Predicting MLB Player Salaries Using Performance Metrics

Yunseo Heo

2025-04-16

Introduction

This project aims to predict Major League Baseball (MLB) player salaries using performance statistics sourced from Fangraphs. The motivation behind this study is twofold: first, to understand which player performance metrics are most influential in determining salary; and second, to estimate the salaries of players whose earnings are not publicly available or difficult to find, using statistical modeling techniques.

To achieve this, the project follows a systematic pipeline, visualized in the schematic diagram below. First, player performance data and payroll information are merged to form a unified dataset. This combined dataset is then split into two groups: players with known salaries and those with missing salary data. For players with known salaries, the data is further divided into a training set (80%) and a test set (20%). The training set is used to build a regression model, and the test set is used to evaluate its predictive accuracy. After validating the model, we apply it to impute missing salaries for the remaining players.

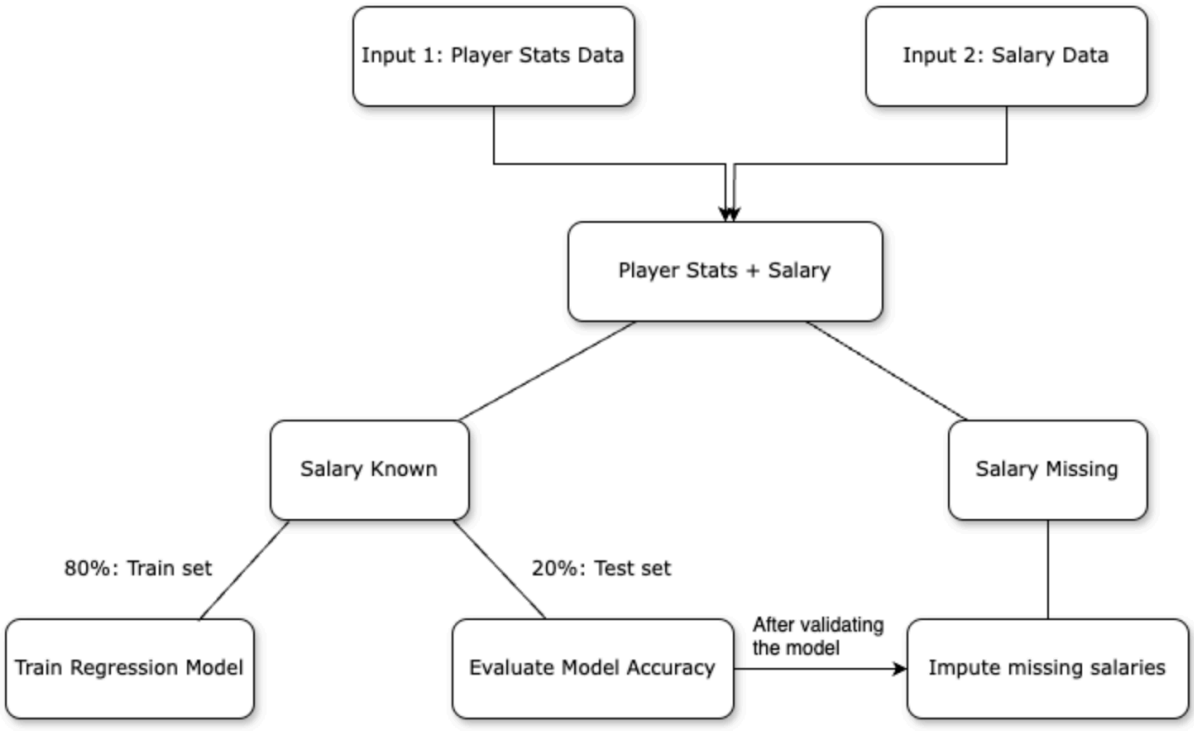


Table of Contents

1. Data Preprocessing

1.1. Load and Clean the Data

```
# Load the player data
fangraph_batter <- read.csv("fangraphs-batter.csv") # Batter
fangraph_pitcher <- read.csv("fangraphs-pitcher.csv") # Pitcher

# Load the payroll data
pay <- read.csv("mlb_salary_data.csv")

# Remove '$' and ',' & filter for 2024 & drop duplicates from the payroll data
pay <- pay %>%
  mutate(Salary = as.numeric(gsub("[$,]", "", Salary))) %>%
  filter(Year == 2024) %>%
  distinct(Name, .keep_all = TRUE)
```

1.2. Merge Datasets on Player Name

```
# Batters
batter_data <- fangraph_batter %>% left_join(pay, by = "Name")

# Pitchers
pitcher_data <- fangraph_pitcher %>% left_join(pay, by = "Name")
```

1.3. Split into Known and Missing Salary Datasets

Players with known salaries will be used to train the regression model, while players with missing salaries will be the target for salary imputation using the model.

Batters

```
# Players with known salaries (training set)
batter_train_data <- batter_data %>% filter(!is.na(Salary))

# Players with missing salaries (to be imputed)
batter_missing_data <- batter_data %>% filter(is.na(Salary))
```

Pitchers

```
# Players with known salaries (training set)
pitcher_train_data <- pitcher_data %>% filter(!is.na(Salary))

# Players with missing salaries (to be imputed)
pitcher_missing_data <- pitcher_data %>% filter(is.na(Salary))
```

2. Exploratory Data Analysis

2.1. Inspect Variables in Player Data

Before modeling, I inspected the structure of the Fangraphs player datasets to understand the available performance metrics.

Batters

```
 [1] "Name"      "Team"      "G"         "PA"        "HR"        "R"
 [7] "RBI"       "SB"        "BB."       "K."        "ISO"       "BABIP"
[13] "AVG"       "OBP"       "SLG"       "wOBA"      "xwOBA"     "wRC."
[19] "BsR"       "Off"       "Def"       "WAR"       "NameASCII" "PlayerId"
[25] "MLBAMID"
```

Pitchers

```
 [1] "Name"      "Team"      "W"         "L"         "SV"        "G"
 [7] "GS"        "IP"        "K.9"       "BB.9"      "HR.9"      "BABIP"
[13] "LOB."      "GB."       "HR.FB"     "vFA..pi."  "ERA"       "xERA"
[19] "FIP"       "xFIP"      "WAR"       "NameASCII" "PlayerId"  "MLBAMID"
```

2.2. Summary Statistics of Salaries

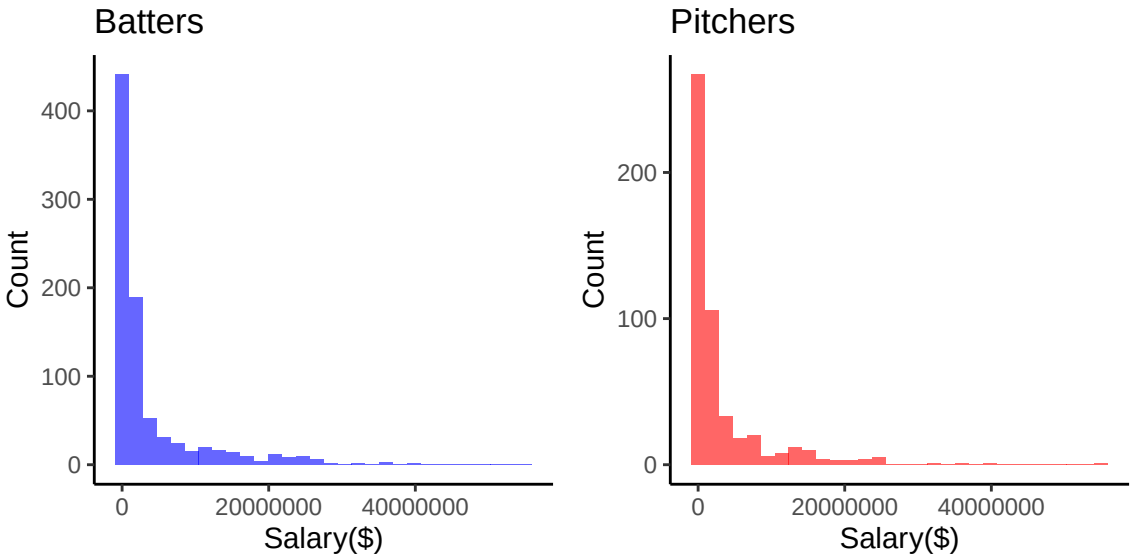
Batters

```
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   3978   500928   914420  4003831  3325000 55000000
```

Pitchers

```
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   3978   483327   795180  3379585  3000000 55000000
```

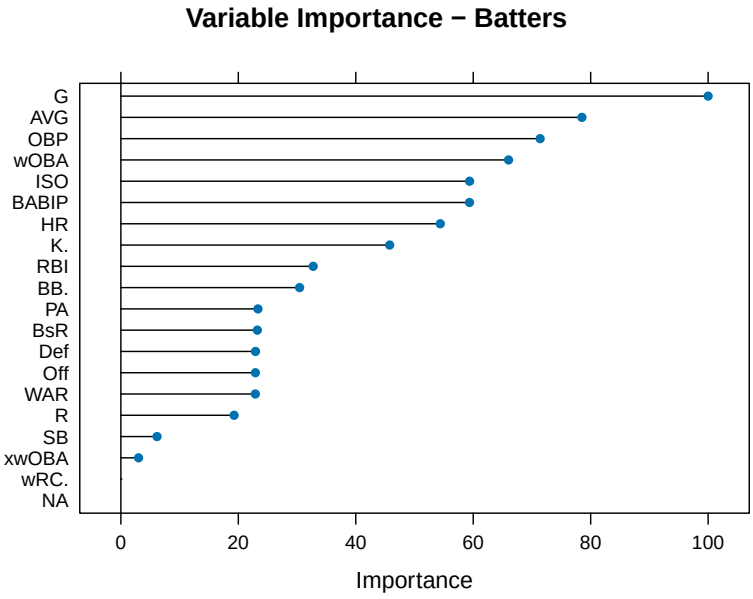
2.3. Salary Distributions



2.4. Variable Importance

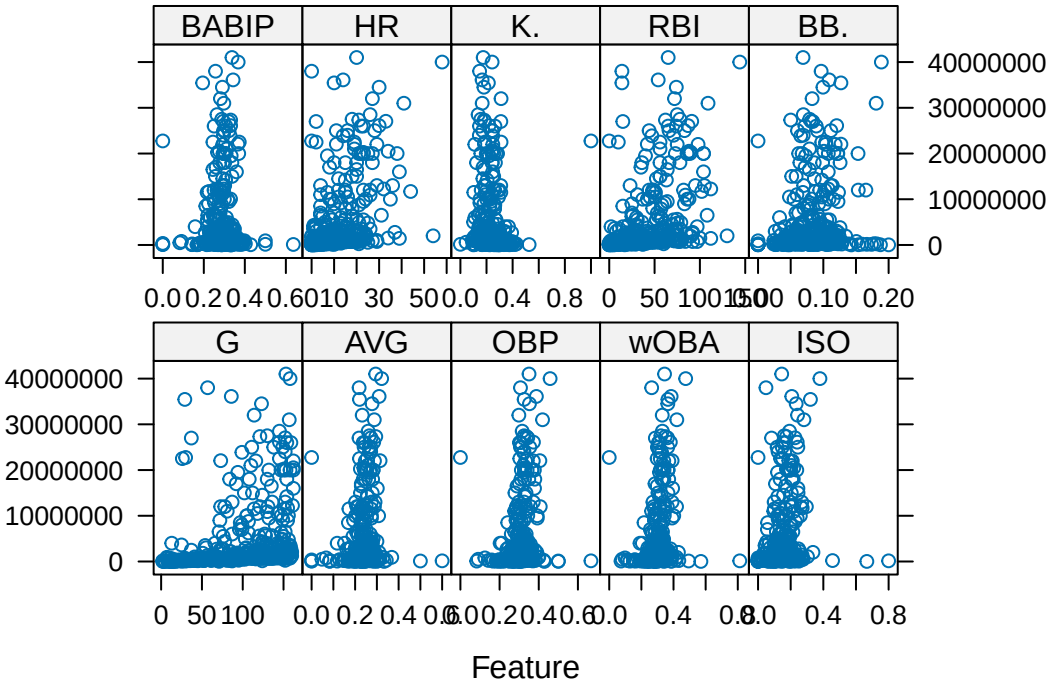
To identify which performance metrics most strongly contribute to player salary, I used the `varImp()` function from the `caret` package. This function fits a linear model and ranks predictor variables by their relative importance based on their contribution to reducing model error.

Rationale for Predictor Selection - Batters



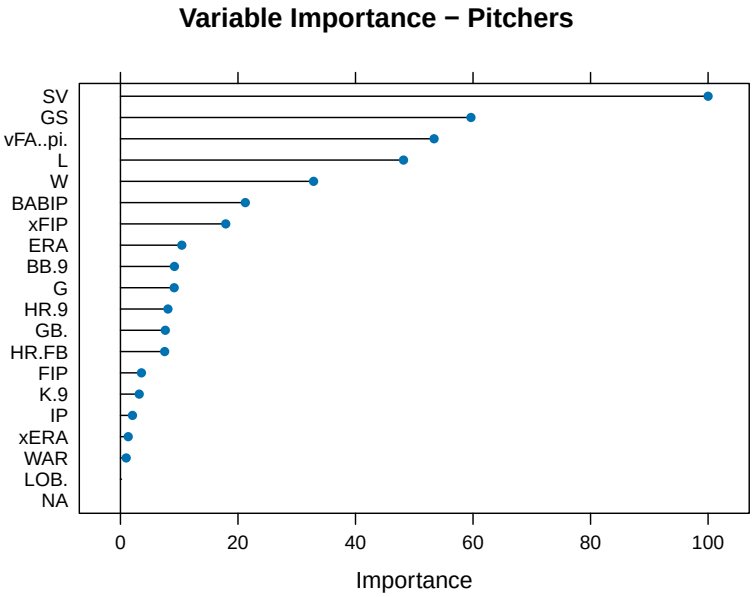
Based on the variable importance plot, for batters, I selected the following predictors because they ranked in the **top 10 based on variable importance** from the output:

G, AVG, OBP, wOBA, ISO, BABIP, HR, K., RBI, and BB.



As shown in the feature plots above, all chosen predictors exhibit a general positive trend with salary, confirming their relevance in the model.

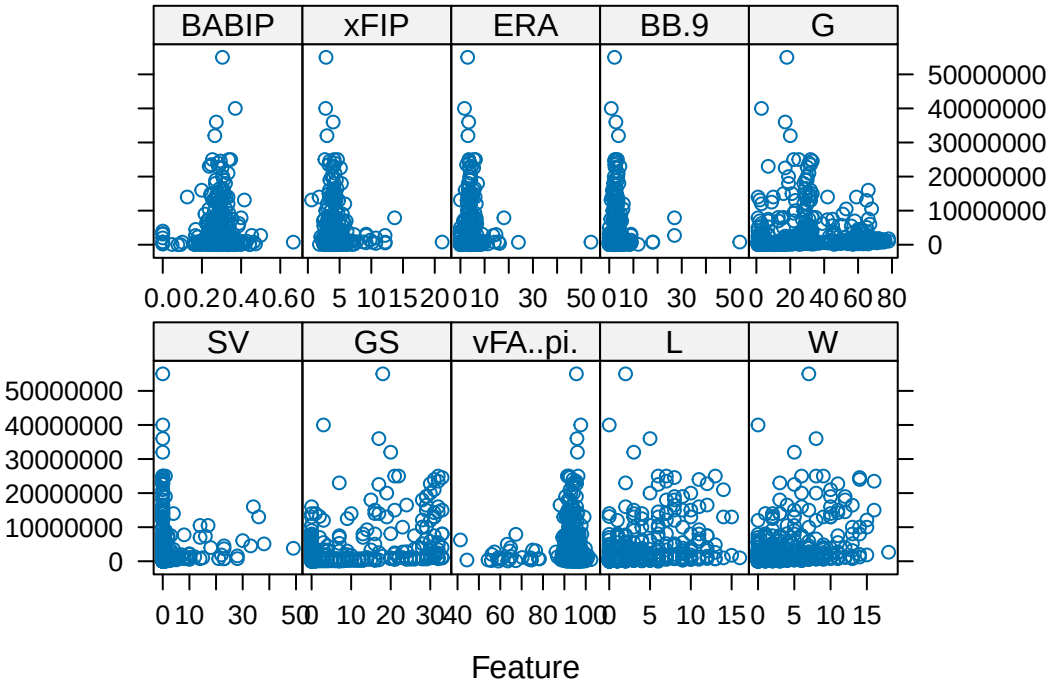
Rationale for Predictor Selection – Pitchers



Based on the variable importance plot, for pitchers, I selected the following predictors because they ranked in the **top 10 based on variable importance** from the output:

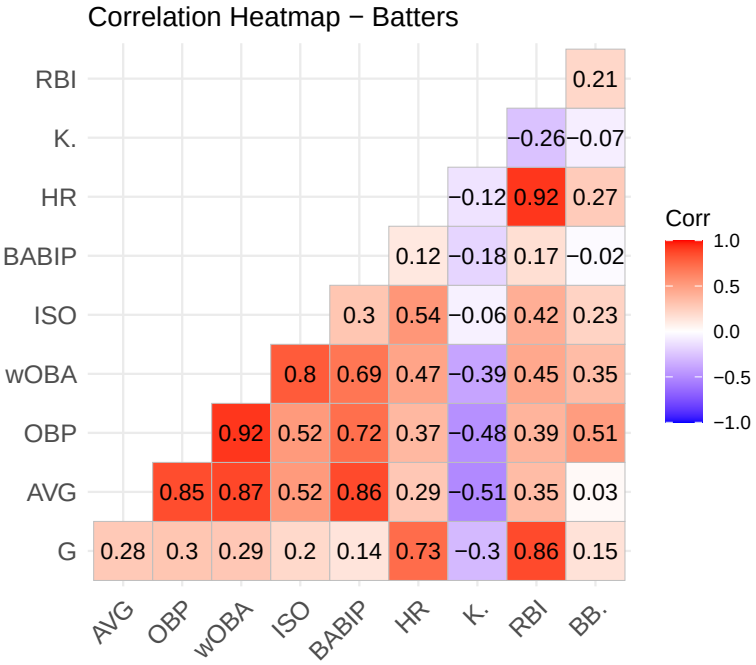
SV, GS, vFA..pi, L, W, BABIP, xFIP, ERA, BB.9, and G.

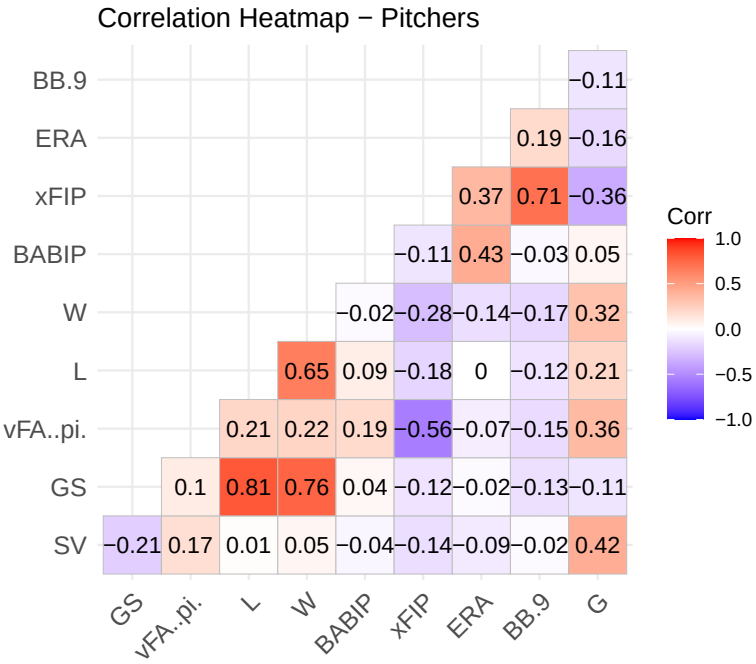
These variables capture various aspects of a pitcher's performance, including **velocity, durability, control, and run prevention**.



As shown in the feature plots above, majority of chosen predictors exhibit a general positive trend with salary, confirming their relevance in the model.

2.5. Correlation Analysis





The correlation heatmaps for both batters and pitchers reveal that some of the selected predictor variables are highly correlated with one another. This suggests the presence of multicollinearity, which can inflate coefficient estimates and reduce model interpretability in ordinary linear regression.

To address this issue, I applied regularization techniques, specifically Ridge and Lasso regression. This approach ensures more reliable estimates and helps reduce overfitting caused by redundant variables.

3. Regularization: Ridge and Lasso Regression

To address potential multicollinearity and avoid overfitting, I applied two types of regularized regression:

- **Ridge regression** ($\alpha = 0$): shrinks coefficients but keeps all predictors.
- **Lasso regression** ($\alpha = 1$): can shrink some coefficients to zero, simplifying the model.

I used cross-validation to tune the lambda (penalty) and compared the models in terms of both **predictive accuracy** and **model simplicity**.

To determine which model performs better, I compared their cross-validation errors. A lower cross-validation error indicates better generalization to unseen data. This comparison helps select the model that balances predictive accuracy and interpretability.

Batters

```
batter_train <- batter_train_clean_filtered %>%
select(Salary, G, AVG, OBP, wOBA, ISO, BABIP, HR, K., RBI, BB.) #Selected variables

x_batter <- model.matrix(Salary ~ ., data = batter_train)[, -1]
y_batter <- batter_train$Salary
grid <- 10^seq(5, -2, length = 100)

# Ridge
ridge_mod <- glmnet(x_batter, y_batter, alpha = 0, lambda = grid)
set.seed(123)
cv_ridge <- cv.glmnet(x_batter, y_batter, alpha = 0, lambda = grid)
ridge_cv_error <- min(cv_ridge$cvm)

# Lasso
lasso_mod <- glmnet(x_batter, y_batter, alpha = 1, lambda = grid)
set.seed(123)
cv_lasso <- cv.glmnet(x_batter, y_batter, alpha = 1, lambda = grid)
lasso_cv_error <- min(cv_lasso$cvm)
```

Table 1: Cross Validation Error

RegressionType	CV_error
Ridge	49901348177771
Lasso	49911624981970

Final Model Choice - Batters

Since the cross-validation errors of Ridge and Lasso were close in magnitude, I calculated the percentage difference between them to better assess the significance of the gap.

```
error_diff <- abs(ridge_cv_error - lasso_cv_error)
percent_diff <- (error_diff / ridge_cv_error) * 100
percent_diff
```

```
[1] 0.02059424
```

Rationale

The cross-validation error difference between Ridge and Lasso was only 0.02%, indicating similar predictive performance. Therefore, **I selected Lasso regression** for its ability to simplify the model by setting irrelevant coefficients to zero, resulting in a more interpretable and efficient model.

Pitchers

```
pitcher_train <- pitcher_train_clean_filtered %>%
  select(Salary, SV, GS, vFA..pi., L, W, BABIP, xFIP, ERA, BB.9, G)

x_pitcher <- model.matrix(Salary ~ ., data = pitcher_train)[, -1]
y_pitcher <- pitcher_train$Salary

grid <- 10^seq(5, -2, length = 100)

# Ridge
ridge_mod_pitcher <- glmnet(x_pitcher, y_pitcher, alpha = 0, lambda = grid)
set.seed(123)
cv_ridge_pitcher <- cv.glmnet(x_pitcher, y_pitcher, alpha = 0, lambda = grid)
ridge_cv_error_pitcher <- min(cv_ridge_pitcher$cvm)

# Lasso
lasso_mod_pitcher <- glmnet(x_pitcher, y_pitcher, alpha = 1, lambda = grid)
set.seed(123)
cv_lasso_pitcher <- cv.glmnet(x_pitcher, y_pitcher, alpha = 1, lambda = grid)
lasso_cv_error_pitcher <- min(cv_lasso_pitcher$cvm)
```

Table 2: Cross Validation Error

RegressionType	CV_error
Ridge	32022232416330
Lasso	32036345575839

Final Model Choice - Pitchers

```
error_diff_pitcher <- abs(ridge_cv_error_pitcher - lasso_cv_error_pitcher)
percent_diff_pitcher <- (error_diff_pitcher / ridge_cv_error_pitcher) * 100
percent_diff_pitcher
```

```
[1] 0.044073
```

Rationale

The cross-validation error difference between Ridge and Lasso was only 0.04%, indicating similar predictive performance. Therefore, I selected **Lasso regression** for its ability to simplify the model by setting irrelevant coefficients to zero, resulting in a more interpretable and efficient model.

4. Model Evaluation

To assess the predictive performance of the Lasso regression model, I split the dataset of players with known salaries into training and test sets using an 80-20 split. The Lasso model is trained on the 80% training subset and then used to predict salaries for the remaining 20% test subset. Since the test set consists of players whose actual salaries are known, this allows for an objective evaluation of the model's accuracy. By comparing the predicted and actual salaries, I can quantify how well the model generalizes to unseen data.

4.1. Train-Test Split

Batters

```
set.seed(123)
batter_index <- createDataPartition(batter_train$Salary, p = 0.8, list = FALSE)
batter_train_split <- batter_train[batter_index, ]
batter_test_split <- batter_train[-batter_index, ]
```

Pitchers

```
set.seed(123)
pitcher_index <- createDataPartition(pitcher_train$Salary, p = 0.8, list = FALSE)
pitcher_train_split <- pitcher_train[pitcher_index, ]
pitcher_test_split <- pitcher_train[-pitcher_index, ]
```

4.2 Train Lasso

Batters

```
set.seed(123)
batter_lasso_model <- train(
  Salary ~ G + AVG + OBP + wOBA + ISO + BABIP + HR + K. + RBI + BB.,
  data = batter_train_split,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = 10^seq(5, -2, length = 100))
)
```

Pitchers

```
set.seed(123)
pitcher_lasso_model <- train(
  Salary ~ SV + GS + vFA..pi. + L + W + BABIP + xFIP + ERA + BB.9 + G,
  data = pitcher_train_split,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = 10^seq(5, -2, length = 100))
)
```

4.3 Make Predictions on the test data (20% test set)

Batters

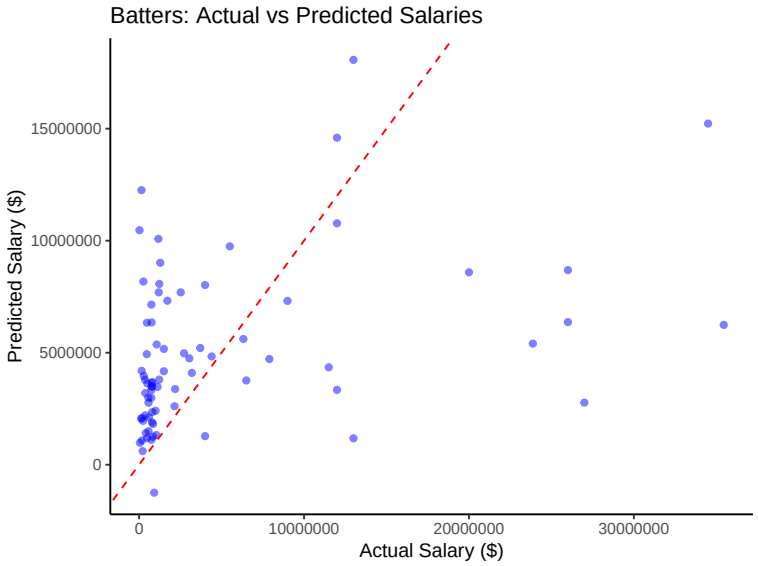
```
pred_batter <- predict(batter_lasso_model, newdata = batter_test_split)
```

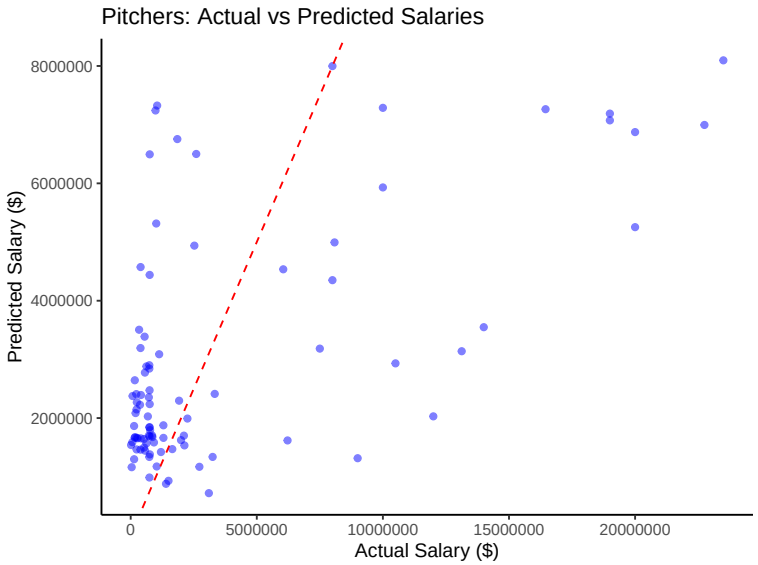
Pitchers

```
pred_pitcher <- predict(pitcher_lasso_model, newdata = pitcher_test_split)
```

4.4 Performance Evaluation

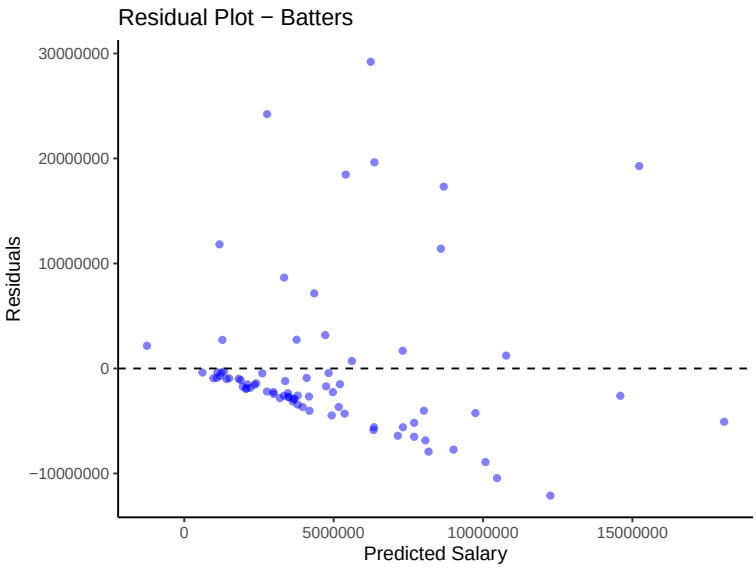
Visualizations: Actual vs. Predicted

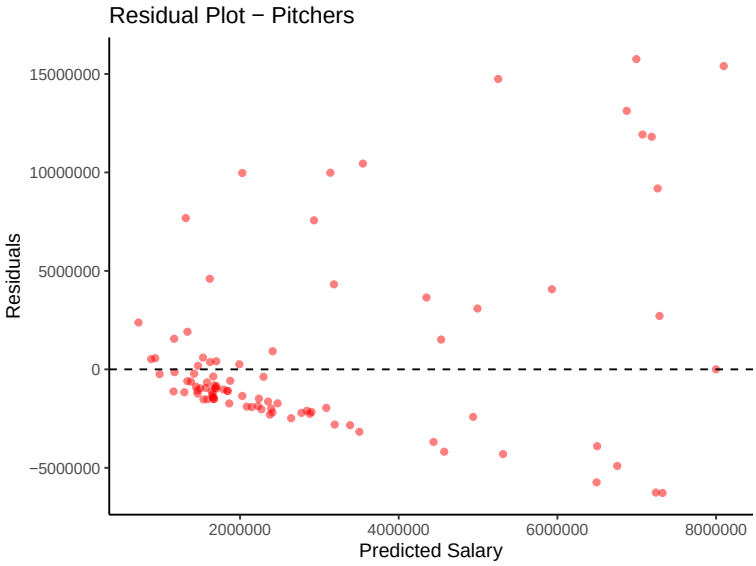




Although the scatter plots provide a visual overview of predicted vs. actual salaries, they show significant dispersion particularly at higher salary ranges. Many predicted salaries fall well below the actual salaries for top-earning players, and predictions for mid-range players also appear scattered. This visual pattern indicates that the model struggles to accurately capture the full range of salary values. To supplement the plots, I used Residual Plot for the further visualization and used RMSE and R-squared for quantitative evaluation.

Residual Plots





- **Non-Random Pattern**: The residuals are not evenly scattered around the zero line. Most of the points hug near-zero for lower predicted salaries, but as predicted salaries increase, residuals become extremely spread out.
- **Heteroscedasticity**: There's a clear funnel shape. The variance of residuals increases with predicted salary. This violates the assumption of homoscedasticity (constant variance), which is crucial for reliable inference in linear models.
- **Therefore, we can say both residual plots are problematic.**

Statistical Evaluation of Model Accuracy

RMSE

PlayerType	RMSE
Batters	7447949
Pitchers	4782136

The RMSE for batters and pitchers is approximately **$7.4M and $4.8M**, respectively, which may seem high at first glance. However, it is important to contextualize these values: a small number of very high salaries (e.g., $30M+) inflate the error, while the model performs much better for players earning less than $5M, which is the majority.

R-squared

Table 4: R-squared Comparison by Player Type

PlayerType	R.squared
Batters	0.1532794
Pitchers	0.3019531

While an R-squared of **0.15 for batters and 0.30 for pitchers** may be considered moderate to strong in the context of social science data, the heteroscedastic pattern observed in the residuals suggests room for improvement. In particular, the increasing variance of residuals at higher salary levels indicates that a transformation could help stabilize variance and improve model performance.

To address this, I applied a logarithmic transformation to the salary variable.

5. Model Transformation

5.1. Model Refit with log(Salary)

Log-transform Salary

```
batter_train_data <- batter_train_data %>%
  mutate(log_Salary = log(Salary))

pitcher_train_data <- pitcher_train_data %>%
  mutate(log_Salary = log(Salary))
```

Update the previous train-test split and model formulas to use log_Salary

Batters

```
set.seed(123)

batter_index <- createDataPartition(batter_train_data$log_Salary, p = 0.8, list = FALSE)
```

```r
batter_train_split <- batter_train_data[batter_index, ] %>%
  select(Salary, log_Salary, G, AVG, OBP, wOBA, ISO, BABIP, HR, K., RBI, BB.) %>%
  drop_na()

batter_test_split <- batter_train_data[-batter_index, ] %>%
  select(Salary, log_Salary, G, AVG, OBP, wOBA, ISO, BABIP, HR, K., RBI, BB.) %>%
  drop_na()
```

Pitchers

```r
set.seed(123)
pitcher_index <- createDataPartition(pitcher_train_data$log_Salary, p = 0.8, list = FALSE)

pitcher_train_split <- pitcher_train_data[pitcher_index, ] %>%
  select(Salary, log_Salary, SV, GS, vFA..pi., L, W, BABIP, xFIP, ERA, BB.9, G) %>%
  drop_na()

pitcher_test_split <- pitcher_train_data[-pitcher_index, ] %>%
  select(Salary, log_Salary, SV, GS, vFA..pi., L, W, BABIP, xFIP, ERA, BB.9, G) %>%
  drop_na()
```

Train the Lasso models with log_Salary

Batters

```r
set.seed(123)
batter_lasso_model_log <- train(
  log_Salary ~ G + AVG + OBP + wOBA + ISO + BABIP + HR + K. + RBI + BB.,
  data = batter_train_split,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = 10^seq(5, -2, length = 100))
)
```

Pitchers

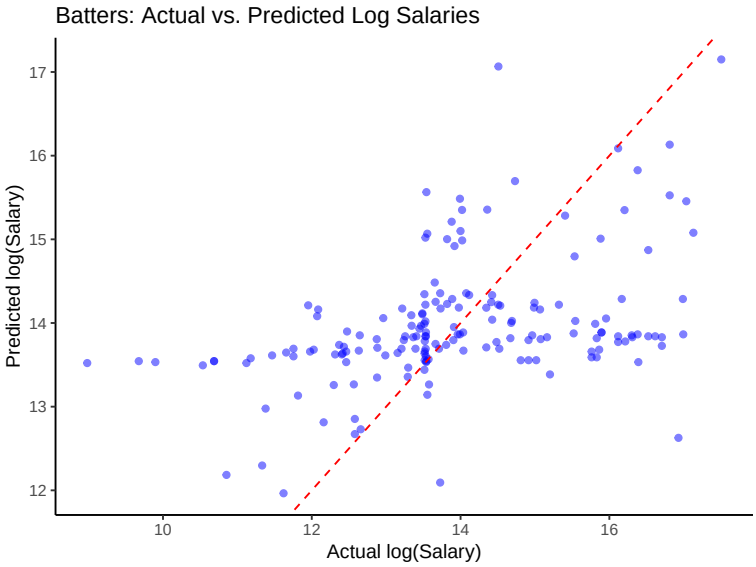
```r
set.seed(123)
pitcher_lasso_model_log <- train(
  log_Salary ~ SV + GS + vFA..pi. + L + W + BABIP + xFIP + ERA + BB.9 + G,
  data = pitcher_train_split,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = 10^seq(5, -2, length = 100))
)
```

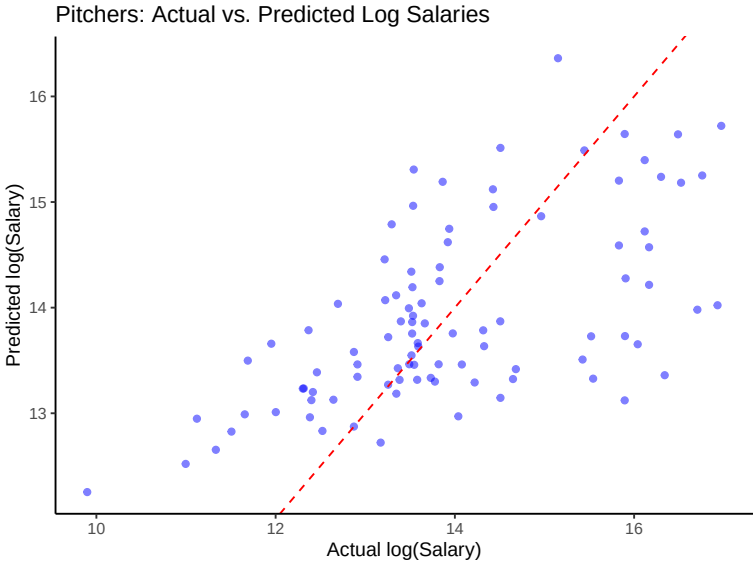
Make Predictions On the Test Data

```r
pred_log_batter <- predict(batter_lasso_model_log, newdata = batter_test_split)
pred_log_pitcher <- predict(pitcher_lasso_model_log, newdata = pitcher_test_split)
```

5.2. Performance Evaluation

Visualizations

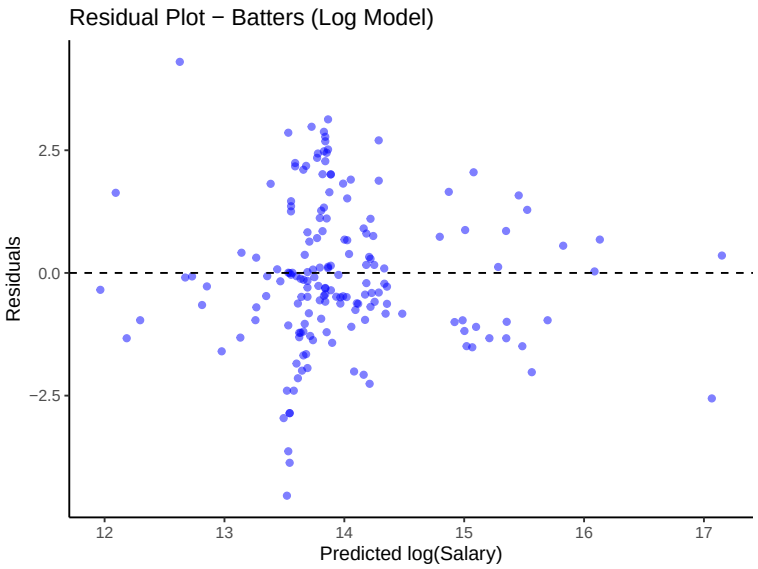


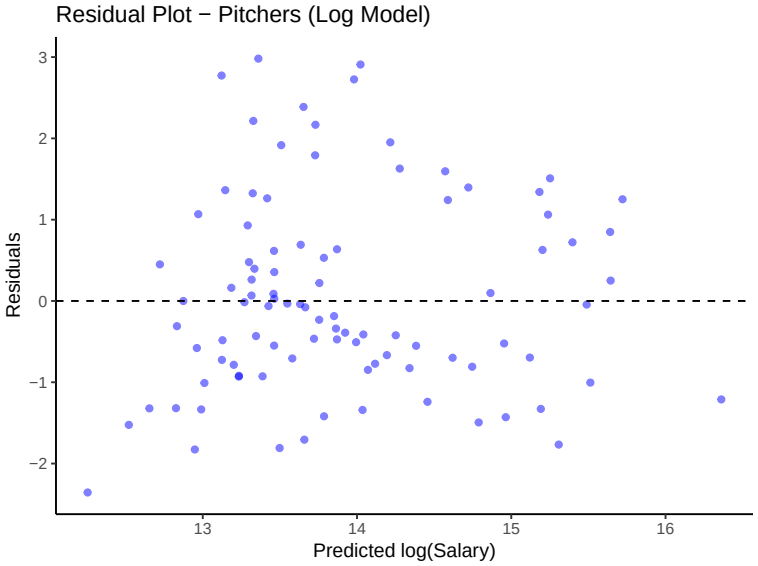


Similar to Section 4, this scatter plot shows a general upward trend between actual and predicted log-transformed salaries for batters. However, the points are spread widely and do not closely cluster around the diagonal reference line.

Again, the visual is too vague to assess accuracy, so we proceed to examine residual plots, RMSE, and R^2 to evaluate model performance more precisely.

Residual Plots (Back-transformed)





- The residuals appear randomly scattered around zero.
- There's no obvious pattern, curvature, or funnel shape.
- The variance of residuals seems relatively constant, which suggests homoscedasticity.
- Therefore,these residual plots look good and don't raise major concerns. It is clear that the log transformation helped.

Statistical Evaluation

Table 5: RMSE Comparison by Player Type

PlayerType	Log_RMSE	RMSE
Batters	1.487244	4.424883
Pitchers	1.195997	3.306853

The RMSE on the **log scale** is approximately **1.49 for batters and 1.20 for pitchers**. Since this scale can be unintuitive, I exponentiated these values to interpret them in multiplicative terms. The exponentiated RMSE values are **4.42 for batters and 3.31 for pitchers**. This means that, on average, the predicted salaries differ from the actual salaries by a factor of around **4.4 times** and **3.3 times**, respectively. While this might seem large, these are typical margins in salary prediction due to the highly skewed nature of player earnings in professional sports.

Table 6: R-squared Comparison by Player Type

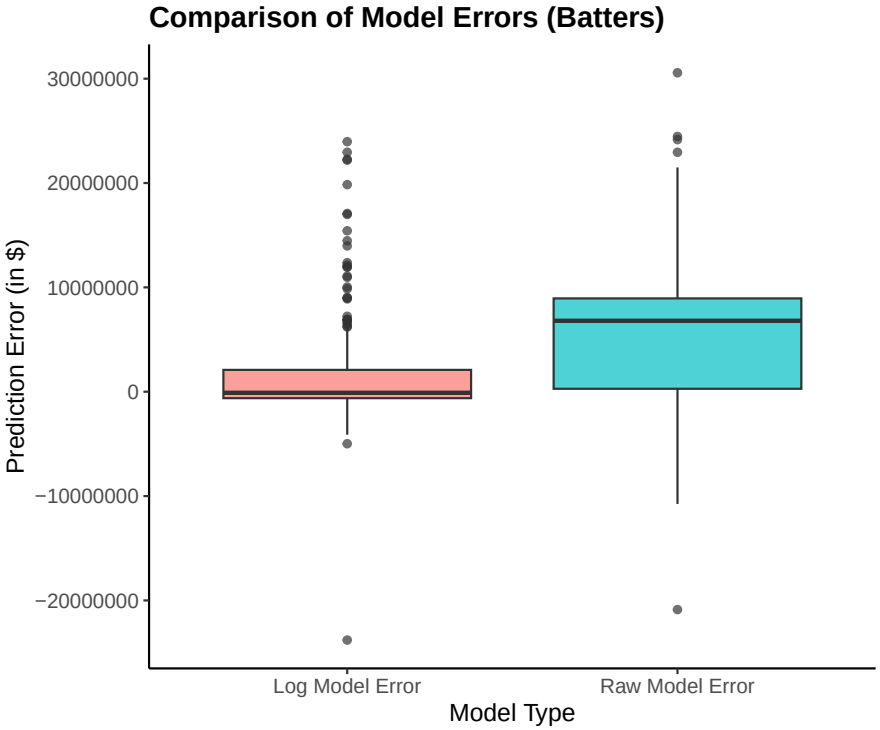
PlayerType	R.squared
Batters	0.1850430
Pitchers	0.3673797

The model explains **18.5% of the variation in log-salaries for batters and 36.7% for pitchers**. This marks a slight improvement over the untransformed model, where the R-squared values were 0.15 and 0.30, respectively. While these figures may seem low from a purely mathematical standpoint, they are generally considered acceptable within the context of social science data, where outcomes are often influenced by unobservable factors.

Given the improvement from the raw model, I considered the log-transformed model reasonably valid. However, since the change in R-squared was not substantial, I proceeded to compare both models more directly by analyzing their prediction errors. Specifically, I evaluated the actual vs. predicted salaries and computed the differences to determine which model performs better in practice.

Table 7: Comparison of Salary Predictions (Batters)

Actual Salary	Log Model Prediction	Log Model Error	Raw Model Prediction	Raw Model Error
40000000	28063389	11936611	30271331	9728669
2000000	25798894	-23798894	22877876	-20877876
10000000	9697664	302336	18734630	-8734630
2500000	6556847	-4056847	13255820	-10755820
25000000	5152204	19847796	14036979	10963021
1721471	4661543	-2940072	7319357	-5597886
4901335	4339720	561615	8143202	-3241867
1226750	4643923	-3417173	8070799	-6844049
20000000	5531410	14468590	9469011	10530989
1202146	3609499	-2407353	5252018	-4049872

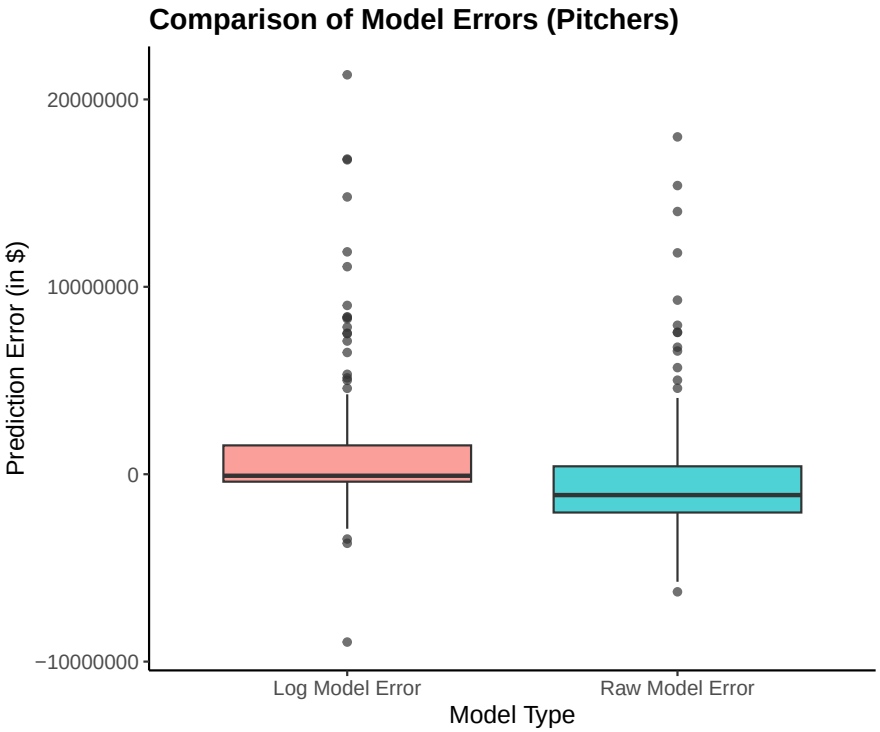


Based on the boxplot comparing errors from the log-transformed model and the raw salary model, **the log model clearly performs better**. The median error is closer to zero, indicating lower bias, and the interquartile range is narrower, showing less variance in prediction errors. In contrast, the raw model has a wider spread and larger outliers, suggesting it struggles more with high-salary players. Overall, the log-transformed model offers more consistent and reliable predictions, making it the better choice for imputing missing salaries.

Additionally, statistical analysis in the previous sections also supports this conclusion — the log model has a lower RMSE and a higher R-squared value, indicating better overall predictive accuracy. Therefore, I will proceed with the log-transformed model for imputing missing salaries, as it provides more consistent and reliable predictions.

Table 8: Comparison of Salary Predictions (Pitchers)

Actual Salary	Log Model Prediction	Log Model Error	Raw Model Prediction	Raw Model Error
23500000	6725007	16774993	8096153	15403847
8000000	6225875	1774125	7998318	1682
10000000	4861571	5138429	7286851	2713149
15000000	3924978	11075022	7050979	7949021
1049617	3959230	-2909613	7325851	-6276234
14500000	6205688	8294312	7721843	6778157
19000000	4203747	14796253	7188629	11811371
1013438	1545978	-532540	5315603	-4302165
1111483	2235083	-1123600	5965006	-4853523
3800000	12756521	-8956521	6924766	-3124766



For pitchers, the log-transformed model is still the better choice. The boxplot shows that its median prediction error is closer to zero, indicating lower bias, and the error distribution is more centered compared to the raw model. Although both models have similar outliers, the raw model displays greater variance in prediction errors. Combined with stronger RMSE and R-squared values from the statistical evaluation, the log model offers more consistent and reliable performance, making it the preferred option for imputing missing salaries.

6. Imputing missing salaries

6.1 Prediction

Prepare clean missing datasets

```
# For Batters
batter_missing_ready <- batter_missing_data %>%
  select(G, AVG, OBP, wOBA, ISO, BABIP, HR, K., RBI, BB.) %>%
  drop_na()

# For Pitchers
pitcher_missing_ready <- pitcher_missing_data %>%
  select(SV, GS, vFA..pi., L, W, BABIP, xFIP, ERA, BB.9, G) %>%
  drop_na()
```

Predict log(Salary), then back-transform to dollar scale

```
# Predict for batters
pred_log_missing_batter <- predict(batter_lasso_model_log, newdata = batter_missing_ready)
pred_salary_batter <- exp(pred_log_missing_batter)

# Predict for pitchers
pred_log_missing_pitcher <- predict(pitcher_lasso_model_log, newdata = pitcher_missing_ready)
pred_salary_pitcher <- exp(pred_log_missing_pitcher)
```

6.2 Integration with Original Dataset

```
# Batter
batter_missing_data_imputed <- batter_missing_data %>%
  filter(complete.cases(select(., G, AVG, OBP, wOBA, ISO, BABIP, HR, K., RBI, BB.))) %>%
  mutate(Imputed_Salary = pred_salary_batter)

# Pitcher
pitcher_missing_data_imputed <- pitcher_missing_data %>%
  filter(complete.cases(select(., SV, GS, vFA..pi., L, W, BABIP, xFIP, ERA, BB.9, G))) %>%
  mutate(Imputed_Salary = pred_salary_pitcher)
```

To evaluate real-world applicability, I manually searched and compared the model's predicted salaries with a few publicly available player salaries from reliable sources.

Batters

Randomly select 5 batters with imputed salaries

Name	Imputed_Salary
Dedniel Núñez	980260.6
César Salazar	289342.0
Martín Maldonado	647614.1
Sammy Peralta	816383.2
Wander Suero	745024.1

Actual Salaries of Selected Players (USD)

- **Dedniel Núñez**: $722,600
- **César Salazar**: $740,000
- **Martín Maldonado**: $760,000
- **Sammy Peralta**: $750,000
- **Wander Suero**: $1,400,000

Table 10: Comparison of Predicted and Actual Salaries for Selected Batters

Name	Predicted Salary (USD)	Actual Salary (USD)	Prediction Error (USD)	Prediction Error (%)
Dedniel Núñez	980261	722600	257661	36
César Salazar	289342	740000	-450658	-61
Martín Maldonado	647614	760000	-112386	-15
Sammy Peralta	816383	750000	66383	9
Wander Suero	745024	1400000	-654976	-47

This comparison shows that while the model can yield reasonably close predictions for some players, there are noticeable misestimations for others. Including qualitative data (e.g., contract history or age) might enhance the model in the future.

Pitchers

Randomly select 5 pitchers with imputed salaries

Name	Imputed_Salary
Josh Winder	285214.3
John Brebbia	733226.8
Geoff Hartlieb	401820.5
Jack O'Loughlin	372044.6
DJ Herz	1326941.4

Actual Salaries of Selected Players (USD)

- **Josh Winder**: $740,000
- **John Brebbia**: $1,500,000
- **Geoff Hartlieb**: $720,000
- **Jack O'Loughlin**: $740,000
- **DJ Herz**: $740,000

Table 12: Comparison of Predicted and Actual Salaries for Selected Pfitchers

Name	Predicted Salary (USD)	Actual Salary (USD)	Prediction Error (USD)	Prediction Error (%)
Josh Winder	285214	740000	-454786	-62
John Brebbia	733227	1500000	-766773	-51
Geoff Hartlieb	401821	720000	-318179	-44
Jack O'Loughlin	372045	740000	-367955	-50
DJ Herz	1326941	740000	586941	79

The model significantly underestimates salaries for several pitchers. This is likely because it does not incorporate the MLB minimum salary threshold (e.g., $720,000 in 2023), which serves as a binding floor for most

players. As a result, predictions for lower-earning players—especially those with limited data or performance history—can fall below this threshold, contributing to large negative errors and skewed accuracy metrics.

To address this problem, I applied a salary floor of $720,000 directly to the model's predictions during the imputation step. This ensures that no player is predicted to earn below the league minimum, improving the realism of the output.

Apply Minimum Salary Cap and Predict Again

Examine the Real Life Applications

Table 13: Comparison of Capped Predicted and Actual Salaries for Selected Batters

Name	Predicted Salary (USD)	Actual_Salary	Prediction Error (USD)	Prediction Error (%)
Dedniel Núñez	980261	722600	257661	36
César Salazar	720000	740000	-20000	-3
Martín Maldonado	720000	760000	-40000	-5
Sammy Peralta	816383	750000	66383	9
Wander Suero	745024	1400000	-654976	-47

- Previously, predictions for low-salary players like César Salazar and Martín Maldonado were heavily underestimated, likely because the model didn't account for the salary floor.
- After capping, predicted values are no longer below the minimum threshold, dramatically reducing percentage errors.
- Overall variance is reduced, and errors are generally closer to zero.

Table 14: Comparison of Capped Predicted and Actual Salaries for Selected Pitchers

Name	Predicted Salary (USD)	Actual_Salary	Prediction Error (USD)	Prediction Error (%)
Josh Winder	720000	740000	-20000	-3
John Brebbia	733227	1500000	-766773	-51
Geoff Hartlieb	720000	720000	0	0
Jack O'Loughlin	720000	740000	-20000	-3
DJ Herz	1326941	740000	586941	79

- The improvements are even more dramatic for pitchers.
- Previously, 4 out of 5 pitchers had errors between -44% and -62%, showing the model failed to recognize the minimum wage constraint.
- After applying the cap, most errors now fall within a -3% to 0% range, indicating that the capped model aligns much better with real-world conditions.
- However, for both batters and pitchers, capping doesn't resolve all limitations — particularly in handling extreme underestimations for higher-paid players or in preserving interpretability across the model.

7. Conclusion

This project aimed to predict MLB player salaries using performance statistics from Fangraphs. Throughout the analysis, I followed a structured modeling approach:

- First, I selected **key performance predictors** for both batters and pitchers based on their relative **variable importance** from linear regression models.
- I then addressed **multicollinearity and overfitting** by applying **regularized regression techniques**, specifically **Ridge** and **Lasso** regression.
- Using **cross-validation**, I compared both models and ultimately selected **Lasso regression** for its simplicity and comparable predictive accuracy.
- To stabilize variance and reduce skewness in salary data, I applied a **logarithmic transformation** to the salary variable.
- After evaluating both raw and log-transformed models, I found that the **log-Lasso model** offered improved predictive performance.

However, the model tended to **underestimate salaries for lower-paid players**, particularly those earning around the MLB minimum wage. To address this, I introduced a **capped minimum salary of $720,000**, ensuring predictions respect the league's salary floor. This significantly improved predictions for lower-income players by eliminating extreme underestimations. Therefore, the **final model selected for salary prediction is the Lasso regression model with a logarithmic transformation of salary, coupled with a capped minimum salary**.

Possible Future Directions

While the current model provides a foundation for predicting MLB salaries using statistical performance metrics, the model still **struggles to predict high-end salaries accurately**. These discrepancies likely stem from omitted variables such as:

- Player popularity and branding
- Free agency leverage
- Team-specific budgets and contract history
- Player age and leadership roles

These factors are not reflected in performance statistics alone. In future studies, incorporating such qualitative or contractual data, as well as exploring alternative modeling techniques (e.g., quantile regression or ensemble methods), could further enhance salary prediction accuracy across the full spectrum of players.

Possible improvements:

- **Incorporate Qualitative and Contextual Variables**: Factors such as player age, contract length, injury history, fan popularity, and leadership qualities are known to influence salaries but were not captured in this analysis.
- **Account for Team and Market Effects**: Salaries can vary widely depending on the team's payroll capacity or market size. Including team-level fixed effects or franchise identifiers could improve model accuracy.
- **Use Advanced Performance Metrics**: Incorporating more granular or advanced statistics, such as WAR (Wins Above Replacement) or defensive ratings, could provide a more nuanced view of player value.

- **Explore Alternative Modeling Techniques**: Tree-based models (e.g., Random Forest, XGBoost) or quantile regression may offer better performance, particularly at the tails of the salary distribution.
- **Time Series or Panel Data Modeling**: A dynamic analysis that tracks player performance and salaries over multiple seasons could capture longitudinal trends and career progression more effectively.
- **Better Handling of Salary Censoring**: Instead of post hoc salary caps, future models could integrate **constrained regression** techniques or explicitly model minimum thresholds.

These directions would strengthen the model's generalizability and bring it closer to the real-world salary determination process observed in professional sports.

References

- Fangraphs. "2024 MLB Player Statistics – Batters and Pitchers." Retrieved from https://www.fangraphs.com
- The Baseball Cube. "MLB Player Payroll Data (2024)." Retrieved from https://www.thebaseballcube.com