Leveraging Launch Angle: Bayesian vs. Frequentist Approaches in Modelling Baseball Statistics

Matthew Ocampo

2025-04-19

https://github.com/mattoc32/STAT447FinalProject.git

Introduction and Problem Formulation

In recent years, launch angle has become a metric of interest in the game of baseball, revolutionizing how hitters and analysts understand offensive performance. What was once seen as a subtle byproduct of swing mechanics, launch angle is now a key driver behind trends in home runs, slugging percentages and the evolving "fly ball revolution" (Sheinen, 2017). As data becomes more central to player development and scouting, providing accurate models to depict the relationship between launch angle and hitting outcomes has become increasingly crucial. This project aims to compare Bayesian and frequentist approaches in predicting player performance statistics using launch angle as a primary predictor. Beyond simple comparisons, the Bayesian framework is extended using domain-informed priors to test whether such enhancements can yield more accurate or robust predictions. Model diagnostics will be run to determine the effectiveness and calibration of such Bayesian approaches.

Literature Review

Launch angle has reshaped how teams train hitters and transformed the way analysts evaluate hitting performance. Sheinin (2017) makes reference to the "launch angle revolution", explaining how hitters have changed their swings to elevate the ball, which has become a desired property of baseball physics in today's game. Such changes have led to shifts in how players approach the art of hitting, more specifically with an increased effort to get the ball in the air. The concept of launch angle has gained popularity in baseball analytics due to the high offensive returns it generates. Defined as the vertical angle at which the ball leaves the bat according to Major League Baseball (2025), launch angle has been shown to correlate with power-hitting metrics. These include measures like home runs and slugging percentage, which is defined as the number of bases a batter records per at-bat (MLB,2025). More home runs thus translates to higher slugging percentages.

$$\text{SLG} = \frac{1 \times 1\text{B} + 2 \times 2\text{B} + 3 \times 3\text{B} + 4 \times \text{HR}}{\text{AB}}$$

Equation 1. Slugging Percentage Formula

A study by Chen (2022) analyzed 132 qualified hitters in the 2021 MLB season and found that players with average launch angles between 15° and 20° consistently had higher slugging percentages. Conversely, extreme launch angles (below 15°) were associated with decreased overall hitting performance. The physics of hitting has been a topic of academic research, as analysts have made notion to an optimal launch angle for ball trajectory. Nathan (n.d.), explores the concept of a "peak launch angle" for maximizing batted ball distance. His findings build upon earlier research into optimal bat swing parameters, showing that undercutting the ball and generating appropriate backspin are key for maximizing trajectory. Furthermore, a similar study by Sawicki, Hubbard and Barnes (2003) attempt to derive optimum swing parameters that generate maximum distance, a desirable quality for generating more home runs.

In baseball analytics, Bayesian methods are gaining popularity for their ability to incorporate prior information and model uncertainty. A recent study by Albert (2023) incorporated Bayesian methods to determine metrics like hit probabilities using informative priors, prior predictive checks and updating beliefs based on observed data. This study incorporates similar methods to compare to frequentist methods in finding optimal models to predict slugging percentage.

Data

The selected dataset is composed of Major League Baseball player data from 2015 to 2024, obtained via Baseball Savant (see references for data source and head of the dataset). Player's are given a specific player ID for filtering, while the data consists of hitting statistics and metrics.

```
baseball_data <- read.csv(
  "/Users/matthewocampo/desktop/STAT 447/STAT447FinalProject/BaseballStatsFull.csv")
library(readr)
library(dplyr)
library(ggplot2)
library(car)
library(rstan)
library(knitr)
library(tidyr)
```

Analysis: Frequentist OLS

We apply a simple OLS regression model to predict slugging percentage using a handful of independent predictors. Along with launch angle, we incorporate barrel rate and exit velocity. A "barrel", according to Clegg (2022), is defined as a batted ball with optimal exit velocity and launch angle. Clegg also notions how optimality of these two metrics translates to higher batting average and slugging percentage, demonstrating their importance in predictions.

```
set.seed(42)
train_index <- sample(1:nrow(baseball_data), 0.7*nrow(baseball_data))
train_data <- baseball_data[train_index, ]
test_data  <- baseball_data[-train_index, ]

model_slg <- lm(slg_percent ~
                  launch_angle_avg + exit_velocity_avg + barrel_batted_rate + batting_avg, train_data)
#See Appendix for model
```

```
vif(model_slg)
```

```
##   launch_angle_avg  exit_velocity_avg barrel_batted_rate        batting_avg
##           1.366745           2.734478           3.061611           1.124588
```

```
test_data$predicted_slg_OLS <- predict(model_slg, newdata = test_data)

rmse_OLS <- sqrt(mean((test_data$slg_percent - test_data$predicted_slg_OLS)^2))
```

All VIF values are less than 10, indicating minimal multicollinearity. See appendix for more info on the model.

Analysis: Bayesian Methods (Informative Priors)

Informative priors will now be used in creating a Bayesian linear regression model. We incorporate the same train and test split used in the OLS analysis. Models are created using the STAN model framework and sampled using an MCMC sampler (see appendix for model construction).

```r
set.seed(42)
stan_data <- list(
  N = nrow(train_data),
  x_launch = train_data$launch_angle_avg,
  x_batting_avg = train_data$batting_avg,
  x_barrel = train_data$barrel_batted_rate,
  x_exit_velo = train_data$exit_velocity_avg,
  y = train_data$slg_percent
)

informative_prior_model <- stan_model("/Users/matthewocampo/Desktop/STAT 447/slg.stan")

informative_prior_fit <- sampling(
  informative_prior_model,
  data = stan_data,
  iter = 2000,
  chains = 1,
  seed = 42
)

posterior_1 <- rstan::extract(informative_prior_fit)
mean_params_1 <- sapply(posterior_1[c("intercept", "beta_launch", "beta_avg", "beta_barrel",
                             "beta_exit_velo")], mean)

test_data$predicted_slg_stan1 <- with(test_data,
                           mean_params_1["intercept"] +
                             mean_params_1["beta_launch"] * launch_angle_avg +
                             mean_params_1["beta_avg"] * batting_avg +
                             mean_params_1["beta_barrel"] * barrel_batted_rate +
                             mean_params_1["beta_exit_velo"] * exit_velocity_avg
)

rmse_stan1 <- sqrt(mean((test_data$slg_percent - test_data$predicted_slg_stan1)^2))
```

Analysis: Bayesian Methods (More Informative Launch Angle Prior)

The second Bayesian model uses a more informative prior for the launch angle variable. The prior distribution now uses a higher mean and lower standard deviation for its normal distribution (see appendix).

```r
informative_prior_model_2 <- stan_model("/Users/matthewocampo/Desktop/STAT 447/slg_2.stan")

informative_prior_fit_2 <- sampling(
  informative_prior_model_2,
  data = stan_data,
  iter = 2000,
  chains = 1,
  seed = 42
)

posterior_2 <- rstan::extract(informative_prior_fit_2)
mean_params_2 <- sapply(posterior_2[c("intercept", "beta_launch", "beta_avg", "beta_barrel",
                                    "beta_exit_velo")], mean)

test_data$predicted_slg_stan2 <- with(test_data,
```

```r
                                          mean_params_2["intercept"] +
                                            mean_params_2["beta_launch"] * launch_angle_avg +
                                            mean_params_2["beta_avg"] * batting_avg +
                                            mean_params_2["beta_barrel"] * barrel_batted_rate +
                                            mean_params_2["beta_exit_velo"] * exit_velocity_avg
)
rmse_stan2 <- sqrt(mean((test_data$slg_percent - test_data$predicted_slg_stan2)^2))
```

Comparisons of Methods

```r
rmse_table <- data.frame(
  Model = c("OLS Regression", "Bayesian Stan (Initial)", "Bayesian Stan (Improved)"),
  RMSE = c(rmse_OLS, rmse_stan1, rmse_stan2)
)

kable(rmse_table, format = "markdown", caption = "Comparison of RMSE Values")
```

Table 1: Comparison of RMSE Values

Model	RMSE
OLS Regression	0.0271071
Bayesian Stan (Initial)	0.0270621
Bayesian Stan (Improved)	0.0297714

The initial Bayesian model is the most effective in predictions, as per the lowest root mean squared error. Hence, the increased influence of launch angle in the second Bayesian model decreases predictive accuracy compared to the first Bayesian model. Calibration of the first Bayesian model is located in the appendix. This includes an analysis of predictive posterior checks on the test data, residual plots and 95% credible interval calibration on the training data (most notably, actual coverage is 94.5%). Calibration is also verified through the testing data via posterior predictive checks and analysis of prediction residuals.

Conclusion

This study aimed to produce predictive models of slugging percentage with a focus on launch angle as a main predictor. OLS regression is implemented first, followed by two Bayesian regression models that aim to improve predictive power. The first Bayesian model proves to be the most effective, as per the lowest RMSE. Subsequent calibration checks are implemented to ensure a well-specified model in the appendix.

While this study was able to produce a model representing considerable predictive accuracy, there are still key limitations. There are many external factors that influence hitting outcomes. Many of these consider contextual elements, such as quality of pitcher competition, stadium effects (weather, field dimensions) and batter handedness. Omission of such factors can lead to incomplete models, leading to possible biased predictions. It is important to consider that while relationships appear causal, they are rather statistical. It cannot be strictly stated that every player should increase launch angle to increase slugging percentage and player performance. As the game of baseball evolves through the analysis of new statistics and metrics, it is important to consider how prior knowledge can reflect accuracy of interpretations and predictions of player success. With the emergence of new trends among hitters (ex. increasing usage of the torpedo bat), future studies can continue to consider how player success is affected and defined by new concepts in the modern game of baseball.

References

Albert, Jim. "Bayesball: Bayesian Thinking in Baseball," July 15,2023, https://bayesball.github.io/BLOG/bayes.html

Chen, Steven Lu. "Launch Angle: How Important Is It to Batting Success?," *Bruins Sports Analytics*, March 9, 2022, https://www.bruinsportsanalytics.com/post/launch-angle

Clegg, Chris. "Statcast 101: Barrels, Launch Angle, and Sweet Spot Percentage," *FantraxHQ* February 2, 2022, https://fantraxhq.com/statcast-101-barrel-rates-launch-angle/.

Major League Baseball. "Custom Leaderboard." *Baseball Savant*, n.d., https://baseballsavant.mlb.com/leaderboard/custom?year=2024%2C2023%2C2022%2C2021%2C2020%2C2019%2C2018%2C2017%2C2016%2C2015&type=batter&filter=&min=q&selections=pa%2Ck_percent%2Cbb_percent%2Cwoba%2Cxwoba%2Csweet_spot_percent%2Cbarrel_batted_rate%2Chard_hit_percent%2Cavg_best_speed%2Cavg_hyper_speed%2Cwhiff_percent%2Cswing_percent&chart=false&x=pa&y=pa&r=no&chartType=beeswarm&sort=xwoba&sortDir=desc

Major League Baseball. "Launch Angle," *MLB Glossary: Statcast*, n.d., https://www.mlb.com/glossary/statcast/launch-angle

Major League Baseball. "Slugging Percentage." *MLB Glossary: Standard Stats*, n.d., https://www.mlb.com/glossary/standard-stats/slugging-percentage

Nathan, Alan M. "The Physics of Baseball: Batting and Swing Dynamics," *University of Illinois*, n.d., https://baseball.physics.illinois.edu/swing.html

Sawicki, Gregory S., Mont Hubbard and William J. Stronge. "Characterizing the Performance of Baseball Bats." American Journal of Physics 71 no.11 (2003): 1152–1162. doi: https://baseball.physics.illinois.edu/AJP-Nov03.pdf

Sheinen, Dave. "MLB's Launch Angle Revolution Is Completely Changing Baseball," *Washington Post*, June 1, 2017, https://www.washingtonpost.com/graphics/sports/mlb-launch-angles-story/

Appendix

Dataset

```r
head(baseball_data)
```

```
##   last_name..first_name player_id year  pa home_run k_percent bb_percent
## 1         Hunter, Torii    116338 2015 567       22      18.5        6.2
## 2          Ortiz, David    120074 2015 614       37      15.5       12.5
## 3       Rodriguez, Alex    121347 2015 620       33      23.4       13.5
## 4       Ramirez, Aramis    133380 2015 516       17      13.2        6.0
## 5        Beltré, Adrian    134181 2015 619       18      10.5        6.6
## 6       Beltrán, Carlos    136860 2015 531       19      16.0        8.5
##   batting_avg slg_percent   xba  xslg  woba xwoba exit_velocity_avg
## 1       0.240       0.409 0.229 0.370 0.304 0.290              88.5
## 2       0.273       0.553 0.301 0.616 0.379 0.420              93.0
## 3       0.250       0.486 0.247 0.494 0.361 0.368              91.3
## 4       0.246       0.423 0.240 0.405 0.309 0.304              87.4
## 5       0.287       0.453 0.295 0.482 0.337 0.360              89.5
## 6       0.276       0.471 0.274 0.448 0.346 0.346              90.6
##   launch_angle_avg sweet_spot_percent barrel_batted_rate poorlytopped_percent
## 1             10.8               28.5                5.0                 39.0
## 2             15.7               34.8               13.1                 27.1
## 3             12.2               31.4               10.9                 35.1
## 4             15.8               33.5                5.6                 29.4
```

```
## 5            12.6             35.7              5.5                33.7
## 6            15.6             34.1              5.8                27.8
##   poorlyweak_percent hard_hit_percent avg_best_speed avg_hyper_speed
## 1                2.9             34.9       98.56340        93.39348
## 2                1.6             49.1      102.85113        96.05306
## 3                1.0             43.9      101.38114        95.01438
## 4                3.4             34.5       97.85126        92.94476
## 5                2.2             40.4       99.25270        93.84241
## 6                1.8             41.9      100.40668        94.46574
##   whiff_percent swing_percent groundballs_percent flyballs_percent
## 1          23.1          53.4                49.4             22.6
## 2          23.2          44.7                37.6             25.6
## 3          32.0          43.9                43.6             24.9
## 4          17.9          52.9                37.6             24.5
## 5          16.8          48.1                42.4             18.4
## 6          18.1          45.4                36.8             26.6
```

OLS Regression Model

```r
summary(model_slg)
```

```
##
## Call:
## lm(formula = slg_percent ~ launch_angle_avg + exit_velocity_avg +
##     barrel_batted_rate + batting_avg, data = train_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.096238 -0.018620 -0.001476  0.017805  0.108412
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.1689198  0.0578715  -2.919   0.0036 **
## launch_angle_avg    0.0035363  0.0002389  14.800   <2e-16 ***
## exit_velocity_avg   0.0012951  0.0006728   1.925   0.0545 .
## barrel_batted_rate  0.0090556  0.0003795  23.861   <2e-16 ***
## batting_avg         1.4496918  0.0329810  43.955   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02786 on 957 degrees of freedom
## Multiple R-squared:  0.8185, Adjusted R-squared:  0.8178
## F-statistic:  1079 on 4 and 957 DF,  p-value: < 2.2e-16
```

Stan Models

Model 1:

```
data {
  int<lower=0> N;
  vector[N] x_launch;
  vector[N] x_batting_avg;
  vector[N] x_barrel;
  vector[N] x_exit_velo;
  vector[N] y;
```

```
}

parameters {
  real beta_launch;
  real beta_avg;
  real beta_barrel;
  real beta_exit_velo;
  real intercept;
  real<lower=0> sigma;
}

model {
  // Informative priors
  beta_launch ~ normal(0.01, 0.001);        // known: higher launch angle increases SLG
  beta_avg ~ normal(1.5, 0.5);              // batting avg strongly affects SLG
  beta_barrel ~ normal(0.01, 0.005);        // barrels should positively influence SLG
  beta_exit_velo ~ normal(0.005, 0.005);    // moderate positive expectation
  intercept ~ normal(0.5, 1);
  sigma ~ exponential(1);

  // Likelihood
  y ~ normal(intercept + beta_launch * x_launch +
             beta_avg * x_batting_avg +
             beta_barrel * x_barrel +
             beta_exit_velo * x_exit_velo, sigma);
}

generated quantities {
  vector[N] y_rep;

  for (n in 1:N) {
    y_rep[n] = normal_rng(
      intercept +
      beta_launch * x_launch[n] +
      beta_avg * x_batting_avg[n] +
      beta_barrel * x_barrel[n] +
      beta_exit_velo * x_exit_velo[n],
      sigma
    );
  }
}
```

Model 2 (More Informative Prior):

```
data {
  int<lower=0> N;
  vector[N] x_launch;
  vector[N] x_batting_avg;
  vector[N] x_barrel;
  vector[N] x_exit_velo;
  vector[N] y;
}

parameters {
  real beta_launch;
```

```
  real beta_avg;
  real beta_barrel;
  real beta_exit_velo;
  real intercept;
  real<lower=0> sigma;
}

model {
  // Informative priors
  beta_launch ~ normal(0.02, 0.0005);        // More informative prior
  beta_avg ~ normal(1.5, 0.5);
  beta_barrel ~ normal(0.01, 0.005);
  beta_exit_velo ~ normal(0.005, 0.005);
  intercept ~ normal(0.5, 1);
  sigma ~ exponential(1);

  // Likelihood
  y ~ normal(intercept + beta_launch * x_launch +
             beta_avg * x_batting_avg +
             beta_barrel * x_barrel +
             beta_exit_velo * x_exit_velo, sigma);
}

generated quantities {
  vector[N] y_rep;

  for (n in 1:N) {
    y_rep[n] = normal_rng(
      intercept +
      beta_launch * x_launch[n] +
      beta_avg * x_batting_avg[n] +
      beta_barrel * x_barrel[n] +
      beta_exit_velo * x_exit_velo[n],
      sigma
    );
  }
}
```

Posterior Calibration Checks

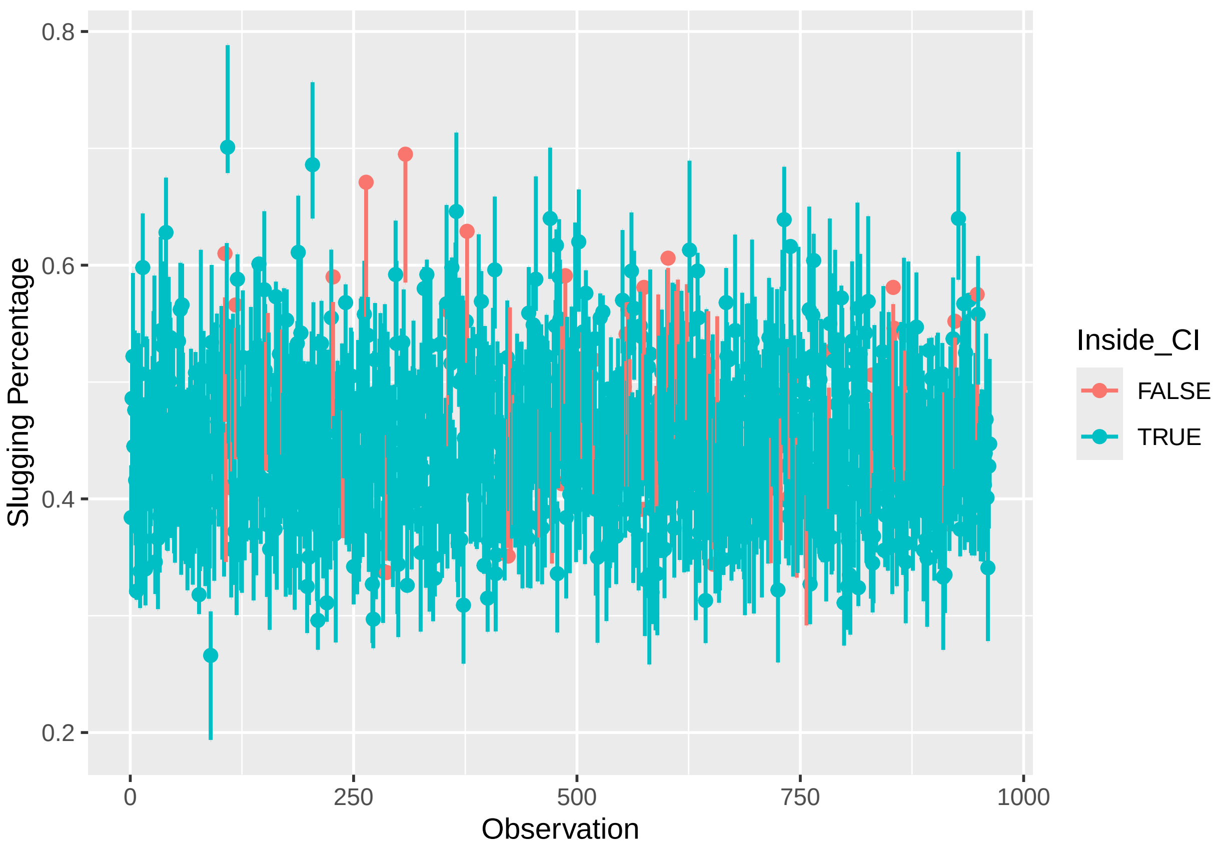
The above analysis implies the first Bayesian model is the most effective at predicting slugging percentage. While comparing predictions to actual test data is a good indicator of model calibration, further posterior predictive checks may be warranted. Specifically, it suffices to test the calibration in relation to the train and test data, which we will see below. The first method creates credible intervals from the predictions generated in the STAN model. Coverage is generated from determining if training data is within each credible interval. The desired credible interval coverage is 95%, while the actual coverage is 94%, indicating a well-specified model.

```r
y_rep <- posterior_1$y_rep
lower <- apply(y_rep, 2, quantile, 0.025)
upper <- apply(y_rep, 2, quantile, 0.975)

within_interval <- train_data$slg_percent >= lower & train_data$slg_percent <= upper
coverage <- mean(within_interval)
```

```r
cat("Empirical 95% coverage:", coverage)
```

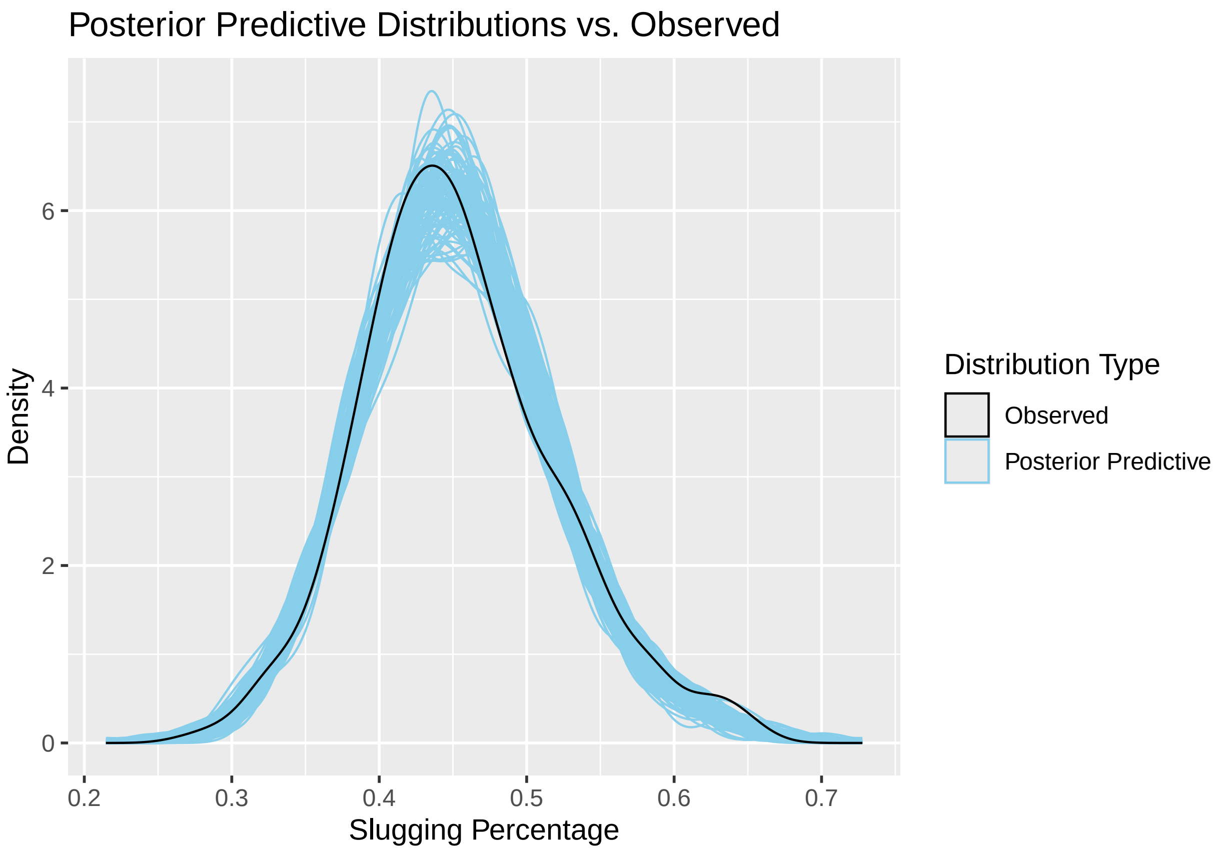
```
## Empirical 95% coverage: 0.9449064
```

```r
coverage_df <- data.frame(
  Observation = 1:length(train_data$slg_percent),
  Slugging_Percentage = train_data$slg_percent,
  Lower = apply(y_rep, 2, quantile, 0.025),
  Upper = apply(y_rep, 2, quantile, 0.975)
)

coverage_df$Inside_CI <- with(coverage_df, Slugging_Percentage >= Lower & Slugging_Percentage <= Upper)

ggplot(coverage_df, aes(x = Observation, y = Slugging_Percentage, color = Inside_CI)) +
  geom_point(size = 2) +
  geom_errorbar(aes(ymin = Lower, ymax = Upper), width = 0.2, size = 0.7) +
  scale_color_manual(values = c("FALSE" = "#F8766D", "TRUE" = "#00BFC4")) +
  labs(
    y = "Slugging Percentage",
    color = "Inside_CI"
  ) +
  theme(
    legend.position = "right"
  )
```



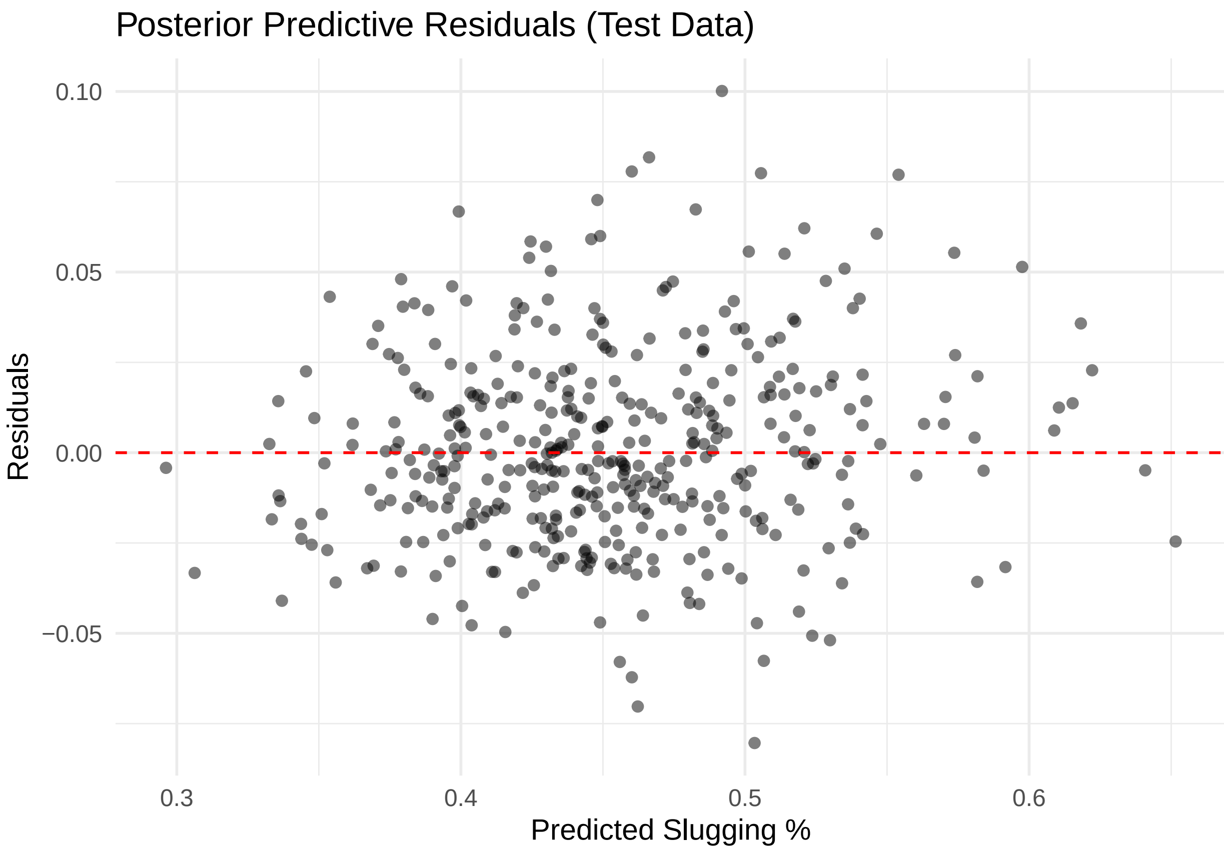
The second calibration method applies a posterior predictive check using simulated data generated by the posterior distribution (posterior draws). It can be observed that the simulated data resembles the test set.

```r
n_draws <- length(posterior_1$intercept)
N_test <- nrow(test_data)

y_rep_test <- matrix(NA, nrow = n_draws, ncol = N_test)

for (i in 1:n_draws) {
  mu <- posterior_1$intercept[i] +
    posterior_1$beta_launch[i] * test_data$launch_angle_avg +
    posterior_1$beta_avg[i] * test_data$batting_avg +
    posterior_1$beta_barrel[i] * test_data$barrel_batted_rate +
    posterior_1$beta_exit_velo[i] * test_data$exit_velocity_avg

  y_rep_test[i, ] <- rnorm(N_test, mean = mu, sd = posterior_1$sigma[i])
}

y_rep_long <- as.data.frame(y_rep_test[1:100, ]) %>%
  mutate(draw = row_number()) %>%
  pivot_longer(
    cols = -draw,
    names_to = "obs",
    values_to = "value"
  ) %>%
  mutate(group = "Posterior Predictive")

observed_df <- data.frame(
  value = test_data$slg_percent,
  group = "Observed"
)

combined_df <- bind_rows(
  y_rep_long %>% select(value, group, draw),
  observed_df %>% mutate(draw = NA)
)

ggplot(combined_df, aes(x = value, color = group, group = interaction(group, draw))) +
  geom_density(alpha = 0.4, size = 0.4) +
  scale_color_manual(values = c("Posterior Predictive" = "skyblue", "Observed" = "black")) +
  labs(x = "Slugging Percentage", y = "Density", color = "Distribution Type",
       title = "Posterior Predictive Distributions vs. Observed")
```

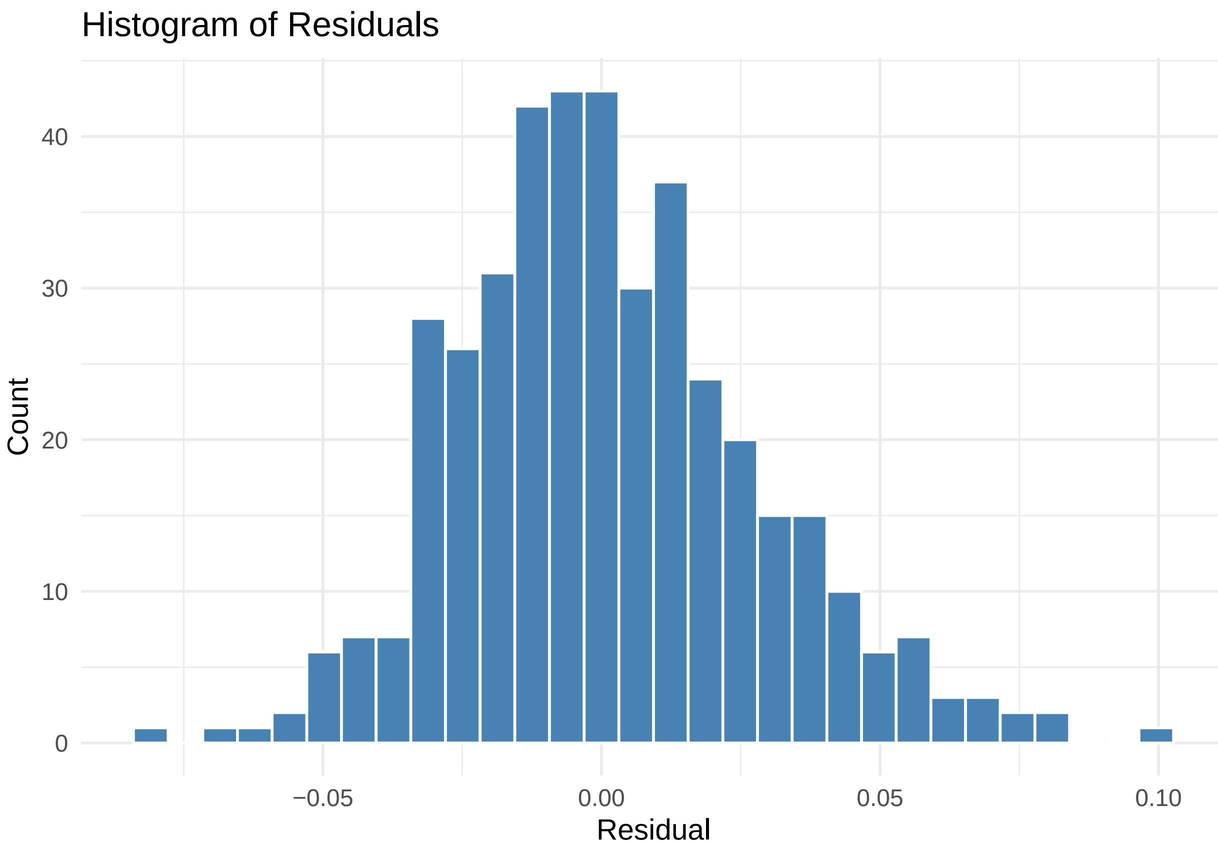


The last calibration method implemented examines the residuals of the predictions in comparison to actual slugging percentage values. We observe randomness, with no apparent trend or pattern. The histogram also replicates a normal distribution.

```r
mean_preds <- colMeans(y_rep_test)

test_data$residuals <- test_data$slg_percent - mean_preds
test_data$predicted_slg_means <- mean_preds

ggplot(test_data, aes(x = predicted_slg_means, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Posterior Predictive Residuals (Test Data)",
       x = "Predicted Slugging %", y = "Residuals") +
  theme_minimal()
```




```
ggplot(test_data, aes(x = residuals)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  labs(title = "Histogram of Residuals", x = "Residual", y = "Count") +
  theme_minimal()
```



Histogram of Residuals
40
30
20
10
0
Count
−0.05
0.00
0.05
0.10
Residual