it much harder for a model to make accurate predictions. Additionally, while there are technically no outliers in salary in this dataset, the extremely large variation in salary could be a reason for this as well.

```
dwtest(formula = elastic_residuals ~ y_predicted, alternative = "two.sided")
```

```
## 
##  Durbin-Watson test
## 
## data:  elastic_residuals ~ y_predicted
## DW = 2.194, p-value = 0.26
## alternative hypothesis: true autocorrelation is not 0
```

As we can see from the Durbin-Watson test above, we have a p-value of 0.26, which is clearly larger than a 0.05 signinficance level. Due to this, we fail to reject the null hypothesis that the error terms are not correlated. Therefore, we can say that the error terms for this model are independent of each other.

```
qqnorm(elastic_residuals)
qqline(elastic_residuals)
```



```
ad.test(elastic_residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  elastic_residuals
## A = 0.36081, p-value = 0.4419
```

Lastly, we have the assumption that the error terms follow a normal distribution. As we can see from the qqplot as well as the Anderson-Darling test above, due to the high p-value of 0.4419, we fail to reject the null hypothesis that the errors follow the normal distirbution. As a result, we can confidently say that the errors in the Elastic-net model are normally distributed.

Lastly, as we have already looked at, there is no severe multicollinearity within this model that could have a notable impact on the model's accuracy. This is a direct result of the transformations I performed on the dataset prior to the start of the creation of the models.

Modified Results

In this section, I wanted to attempt to make the assumption of constant variance in error terms true. I did this by shrinking the range of salaries within the dataset so that the variation in salary was less drastic. I filtered the dataset so that only players with a salary between 1,000,000 and 25,000,000 were included. However, it is worth noting that this dropped the number of observations from 134 to 88, making the models much less reliable.

```
new_mlb_filtered <- mlb %>%
   filter(Salary >= 1000000 & Salary <= 25000000)

# forward selection
null <- lm(Salary ~ 1, data = new_mlb_filtered)
full <- lm(Salary ~ ., data = new_mlb_filtered)
# step(null, scope = list(lower=null, upper=full), direction = "forward")
# step(full, direction = "backward")

# both provides best results
step(null, scope = list(lower=null, upper=full), direction = "forward")

fit.trimmed_reduced <- lm(formula = Salary ~ `Barrel%` + Age + Fielding + AVG, data = new_mlb_filtered)
summary(fit.trimmed_reduced)

# ridge regression
x <- as.matrix(subset(new_mlb_filtered, select = -c(Salary)))
y <- new_mlb_filtered$Salary

# ridge has higher R^2 and lower SSE
cv.ridge <- cv.glmnet(x, y, alpha =  0)
cv.ridge$lambda.min
plot(cv.ridge)
```

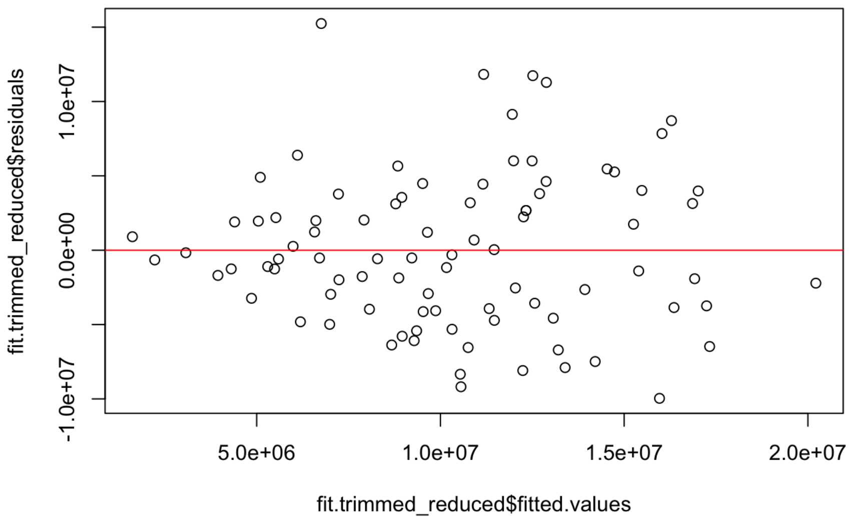
```
model <- glmnet(x, y, alpha = 0, lambda = cv.ridge$lambda.min)
(coef <- coef(model))

# to compute SSE from ridge regression
y_predicted <- predict(model, s=cv.ridge$lambda.min, newx=x)
(SSE <- sum((y_predicted-y)^2))

# computing r^2
SSTO <- sum((y-mean(y))^2)
(R2 <- 1 - SSE/SSTO)

# ELASTIC NET
models <- list()
for (i in 0:20){
   name<- paste0("alpha", i/20)
   set.seed(2000)
   models[[name]] <- cv.glmnet(x, y,alpha=i/20)
}

results <- data.frame()
for (i in 0:20){
   name <- paste0("alpha", i/20)

   # use each model to predict y given the testing data set
   y_predicted <- predict(models[[name]], s=models[[name]]$lambda.min, newx=x)

   # calculate the SSE for testing data
   SSE <- sum((y_predicted-y)^2)

   # store the results
   temp <- data.frame(alpha=i/20, SSE=SSE, name=name)
   results <- rbind(results, temp)
}

print(results)

# consider one after 0 (which is ridge) - minimize SSE
plot(results$alpha, results$SSE)
```

```
y_predicted <- predict(models[["alpha0.05"]], s = models[["alpha0.05"]]$lambda.min, newx = x)

(SSE <- sum((y_predicted-y)^2))

# computing r^2
SSTO <- sum((y-mean(y))^2)
(R2 <- 1 - SSE/SSTO)

elastic_residuals <- y - y_predicted

# RSE = sqrt(SSE / n-p-1) - n=134, p = 14
(rse <- sqrt(SSE / 134 - 15))

(coefs <- predict(models[["alpha0.05"]], s = models[["alpha0.05"]]$lambda.min, type="coef"))
```

```
fit.trimmed_reduced <- lm(formula = Salary ~ `Barrel%` + Age + Fielding + AVG, data = new_mlb_filtered)
summary(fit.trimmed_reduced)
```

```
##
## Call:
## lm(formula = Salary ~ `Barrel%` + Age + Fielding + AVG, data = new_mlb_filtered)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -9961628 -3873546  -550087  3279718 15242705
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -30425078    9424654  -3.228  0.00178 **
## `Barrel%`    78298820   14901385   5.254 1.13e-06 ***
## Age            856935     188168   4.554 1.79e-05 ***
## Fielding      -127720      81892  -1.560  0.12266
## AVG          34038320   22688146   1.500  0.13734
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5266000 on 83 degrees of freedom
## Multiple R-squared:  0.3724, Adjusted R-squared:  0.3421
## F-statistic: 12.31 on 4 and 83 DF,  p-value: 6.628e-08
```

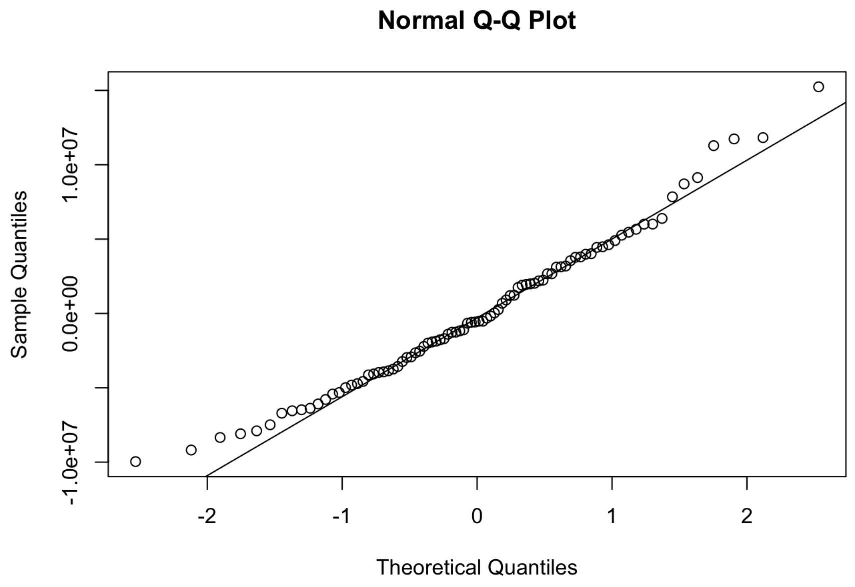
```
# check assumptions for modified/trimmed dataset
plot(fit.trimmed_reduced$fitted.values, fit.trimmed_reduced$residuals)
abline(h=0, col = "red")
```



```
# funnel out variance - constant error variance assumption violated

dwtest(formula = fit.trimmed_reduced, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  fit.trimmed_reduced
## DW = 1.9088, p-value = 0.6558
## alternative hypothesis: true autocorrelation is not 0
```

```
qqnorm(fit.trimmed_reduced$residuals)
qqline(fit.trimmed_reduced$residuals)
```



```
ad.test(fit.trimmed_reduced$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  fit.trimmed_reduced$residuals
## A = 0.33822, p-value = 0.4951
```

```
# worse results but assumptions fixed
```

As the plot of fitted values versus residuals above shows us, the data points no longer fan out as much and now form more of a horizontal band around 0. As a result, we can say that this modification to the dataset was able to change the assumption of the linearity of the regression model/constant variance of the error terms from false to true. We can also see that the other assumptions of the error terms being independent and the error terms following the normal distribution are still true. However, as I mentioned above, it is worth noting that these models are worse due to a drastic reduction in the number of observations in the dataset. When recreating the models with this modification, we can see that all of the adjusted R^2 values end up below 0.4 whereas the previous models were all around 0.5 or higher. Despite this, I thought it would be interesting to show how reducing the variation in salary can help prevent the model assumptions from being violated.

Interpretations and Findings

Final model:

Salary=-50385558.36+1641150.05(Age)+30644.34(RBI)+50973.21(SB)+14591262.93(AVG)-47287503.29(K%)+41967.46(wRC+)+96420148.97(Barrel%)-59488.91(Fielding)+1666018.12(Po

As mentioned above, this model shows that three of the more important variables for predicting a player's salary are batting average, strikeout percentage, and barrel percentage. On the other hand, some of the less important variables are wRC+, fielding, stolen bases, and RBIs. It surprised me a little bit that these coefficients were smaller because I assumed wRC+ would be more important since it is a measure of a player's overall contributions on offense. I had a similar thought process with RBIs but I am a little bit less surprised since there are other offensive statistics that give a better idea of what a player is doing on offense. I am not particularly surprised that fielding and stolen bases are not super important because, while it is definitely a plus for players to be able to field well and steal bases, it is not vital and in most situations it will not drastically impact how a front office values a player. This is partially because most players are pretty good at fielding and there usually are not many players stealing a large amount of bases. This lack of variation in these two statistics likely leads to them being less important. It was also interesting to look at the coefficients for each position, with shortstop being the highest. This does not surprise me as shortstops are typically considered to be one of the most important positions on a team and the players who play shortstop tend to be very athletic. I was a little surprised to see DH have a negative coefficient after seeing that it had the highest average salary among all of the positions. However, this could likely be due to one or two very large salaries that drag this average up. I was also a little bit surprised to see that the Elastic-net model did not shrink any coefficients to zero (essentially eliminating them), but it still gave the best performance based on adjusted R^2 and RSE.

As mentioned before, the Elastic-net Regression model gave an adjusted R^2 value of 0.5475. While this is not normally a great number for adjusted R^2, we must consider the context of this dataset. As I have mentioned earlier, there are almost infinite factors that go into contract negotiations and therefore into a player's salary. Many of these factors are not simple enough to be included into a dataset and therefore cannot be accounted for. The great variability in player salaries also makes it extremely difficult for any model to accurately predict a player's salary on a consistent basis. It is also important to remember that this dataset started off with 105 variables and 400+ observations. However, because I had to manually add in every player's salary, I had to trim the dataset down to 134 players with enough at-bats to be considered a "qualified hitter" in order to make this process more manageable. Additionally, I had to drastically reduce the number of variables in order to make the dataset manageable to create an understandable model from. I generally tried to eliminate variables based on whether or not a certain variable might be accounted for elsewhere in a variable that measures a similar statistic. I also used my previous knowledge of this domain to determine when to drop certain variables that I was confident would not have a large impact on any of the models I was going to create. Despite this, it is definitely possible that some of the variables I decided to drop to make the data more manageable could have significantly influenced a player's salary.

However, if you wanted to use a less complex model that does reduce the number of predictors, I would recommend using the stepwise selection procedure to get the reduced model that uses only age, wRC+, Pos_SS, Barrel%, K%, and Pos_3B. This model still gives an adjusted R^2 value above 0.5 and an RSE value that is not super high. If you wanted to further improve this model, you could use it with the square root transformation on salary to raise the adjusted R^2 value closer to 0.55 like the Elastic-net model is.