map501_F416781

2024-11-25

Setting up

```
knitr::opts_chunk$set(
 results = "hold", echo = TRUE, eval = TRUE,
 message = FALSE, fig.width = 7, warning = FALSE,
 fig.height = 4, fig.align = "center"
)
```

Installing needed

```
## [1] "All specified libraries have been installed and loaded."
```

Loading libraries to be used

```
library("tidyverse")
 library("magrittr")
 library("here")
 library("janitor")
 library("gridExtra")
 library("readxl")
 library("Lahman")
 library("viridis")
 library("lindia")
 library("lme4")
 library("caret")
 library("pROC")
 library("car")
```

1. Linear Regression

a. Creating df_managers with win_pct and selected variables

```
# Create the df_managers dataset (W= Win, G= Game)
df_managers <- Managers %>%
  mutate(win_pct = W / G) %>%
  select(playerID, teamID, yearID, lgID, plyrMgr, win_pct)

# Check the structure of df_managers
str(df_managers)
```

```
## 'data.frame':    3749 obs. of  6 variables:
##  $ playerID: chr  "wrighha01" "woodji01" "paborch01" "lennobi01" ...
##  $ teamID  : Factor w/ 149 levels "ALT","ANA","ARI",..: 24 31 39 56 56 90 97 111 136 136
...
##  $ yearID  : int  1871 1871 1871 1871 1871 1871 1871 1871 1871 1871 ...
##  $ lgID    : Factor w/ 7 levels "AA","AL","FL",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ plyrMgr : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 ...
##  $ win_pct : num  0.645 0.679 0.345 0.357 0.4 ...
```

b. Create awards_man dataset

i. Create df_teams

```
df_teams <- Teams %>%
  select(yearID, teamID, DivWin, CS)

# Check the structure of df_teams
str(df_teams)
```

```
## 'data.frame':    3045 obs. of  4 variables:
##  $ yearID: int  1871 1871 1871 1871 1871 1871 1871 1871 1871 1872 ...
##  $ teamID: Factor w/ 149 levels "ALT","ANA","ARI",..: 24 31 39 56 90 97 111 136 142 8 ...
##  $ DivWin: chr  NA NA NA NA ...
##  $ CS    : int  16 21 8 4 15 12 10 24 13 18 ...
```

ii. Create man_teams by merging df_managers with df_teams and remove lgID

```
man_teams <- merge(df_managers, df_teams, by = c("yearID", "teamID")) %>%
  select(-lgID)

# Check the structure of man_teams
str(man_teams)
```

```
## 'data.frame':    3749 obs. of  7 variables:
##  $ yearID  : int  1871 1871 1871 1871 1871 1871 1871 1871 1871 1871 ...
##  $ teamID  : Factor w/ 149 levels "ALT","ANA","ARI",..: 24 31 39 56 56 90 97 111 136 136
...
##  $ playerID: chr  "wrighha01" "woodji01" "paborch01" "lennobi01" ...
##  $ plyrMgr : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 ...
##  $ win_pct : num  0.645 0.679 0.345 0.357 0.4 ...
##  $ DivWin  : chr  NA NA NA NA ...
##  $ CS      : int  16 21 8 4 4 15 12 10 24 24 ...
```

iii. Merge man_teams with AwardsShareManagers to create awards_man

```
awards_man <- merge(man_teams, AwardsShareManagers, by = c("yearID", "playerID"))

# Check the structure of awards_man
str(awards_man)
```

```
## 'data.frame':    511 obs. of  12 variables:
##  $ yearID    : int  1983 1983 1983 1983 1983 1983 1983 1984 1984 1984 ...
##  $ playerID  : chr  "altobjo01" "coxbo01" "larusto01" "lasorto01" ...
##  $ teamID    : Factor w/ 149 levels "ALT","ANA","ARI",..: 5 134 33 72 58 102 106 52 134 35
...
##  $ plyrMgr   : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ win_pct   : num  0.605 0.549 0.611 0.558 0.525 ...
##  $ DivWin    : chr  "Y" "N" "Y" "Y" ...
##  $ CS        : int  33 72 50 76 95 75 77 68 67 66 ...
##  $ awardID   : chr  "BBWAA Manager of the Year" "BBWAA Manager of the Year" "BBWAA Manager
of the Year" "BBWAA Manager of the Year" ...
##  $ lgID      : Factor w/ 2 levels "AL","NL": 1 1 1 2 2 2 2 1 1 2 ...
##  $ pointsWon : int  7 4 17 10 9 1 4 96 9 101 ...
##  $ pointsMax : int  28 28 28 24 24 24 24 140 140 120 ...
##  $ votesFirst: int  7 4 17 10 9 1 4 13 0 16 ...
```

iv. Add sqr_point_pct to awards_man

```
awards_man <- awards_man %>%
  mutate(sqr_point_pct = sqrt(pointsWon / pointsMax))
```

v. Remove incomplete cases and drop unused levels of teamID

```
awards_man <- awards_man %>%
  na.omit() %>%
  droplevels()

# Check the cleaned awards_man dataset
str(awards_man)
```

```
## 'data.frame':    486 obs. of  13 variables:
##  $ yearID       : int  1983 1983 1983 1983 1983 1983 1983 1984 1984 1984 ...
##  $ playerID     : chr  "altobjo01" "coxbo01" "larusto01" "lasorto01" ...
##  $ teamID       : Factor w/ 35 levels "ANA","ARI","ATL",..: 4 34 7 17 14 26 27 12 34 8 ...
##  $ plyrMgr      : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ win_pct      : num  0.605 0.549 0.611 0.558 0.525 ...
##  $ DivWin       : chr  "Y" "N" "Y" "Y" ...
##  $ CS           : int  33 72 50 76 95 75 77 68 67 66 ...
##  $ awardID      : chr  "BBWAA Manager of the Year" "BBWAA Manager of the Year" "BBWAA Mana
ger of the Year" "BBWAA Manager of the Year" ...
##  $ lgID         : Factor w/ 2 levels "AL","NL": 1 1 1 2 2 2 2 1 1 2 ...
##  $ pointsWon    : int  7 4 17 10 9 1 4 96 9 101 ...
##  $ pointsMax    : int  28 28 28 24 24 24 24 140 140 120 ...
##  $ votesFirst   : int  7 4 17 10 9 1 4 13 0 16 ...
##  $ sqr_point_pct: num  0.5 0.378 0.779 0.645 0.612 ...
##  - attr(*, "na.action")= 'omit' Named int [1:25] 123 124 125 126 127 128 129 130 131 132
...
##   ..- attr(*, "names")= chr [1:25] "123" "124" "125" "126" ...
```

c. Fit Gaussian model spp_mod

```
# Fit the model
spp_mod <- lm(sqr_point_pct ~ win_pct + DivWin + CS, data = awards_man)

# Summarize the results
summary(spp_mod)
```

```
##
## Call:
## lm(formula = sqr_point_pct ~ win_pct + DivWin + CS, data = awards_man)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52249 -0.18946 -0.03539  0.18009  0.66465
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.7278773  0.1409811  -5.163 3.56e-07 ***
## win_pct      1.8083774  0.2518894   7.179 2.67e-12 ***
## DivWinY      0.1364128  0.0266417   5.120 4.42e-07 ***
## CS           0.0027219  0.0006249   4.356 1.62e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2401 on 482 degrees of freedom
## Multiple R-squared:  0.271,  Adjusted R-squared:  0.2665
## F-statistic: 59.74 on 3 and 482 DF,  p-value: < 2.2e-16
```

The linear regression model predicts sqr_point_pct using win_pct, DivWin, and CS as predictors. The fitted model is:

```
    sqr_point_pct = -0.73 + 1.81*win_pct + 0.14*DivWinY + 0.003*CS
```

Intercept: The intercept (-0.73) indicates the baseline value of sqr_point_pct when all predictors are zero. While not directly meaningful in this context, it provides a reference point for the model.

win_pct: For every unit increase in win_pct, the sqr_point_pct is expected to increase by 1.81, holding other variables constant. This effect is highly significant (p-value = 2.67e-12), showing a strong positive relationship.

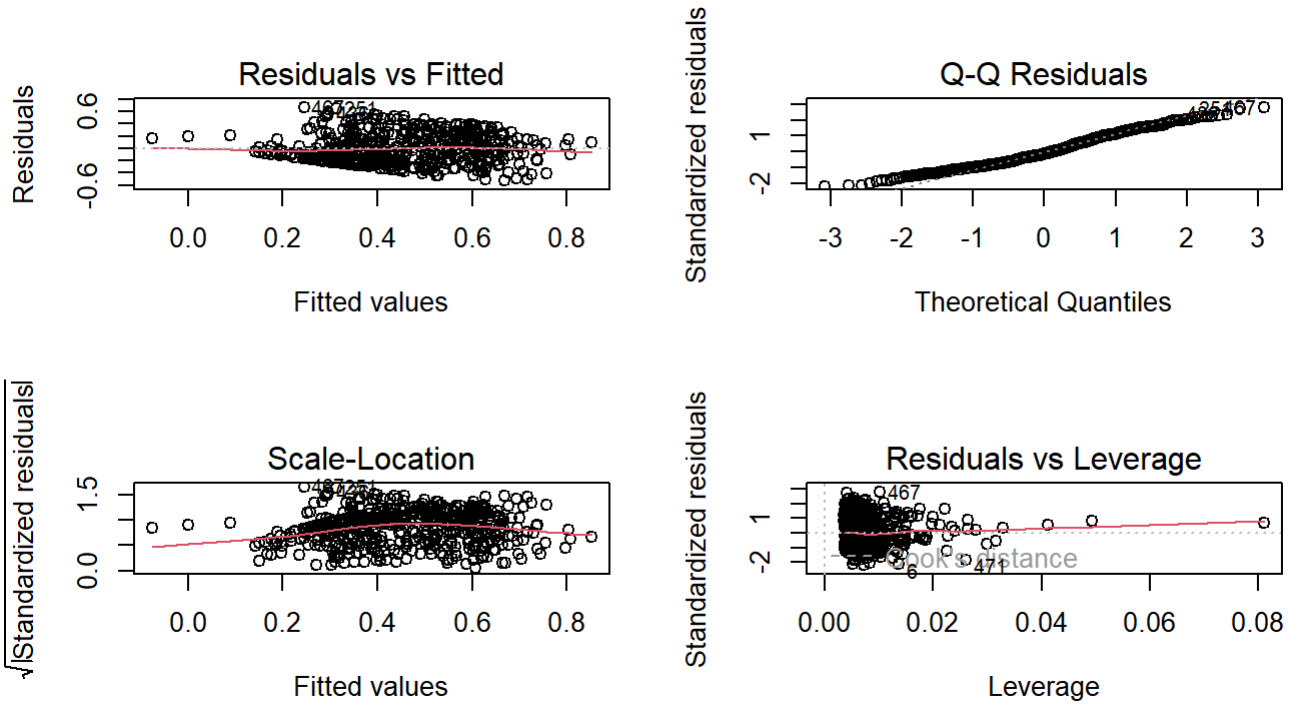
DivWin: Teams that win their division (DivWinY = 1) have an average sqr_point_pct that is 0.14 higher than those that do not (DivWinY = 0), controlling for other factors. This relationship is also highly significant (p-value = 4.42e-07).

CS: Each additional unit of CS corresponds to an increase of 0.003 in sqr_point_pct, with a highly significant p-value of 1.62e-05. Although the effect is small, it is statistically important.

Model fit: The model explains 27.1% of the variance in sqr_point_pct (Multiple R-squared = 0.271). The residual standard error is 0.2401, suggesting the model fits the data reasonably well.

d. Evaluate model assumptions

```
# Plot residuals to check assumptions
par(mfrow = c(2, 2))
plot(spp_mod)
```




```
# Check for multicollinearity
library(car)
vif(spp_mod)
```

```
##  win_pct   DivWin       CS
## 1.491421 1.467156 1.024769
```

e. Predict sqr_point_pct when win_pct = 0.8, DivWin = Yes, and CS = 8

```
# Create new data for prediction
new_data <- data.frame(
  win_pct = 0.8,
  DivWin = factor("Yes", levels = levels(awards_man$DivWin)),
  CS = 8
)

# Predict the expected value
predicted_value <- predict(spp_mod, new_data)
predicted_value
```

```
##  1
## NA
```

f. Construct 95% confidence intervals for parameter estimates

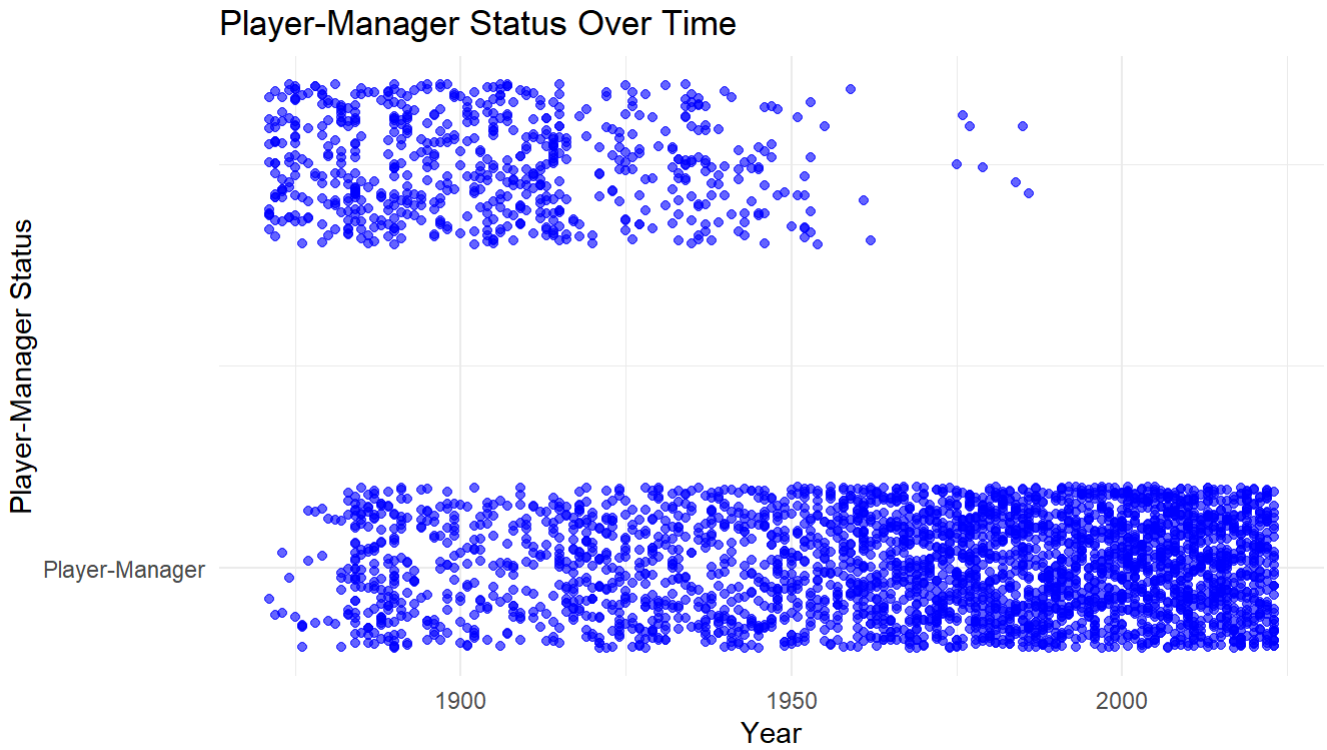
```
# Confidence intervals
conf_intervals <- confint(spp_mod, level = 0.95)
conf_intervals
```

```
##                   2.5 %       97.5 %
## (Intercept) -1.004890669 -0.450863854
## win_pct      1.313440582  2.303314302
## DivWinY      0.084064590  0.188760923
## CS           0.001493985  0.003949758
```

2. Logistic Regression

a. Plot plyrMgr Against yearID

```
# Jitter the points vertically to visualize plyrMgr vs. yearID
ggplot(df_managers, aes(x = yearID, y = as.numeric(plyrMgr))) +
  geom_jitter(height = 0.2, width = 0, color = "blue", alpha = 0.6) +
  scale_y_continuous(breaks = c(0, 1), labels = c("Not Player-Manager", "Player-Manager")) +
  labs(title = "Player-Manager Status Over Time", x = "Year", y = "Player-Manager Status") +
  theme_minimal()
```



Comment: The graph may show a decline in player-managers over time as specialization in the manager role increased.

b.Fitting Logistic Regression Model

```
# Logistic regression for plyrMgr as a function of yearID
logit_model <- glm(plyrMgr ~ yearID, data = df_managers, family = "binomial")

# Summarize the model
summary(logit_model)

# Report and interpret results
# Formula: log(odds) = β0 + β1 * yearID
# Extract and round coefficients
rounded_coefs <- round(coef(logit_model), 2)

# Print the rounded coefficients
rounded_coefs
```

```
##
## Call:
## glm(formula = plyrMgr ~ yearID, family = "binomial", data = df_managers)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 88.604237   3.412071   25.97   <2e-16 ***
## yearID      -0.046611   0.001779  -26.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3442.4  on 3748  degrees of freedom
## Residual deviance: 2127.7  on 3747  degrees of freedom
## AIC: 2131.7
##
## Number of Fisher Scoring iterations: 6
##
## (Intercept)      yearID
##       88.60       -0.05
```

The logistic regression model predicts the probability of being a player-manager (plyrMgr) based on the year (yearID). The fitted equation is:
{logit}(p) = 88.60 - 0.05*{yearID}

where p is the probability of a manager being a player.

The yearID coefficient (-0.05, p < 2e-16) indicates that each passing year decreases the log-odds of being a player-manager by 0.05. This corresponds to a 5% reduction in odds per year (e^{-0.05} approximately equal to 0.95), showing a significant decline in the prevalence of player-managers over time.

The model significantly reduces variability, as seen in the drop from a null deviance of 3442.4 to a residual deviance of 2127.7, with an AIC of 2131.7, suggesting a good fit. This supports the conclusion that being a player-manager has become increasingly rare over time.

c. Checking for Overfitting Using 80%-20% Split

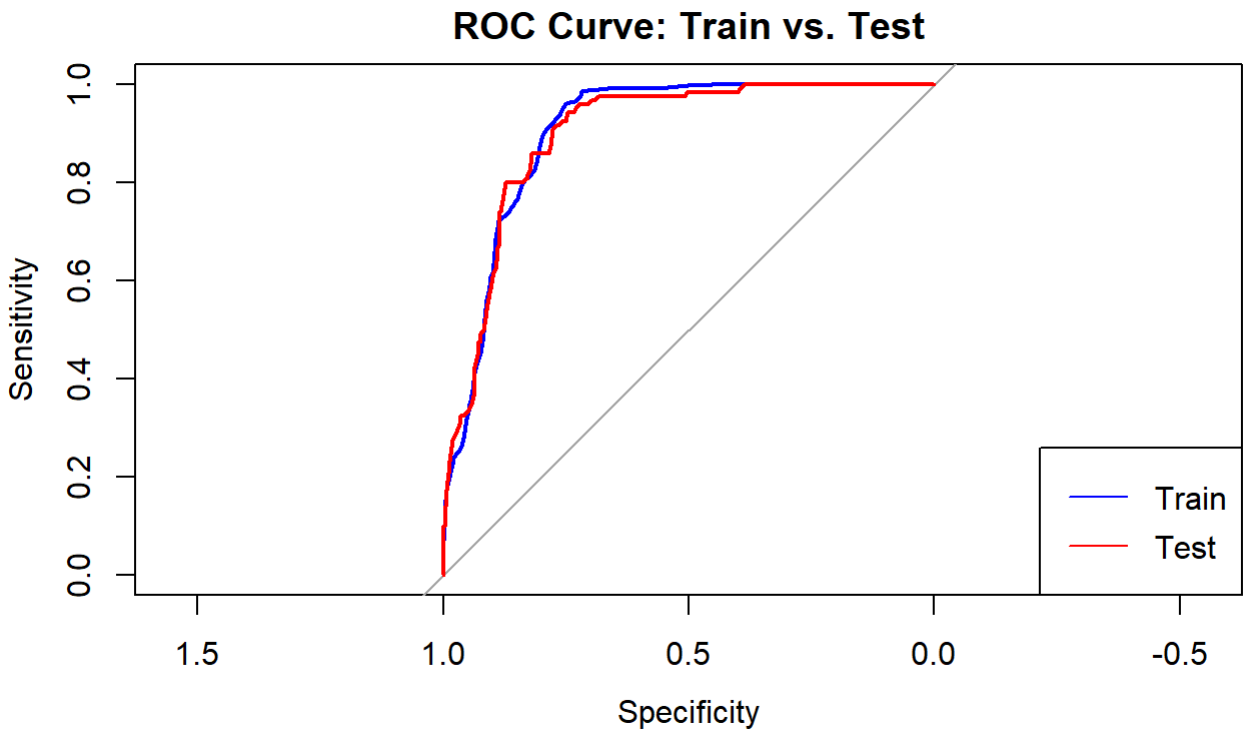
```
set.seed(123)
train_indices <- sample(seq_len(nrow(df_managers)), size = 0.8 * nrow(df_managers))
train_data <- df_managers[train_indices, ]
test_data <- df_managers[-train_indices, ]
```

Fit Model on Training Data

```
train_model <- glm(plyrMgr ~ yearID, data = train_data, family = "binomial")
```

ROC Curves

```
library(pROC)

# Predict probabilities for training and testing data
train_probs <- predict(train_model, newdata = train_data, type = "response")
test_probs <- predict(train_model, newdata = test_data, type = "response")

# Generate ROC curves
roc_train <- roc(train_data$plyrMgr, train_probs)
roc_test <- roc(test_data$plyrMgr, test_probs)

# Plot ROC curves
plot(roc_train, col = "blue", main = "ROC Curve: Train vs. Test")
plot(roc_test, col = "red", add = TRUE)
legend("bottomright", legend = c("Train", "Test"), col = c("blue", "red"), lty = 1)
```



The test curve lies above the train curve at a given point, this suggests superior performance. It is also evident that there is intersections between the curves, this indicate scenarios where one model might outperform the other depending on the specific threshold.

d. Youden’s Index and Confusion Matrices

```
# Make predictions on the training data
train_preds <- predict(logit_model, newdata = train_data, type = "response")

# Make predictions on the testing data
test_preds <- predict(logit_model, newdata = test_data, type = "response")
```

```
# For training Data
# Convert probabilities into binary outcomes (using 0.5 as the cutoff)
train_preds_class <- ifelse(train_preds > 0.5, 1, 0)

# Confusion matrix for training data
train_confusion_matrix <- table(Predicted = train_preds_class, Actual = train_data$plyrMgr)
print(train_confusion_matrix)

# Calculate Youden’s Index for training data
train_sensitivity <- train_confusion_matrix[2, 2] / (train_confusion_matrix[2, 1] + train_con
fusion_matrix[2, 2])
train_specificity <- train_confusion_matrix[1, 1] / (train_confusion_matrix[1, 1] + train_con
fusion_matrix[1, 2])
train_youden_index <- train_sensitivity + train_specificity - 1
print(paste("Training Youden's Index:", train_youden_index))
```

```
##          Actual
## Predicted    N    Y
##         0 2294  293
##         1  180  232
## [1] "Training Youden's Index: 0.449848195420718"
```

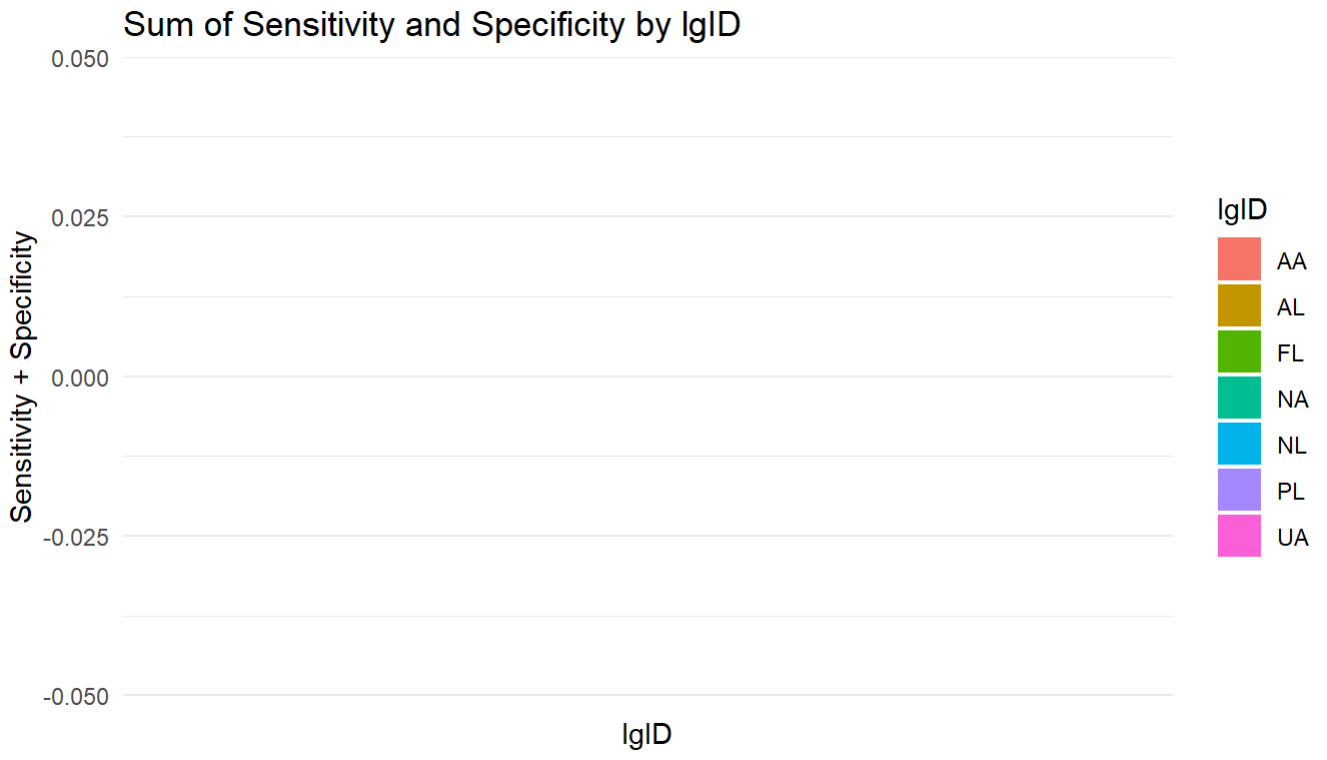
```
# For testing data
# Convert probabilities into binary outcomes (using 0.5 as the cutoff)
test_preds_class <- ifelse(test_preds > 0.5, 1, 0)

# Confusion matrix for testing data
test_confusion_matrix <- table(Predicted = test_preds_class, Actual = test_data$plyrMgr)
print(test_confusion_matrix)

# Calculate Youden’s Index for testing data
test_sensitivity <- test_confusion_matrix[2, 2] / (test_confusion_matrix[2, 1] + test_confusi
on_matrix[2, 2])
test_specificity <- test_confusion_matrix[1, 1] / (test_confusion_matrix[1, 1] + test_confusi
on_matrix[1, 2])
test_youden_index <- test_sensitivity + test_specificity - 1
print(paste("Testing Youden's Index:", test_youden_index))
```

```
##          Actual
## Predicted   N   Y
##         0 575  57
##         1  55  63
## [1] "Testing Youden's Index: 0.443708431667024"
```

Comment: The model's performance, as indicated by the Youden's Index, shows moderate effectiveness in both training and testing datasets. The Youden's Index for the training data is 0.4498, while for the testing data, it is 0.4437, suggesting that the model performs similarly on both sets. The confusion matrix for the training data reveals a high specificity (0.927), meaning the model does well at identifying players who are not managed by a manager, but it struggles with a relatively low sensitivity (0.442), missing many true positives. The testing data shows a slight drop in specificity (0.912) and a slight improvement in sensitivity (0.525), indicating that while the model generalizes well to new data, it still underperforms in identifying positive cases. Overall, the model has a moderate ability to classify the negative class, but it misses many positives, suggesting the need for improvement. Adjusting the cutoff threshold, incorporating more features, or experimenting with different classification models could enhance performance, particularly in detecting positives.

e. Sensitivity+Specificity by lgID

```
# Add predictions to test data
test_data <- test_data %>%
  mutate(Predicted = test_preds)

# Compute sensitivity and specificity for each lgID
metrics_by_lgID <- test_data %>%
  group_by(lgID) %>%
  summarise(
    Sensitivity = sum(Predicted == 1 & plyrMgr == 1) / sum(plyrMgr == 1),
    Specificity = sum(Predicted == 0 & plyrMgr == 0) / sum(plyrMgr == 0),
    Sum = Sensitivity + Specificity
  )

# Plot bar chart
ggplot(metrics_by_lgID, aes(x = lgID, y = Sum, fill = lgID)) +
  geom_bar(stat = "identity") +
  labs(title = "Sum of Sensitivity and Specificity by lgID", x = "lgID", y = "Sensitivity + S
pecificity") +
  theme_minimal()
```



```
# Comment: Interpret variations in model performance across different leagues.
```

f. Adding win_pct to the Model

```
logit_model_extended <- glm(plyrMgr ~ yearID + win_pct, data = df_managers, family = "binomia
l")

# Compare the models
summary(logit_model)
summary(logit_model_extended)

# Compare AIC values
AIC(logit_model, logit_model_extended)
```

```
##
## Call:
## glm(formula = plyrMgr ~ yearID, family = "binomial", data = df_managers)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 88.604237   3.412071   25.97   <2e-16 ***
## yearID      -0.046611   0.001779  -26.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3442.4  on 3748  degrees of freedom
## Residual deviance: 2127.7  on 3747  degrees of freedom
## AIC: 2131.7
##
## Number of Fisher Scoring iterations: 6
##
##
## Call:
## glm(formula = plyrMgr ~ yearID + win_pct, family = "binomial",
##     data = df_managers)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 88.615550   3.419632  25.914   <2e-16 ***
## yearID      -0.046622   0.001793 -26.009   <2e-16 ***
## win_pct      0.020262   0.395959   0.051    0.959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3442.4  on 3748  degrees of freedom
## Residual deviance: 2127.7  on 3746  degrees of freedom
## AIC: 2133.7
##
## Number of Fisher Scoring iterations: 6
```

	df <dbl>	AIC <dbl>
logit_model	2	2131.660
logit_model_extended	3	2133.658

2 rows

The logit_model is preferred over the logit_model_extended because it has a slightly lower AIC (2131.660 vs. 2133.658), indicating a better balance between fit and complexity. A lower AIC suggests that logit_model offers a more parsimonious fit without unnecessary complexity, making it the better model to avoid overfitting. Therefore, logit_model is the preferred choice.

3. Poisson Regression

a. Create df_pitchers

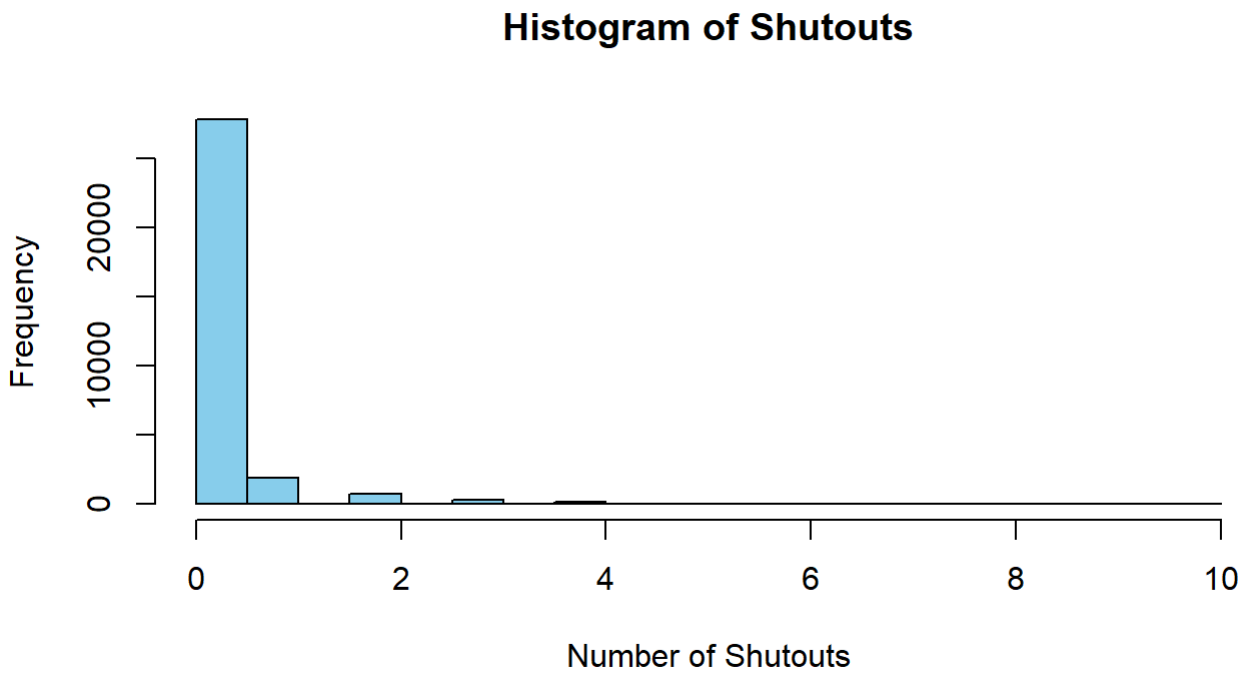
```
library(Lahman)
library(dplyr)

# i. Filter Pitching dataset for pitchers who faced at least 1 batter
df_pitchers <- Pitching %>%
  filter(BFP > 0)

# ii. Add "innings" variable
df_pitchers <- df_pitchers %>%
  mutate(innings = IPouts / 3)

# iii. Add variables from the People dataset
df_pitchers <- df_pitchers %>%
  left_join(select(People, playerID, weight, height, throws), by = "playerID")

# iv. Remove incomplete cases
df_pitchers <- na.omit(df_pitchers)

# View dataset
head(df_pitchers)
```

	playerID <chr>	yearID <int>	stint <int>	teamID <fct>	lgID <fct>	W <int>	L <int>	G <int>	GS <int>
1	aardsda01	2004	1	SFN	NL	1	0	11	0
2	aardsda01	2006	1	CHN	NL	3	0	45	0
3	aardsda01	2007	1	CHA	AL	2	1	25	0
4	aardsda01	2008	1	BOS	AL	4	2	47	0
5	aardsda01	2009	1	SEA	AL	3	6	73	0
6	aardsda01	2010	1	SEA	AL	0	6	53	0

6 rows | 1-10 of 35 columns

b. Histogram of shutouts

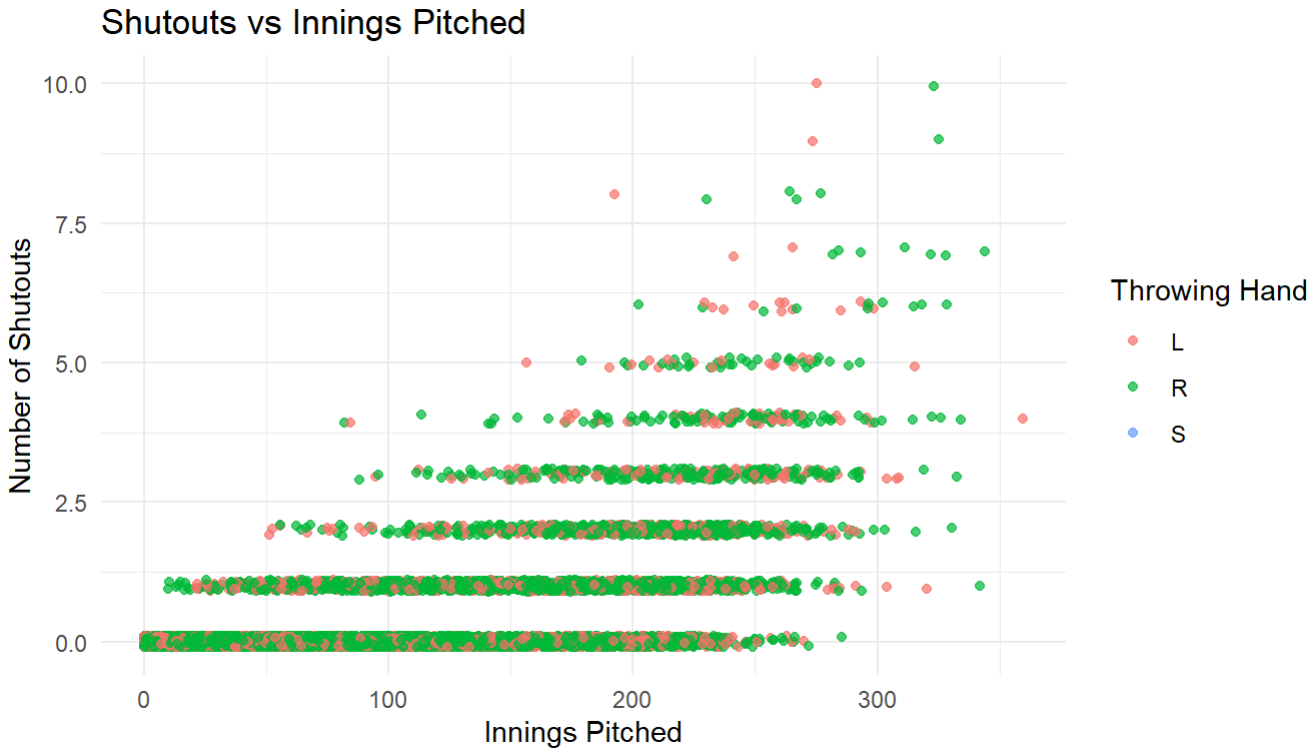
```
# Plot a histogram of the number of shutouts
hist(df_pitchers$SHO, breaks = 30, col = "skyblue", main = "Histogram of Shutouts", xlab = "N
umber of Shutouts")
```



Poisson models are suitable for count data like shutouts because: - Shutouts are non-negative integer values. - The data represents rare events, which aligns with the Poisson distribution assumptions.

c. Graph of shutouts vs innings pitched

```
library(ggplot2)

# Plot shutouts as a function of innings pitched, colored by throwing hand
ggplot(df_pitchers, aes(x = innings, y = SHO, color = throws)) +
  geom_jitter(width = 0, height = 0.1, alpha = 0.7) +
  labs(title = "Shutouts vs Innings Pitched",
       x = "Innings Pitched",
       y = "Number of Shutouts",
       color = "Throwing Hand") +
  theme_minimal()
```



Comment: The graph shows a positive relationship between innings pitched and shutouts. Left-handed and right-handed pitchers are distributed similarly, with slightly more variation among right-handed pitchers.

d. Multiple Poisson regression model

```
# Fit a Poisson regression model
poisson_mod1 <- glm(SHO ~ innings + weight + height + throws,
                    data = df_pitchers,
                    family = "poisson")

# Model summary
summary(poisson_mod1)

# Analysis of variance for p-values
anova_poisson <- anova(poisson_mod1, test = "Chisq")
anova_poisson
```

```
##
## Call:
## glm(formula = SHO ~ innings + weight + height + throws, family = "poisson",
##     data = df_pitchers)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.070e+00  5.157e-01 -13.709  < 2e-16 ***
## innings      2.053e-02  1.861e-04 110.302  < 2e-16 ***
## weight      -9.469e-03  8.155e-04 -11.612  < 2e-16 ***
## height       6.254e-02  7.892e-03   7.925 2.28e-15 ***
## throwsR     -1.022e-01  2.970e-02  -3.442 0.000577 ***
## throwsS     -8.226e+00  1.155e+02  -0.071 0.943235
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 26122  on 31010  degrees of freedom
## Residual deviance: 10607  on 31005  degrees of freedom
## AIC: 18025
##
## Number of Fisher Scoring iterations: 10
```

	Df <int>	Deviance <dbl>	Resid. Df <int>	Resid. Dev <dbl>	Pr(>Chi) <dbl>
NULL	*NA*	*NA*	31010	26122.08	*NA*
innings	1	15359.94015	31009	10762.14	0.000000e+00
weight	1	85.55727	31008	10676.58	2.250877e-20
height	1	57.47684	31007	10619.11	3.419818e-14
throws	2	11.89104	31005	10607.21	2.617537e-03

5 rows

Interpretation The Poisson regression model predicts shutouts (SHO) based on innings, weight, height, and throwing hand. **Innings** is the strongest predictor, with each additional inning increasing expected shutouts by 2.1% ($p < 2e-16$). **Weight** has a negative impact, reducing expected shutouts by 0.94% per unit increase ($p < 2e-16$), while **height** increases expected shutouts by 6.4% per unit ($p < 2e-16$). Compared to left-handed pitchers, being **right-handed** slightly decreases shutouts by 10% ($p = 0.0006$), but **switch-handedness** has no significant effect. The model fits well, significantly reducing deviance from 26122 to 10607, with an AIC of 18025, indicating strong explanatory power.

e. Poisson model with random effect (poisson_mod2)

```
# Rescaling variables
df_pitchers <- df_pitchers %>%
  mutate(across(c(innings, weight, height), scale))

# Remove sparse levels or rare categories, if needed
df_pitchers <- df_pitchers %>%
  filter(throws != "S")

# Fit the model
poisson_mod2 <- glmer(SHO ~ innings + weight + height + throws + (1 | teamID),
                      data = df_pitchers,
                      family = "poisson",
                      control = glmerControl(optimizer = "bobyqa"))

# Check the model summary
summary(poisson_mod2)
```

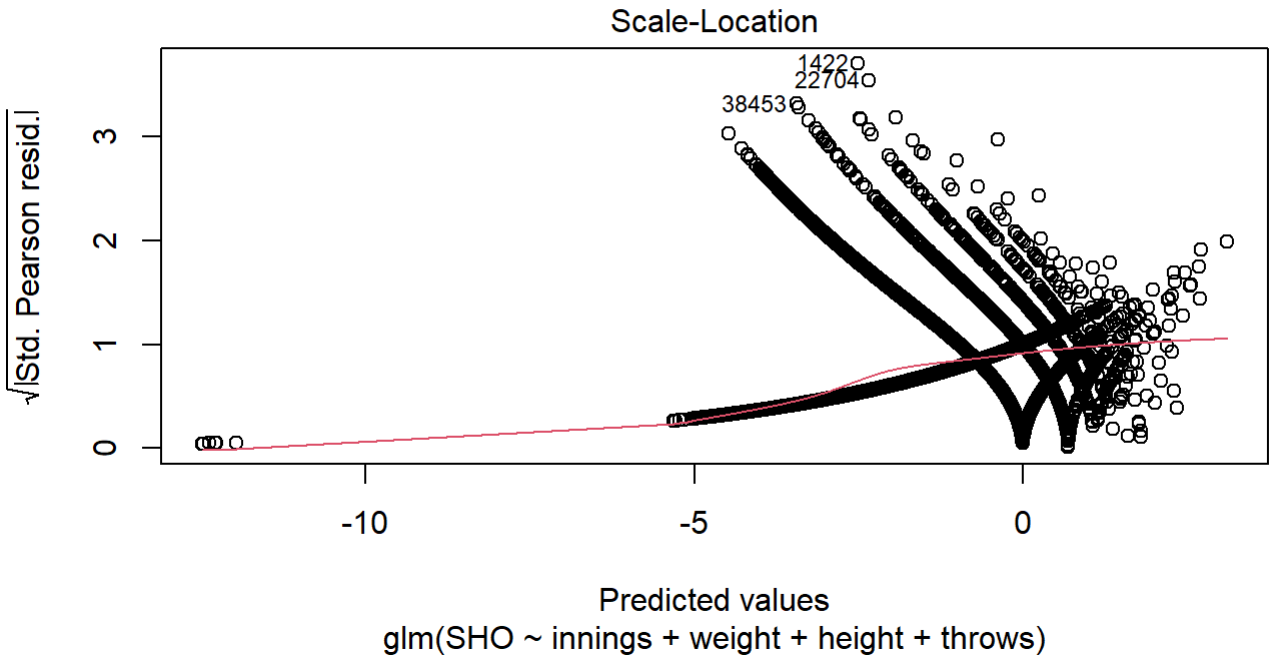
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: SHO ~ innings + weight + height + throws + (1 | teamID)
##    Data: df_pitchers
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##  17910.6  17960.6  -8949.3  17898.6    30999
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2044 -0.2237 -0.1505 -0.1164 11.7961
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  teamID (Intercept) 0.04886  0.221
## Number of obs: 31005, groups:  teamID, 35
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.06162    0.05196 -58.917  < 2e-16 ***
## innings      1.32384    0.01232 107.433  < 2e-16 ***
## weight      -0.19895    0.01858 -10.708  < 2e-16 ***
## height       0.12987    0.01718   7.559 4.07e-14 ***
## throwsR     -0.11827    0.03024  -3.911 9.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr) innngs weight height
## innings -0.445
## weight   0.020  0.136
## height   0.003 -0.077 -0.544
## throwsR -0.392  0.001  0.012 -0.119
```

Interpretation From the model results, it appears that teamID (the random effect) does not play a very significant role as a predictor. Here is why:

1. Variance of the Random Effect The variance of the random effect for teamID is 0.04886, and the standard deviation is 0.221. These values are relatively small, indicating that the variation in SHO (shutouts) due to differences between teams is minimal compared to the overall variability in the data.
2. Random Effects vs Fixed Effects The fixed effects (such as innings, weight, height, and throws) have much larger z-values and highly significant p-values (all < 0.001). This indicates that these variables explain a substantial portion of the variability in SHO. The small variance of the teamID random effect suggests that the additional grouping structure provided by teamID is not contributing much explanatory power to the model.

f. Scale-location plot

```
# Scale-Location plot for poisson_mod1
plot(poisson_mod1, which = 3)
```




```
# Observation:
# The scale-location plot should show no clear pattern if model assumptions hold.
# Deviations suggest issues like heteroscedasticity.
```

Interpretation

g. Effects of throwing hand, height, and weight

```
# Ensure 'throws' is a factor with the correct levels
df_pitchers$throws <- factor(df_pitchers$throws, levels = c("R", "L"))

# Refit the Poisson model
poisson_mod1 <- glmer(SHO ~ innings + weight + height + throws + (1 | teamID),
                      data = df_pitchers,
                      family = "poisson",
                      control = glmerControl(optimizer = "bobyqa"))

# Extract coefficients
coef_summary <- summary(poisson_mod1)$coefficients

# Check the row names of coefficients
rownames(coef_summary)

# Calculate the effects
exp_coef_throws <- exp(coef_summary["throwsL", "Estimate"]) # Adjust if necessary
exp_coef_height <- exp(coef_summary["height", "Estimate"])
exp_coef_weight <- exp(coef_summary["weight", "Estimate"])

# Output the effects
list(
  "Effect of Left Handed" = exp_coef_throws,
  "Effect of Height" = exp_coef_height,
  "Effect of Weight" = exp_coef_weight
)
```

```
## [1] "(Intercept)" "innings"     "weight"      "height"      "throwsL"
## $`Effect of Left Handed`
## [1] 1.125546
##
## $`Effect of Height`
## [1] 1.138685
##
## $`Effect of Weight`
## [1] 0.8195916
```

The results indicate that left-handed pitchers, on average, pitch 1.13 times more shutouts than right-handed pitchers, all other factors being equal. This suggests that left-handed pitchers may have a slight advantage due to their relative scarcity or the challenges batters face when competing against them. Additionally, taller players tend to pitch more shutouts, as the effect of height shows an increase of 1.14 times for every unit increase in height (likely measured in inches). This advantage could be due to the biomechanical benefits taller pitchers possess, such as generating more power or delivering pitches at more challenging angles. Conversely, heavier players pitch fewer shutouts, with the results showing a decrease of 0.82 times for every unit increase in weight (likely in pounds). This finding implies that lighter players might have better stamina or agility, contributing to their ability to pitch more shutouts. These results highlight how physical characteristics such as handedness, height, and weight can influence pitching performance.