Baseball Salary Analysis

Edgar Hernandez, Ryan Sims, Jessica Tomas

2022-08-04

Contents

Introduction

We will be examining what statistics related to a baseball player's performance have a significant effect on the salary of a player. This is particularly interesting as it is sort of the flip-side of the coin to the traditional Sabermetrics employed by the Oakland Athletics in the 1990's to analyze player statistics to try to assemble a winning team. We will be instead be leveraging the dataset to instead work for the players, hopefully determining which statistic(s) a player should focus on improving in order to increase their compensation.

Before we begin our analysis, we must first load the data using the `Lahman` library:

```
library(Lahman)
library(knitr)
```

We then join the Salaries, Fielding, and Batting tables:

Methods

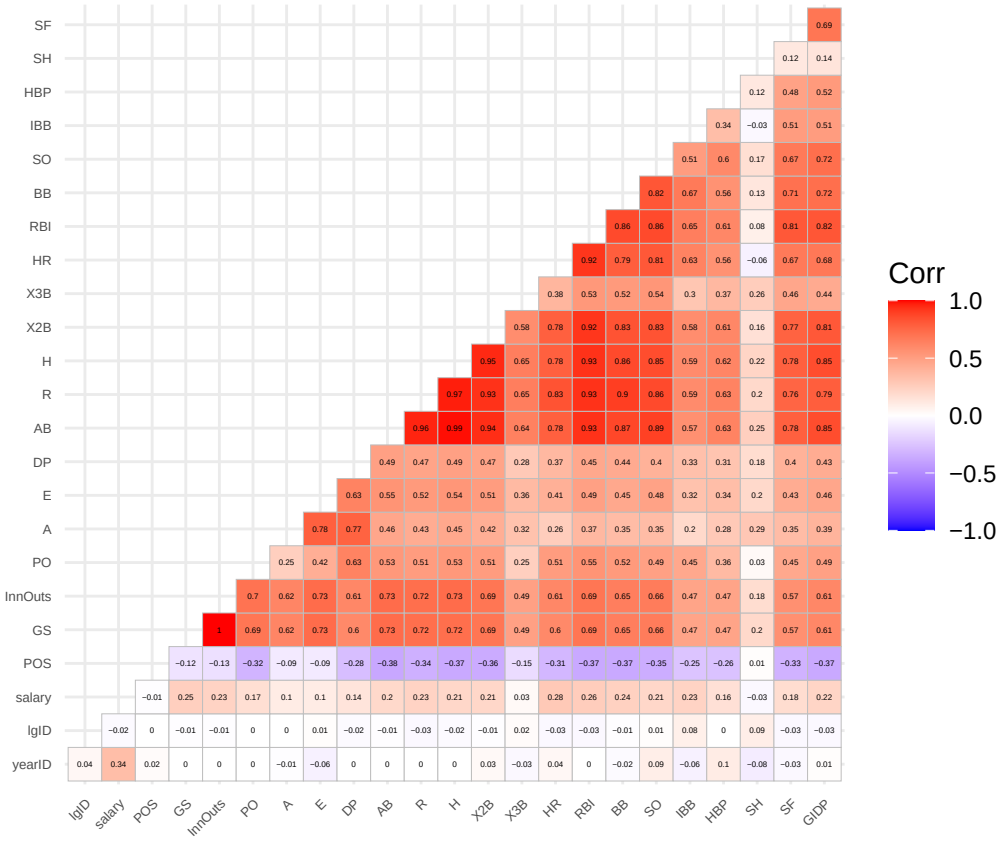
First, we visualize pairwise correlations between `Salary` and other variables:

- Calculating the correlations:

```
salary_data_plot = as.data.frame(lapply(salary_data, as.integer))
salary_data_cor = cor(salary_data_plot)
```

- Plotting the relationships:

```
library(ggcorrplot)
my_plt = ggcorrplot(salary_data_cor, lab = TRUE, lab_size = 1, type = "lower")
my_plt + theme(axis.text.x = element_text(size = 5), axis.text.y = element_text(size = 5))
```



- As expected, certain batting predictors such as `Doubles (X2B)` and `Triples (X3B)` have a very high correlation. We will keep these correlations under consideration as we refine our model.Additionally, we will remove `InnOuts` from our model since it is perfectly correlated with `Games Started (GS)`.

Before generating models, we begin by splitting the data into a test and train dataset. For this project we will split 60% of the data into a training set and 40% of the data into a testing set.

```
salary_trn_idx = sample(nrow(salary_data), size = trunc(0.60 * nrow(salary_data)))
salary_trn = salary_data[salary_trn_idx, ]
salary_tst = salary_data[-salary_trn_idx, ]
```

Next, we create a simple additive model with all of the predictors and a model with no predictors:

```
simple_add = lm(salary ~ .-InnOuts, data = salary_data)
simple_fit = lm(salary ~ 1, data = salary_data)
```

We then use AIC and BIC stepwise procedures to refine our model (See appendix):

```
biggest = formula(lm(salary ~ .-InnOuts, data = salary_data))
back_aic = step(simple_add, direction = "backward", trace = 0)

n = length(resid(simple_add))
back_bic = step(simple_add, direction = "backward", k = log(n), trace = 0)
```

- Now we can compare the LOOCV RMSE and Adjusted R Squared for both models:

```
calc_loocv_rmse = function(model) {
  sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
}
```

```
rsquared = c(summary(back_aic)$adj.r.squared, summary(back_bic)$adj.r.squared)
rmse = c(calc_loocv_rmse(back_aic), calc_loocv_rmse(back_bic))
aic_results = data.frame(Model = c("Backward AIC", "Backward BIC"),
                         AdjR = rsquared, RMSE = rmse)

kable(aic_results,
      col.names = c("Selection Procedure", "Adjusted R. Squared", "LOOCV RMSE"))
```

Selection Procedure	Adjusted R. Squared	LOOCV RMSE
Backward AIC	0.2690598	2773434
Backward BIC	0.2690598	2773434

- In addition to having identical Adjusted R^2 and LOOCV RMSE values, both models use the same coefficients:

```
all.equal(attr(back_aic$terms, "term.labels"),attr(back_bic$terms, "term.labels"))
```

```
## [1] TRUE
```

Appendix

Forward AIC additive model was also identical to backward:

```
forw_aic = step(simple_fit, direction = "forward", scope = biggest, trace = 0)
summary(forw_aic)$adj.r.squared
```

```
## [1] 0.2690598
```

```
calc_loocv_rmse(forw_aic)
```

```
## [1] 2773434
```

```
all.equal(sort(attr(back_aic$terms, "term.labels")), sort(attr(forw_aic$terms, "term.labels")))
```

```
## [1] TRUE
```