RICE UNIVERSITY

Ballpark Biases in Baseball: A Bayesian Analysis

STAT 425

Fall 2024

INSTRUCTOR: Marina Vannucci

DUE DATE: 12/13/2024

GROUP 9

Brandon Fantine, Aditya Daga, Andrew Sun

Table of Contents

Introduction

Predicting baseball is a complicated beast. Many spend their entire careers trying to figure out the best way to determine player success, and, especially with the onset of Fantasy Leagues and online sports betting, it has become increasingly important for the average person to be accurate in their predictions. However, an element of Baseball that goes commonly overlooked is the Baseball Stadium itself. Through a Bayesian analysis, we attempt to make sense of the relationship between ballpark and player in an effort to determine the validity of including stadiums in future baseball prediction algorithms.

Stadium Dimensions

The Tampa Bay Rays have been exploring the idea of building a new baseball stadium, influenced by various economic, geographic, and team performance considerations. A key question in this process is whether certain stadium design features could be leveraged to give their players a competitive advantage. While factors such as location, fan experience, and cost are typically paramount, the team might also consider how structural and dimensional characteristics of a ballpark could influence player performance. Specifically, if the Rays find that their core players historically fare better under certain

stadium conditions, they may choose to incorporate those elements into their new stadium to maximize on-field success.

In this investigation, we focus on three players from the Rays who each accumulated over 350 at-bats in a given season: Yandy Diaz, Josh Lowe, and Jose Caballero. By examining how these players have performed across a variety of ballparks, we aim to identify whether there are notable stadium-related factors that correlate with improved offensive outputs. If such relationships exist, it could inform the strategic design of a new stadium.

Data Overview

To conduct this analysis, we draw upon two primary data sources:

1. Stadium Characteristics Data:
   The first dataset, sourced from a .CSV file (Clem's Baseball data), provides a comprehensive look at various Major League Baseball stadiums. It includes attributes like seating capacity, number of seating rows, deck arrangements, shading conditions, foul and fair territory measurements, fence heights at different outfield positions, and detailed outfield dimensions.
2. Player Performance Data (Statcast):
   The second dataset comes from the MLB's Statcast data, accessed via the baseballr package in R. For the specified date range (April through October 2024), we retrieved pitch-by-pitch data for the selected players. We then aggregated this data to the game level and calculated key offensive metrics, such as On-Base Plus Slugging (OPS). After computing player-specific average OPS values, we derived a performance metric (OPS_diff) representing how each player's performance in a given stadium compared to their own season-long average.

We merged these datasets—linking player performance metrics with the corresponding stadium characteristics of the venue where each game was played—we obtain a final, integrated dataset, which player performances corresponding to the stadium dimensions they played at.

Model Selection

The choice of an appropriate statistical model was guided by the characteristics of the response variable and the theoretical considerations related to stadium design factors. Initially, a Student's t-distribution model was considered due to our belief that player's performances are quite random, and the t-distribution offered a robust alternative to the normal distribution, accommodating the possibility of heavier tails and outliers. However, after examining the distribution of OPS_diff values, (see **Appendix 1**), it became apparent that a significant proportion of the data was skewed and not well-captured by symmetrical distributions like the t-distribution. To address this issue, we transformed the OPS_diff values by adding a constant offset, ensuring that all observations were strictly positive. This allowed for the use of a Gamma regression model, which is naturally defined for positive-valued outcomes and can flexibly model skewed data. The Gamma model with a log-link provided a better conceptual and statistical fit, as it connects multiplicative effects of covariates on the outcome and directly accommodates the inherently positive and asymmetric nature of the performance differences.

The priors for the Gamma regression model's coefficients were carefully chosen to reflect modest, directionally plausible effects of stadium features on player performance. Each prior followed a normal distribution, with a specified mean and precision (the inverse of variance). For example, the intercept was assigned a prior of dnorm(0, 0.1). In this case, the precision of 0.1 corresponds to a variance of 10, providing some cushion around zero but still offering a moderately centered expectation that, in the absence of any predictors, player performance differences would not be dramatically skewed.

One noteworthy prior relates to shading conditions, such as Lower Deck Shade. Here, the mean of -0.2 and precision of 9 (variance ≈ 0.111) reflect an expectation that increased shading in the lower deck has a modest but more confidently negative effect on player OPS differences. This stems from our belief that shade impacts visibility or ball trajectory in a predictable way—maybe the dimmer conditions slightly disadvantage hitters who rely on visual cues. While still not highly restrictive, this prior signals a stronger prior assumption compared to some other parameters.

Posterior Inference Process (Student's T)

The first model we ran was the Student's t-distributed likelihood with uninformative normal priors for the regression coefficients, a half-Cauchy prior for the scale parameter, and an exponential prior for the degrees of freedom. The model was implemented using JAGS with three parallel chains and an adaptation phase of 1,000 iterations to fine-tune the samplers. An additional 1,000 iterations were used as a burn-in period, ensuring that the chains converged before sampling began. Finally, 5,000 iterations were run per chain, with a thinning interval of 5 to reduce autocorrelation, resulting in 1,000 posterior draws per chain. Diagnostics (see **Appendix 2**) confirm the quality of the posterior samples. The trace plots show stable mixing across chains with no visible trends, and the autocorrelation plots reveal minimal lag dependence. Gelman-Rubin diagnostics yielded values near 1 for all parameters, confirming convergence. These diagnostics validate the robustness of the posterior inferences drawn from the Student's t model.

Posterior Inference Process (Gamma)

We evaluated a Gamma regression model with a log-link function, using a Gamma-distributed likelihood for the positively shifted OPS_diff_shifted variable. The shape parameter (α) was assigned a prior of dgamma(2, 0.5) to accommodate moderate dispersion in the data while maintaining flexibility. The model was implemented in JAGS with three parallel chains and an adaptation phase of 1,000 iterations. After an additional burn-in period of 1,000 iterations to ensure convergence, 5,000 iterations per chain were sampled, with thinning set to 5, resulting in 1,000 posterior draws per chain. Diagnostics (**Appendix 3**) confirm the robustness of the posterior inferences. Traceplots display smooth and stable mixing, with slightly better consistency and reduced noise compared to the Student's t model. Autocorrelation plots exhibit minimal lag, and Gelman-Rubin diagnostics yielded values near 1 for all parameters, confirming convergence.

Posterior Inference Results (Combined)

The Student's t-distribution model (see **Appendix 4**), provided a robust framework for analyzing stadium characteristics' effects on player performance, effectively handling potential outliers. Posterior means for regression coefficients (β_i) were generally small, with β_1 (Seating Capacity) showing a modest positive effect (mean: 0.091) and β_{13} (Left Field Distance) exhibiting a negative relationship (mean: -0.193). Standard deviations indicated moderate uncertainty, especially for parameters like β_5 (Lower Deck Shade) and β_9(Fence Height Left Field). The degrees of freedom parameter (ν) had a mean of 5.41, indicating flexibility for heavy-tailed data, and the scale parameter (σ) was stable (mean: 0.52). Overall, diagnostics confirmed good convergence and mixing, supporting the validity of the results.

The Gamma regression model (see **Appendix 5**) leveraged a log-link function to model the positively shifted OPS_diff variable. Posterior means for coefficients (β) similarly reflected small to moderate effects, with β_1 (Seating Capacity) showing a slight positive impact (mean: 0.086) and β_{13} (Left Field Distance) a stronger negative effect (mean: -0.207). Standard deviations showed stability across parameters, and variables like β_9 (Fence Height Left Field) exhibited lower uncertainty compared to the t model. The shape (α) and scale (σ) parameters, with means of 5.41 and 0.52 respectively, reinforced the Gamma model's suitability for the skewed, positive response. Diagnostics demonstrated strong convergence and mixing.

Both models identified small to moderate effects of stadium characteristics, with consistent parameter estimates for variables like β_1 and β_{13}. The Student's t model excelled in handling potential outliers and heavy tails but showed higher uncertainty for some parameters. In contrast, the Gamma model, with its informed priors, provided more stable estimates and narrower credible intervals, especially for parameters like β_9. While the t model offered flexibility with its degrees of freedom parameter (nu), the Gamma model's alignment with the data's skewed nature made it better suited for this analysis, offering improved interpretability and robustness.

Our sensitivity analysis demonstrated that the Gamma model's results were robust to changes in the priors, with posterior means for key parameters remaining consistent across original, tighter, wider, and very uninformative priors. For instance, β_{13} (Left Field Distance) consistently showed a strong negative effect, with mean estimates ranging narrowly from -0.194 to -0.207 across prior specifications. Similarly, β_9 (Fence Height Left Field) maintained a positive relationship, with slight variations in the posterior standard deviations depending on the priors' precision. While tighter priors reduced variability, resulting in narrower credible intervals, wider priors slightly increased uncertainty but did not shift the central estimates significantly. This consistency highlights the robustness of the Gamma model and reinforces the validity of its results.

Conclusions

This analysis explored the impact of stadium design features on player performance, focusing on key players from the Tampa Bay Rays. Using two modeling approaches—a Student's t-distribution model and a Gamma regression model—we identified small but notable effects of stadium dimensions, such as

Left Field Distance and Fence Height Left Field, on player OPS differences. While the Student's t model excelled in accommodating outliers and heavy-tailed data, the Gamma model proved to be a better fit for the skewed, positive nature of the performance metric, providing more stable estimates and narrower credible intervals. The sensitivity analysis further confirmed the robustness of the Gamma model, with consistent results across different prior specifications. These findings offer valuable insights into how stadium features influence player performance and can guide the Rays in designing a ballpark that strategically supports their players.

Home Field Advantage

But the Tampa Bay Rays aren't the only team in the MLB. A primary question guiding this analysis is: *Does a team's run rate improve significantly when playing at their home stadium versus away stadiums?* Home-field advantage is a widely discussed concept in sports, often linked to factors such as fan support, familiarity with the playing field, and reduced travel fatigue. However, its impact on specific metrics like run rates in baseball has not been conclusively established. This study aims to investigate whether playing at home leads to a measurable improvement in scoring outcomes for Major League Baseball teams.

The dataset, sourced from Kaggle, includes game-by-game player and team-level baseball data spanning the 2017 to 2020 seasons. It provides comprehensive details on player performance, team results, and game conditions. The key variable of interest, Team.R (total runs scored by the team), serves as the primary response variable. Predictor variables include H.A (home or away indicator) and Opponent.Strength (derived from the Moneyline odds). This dataset of roughly 240,000 rows and 70 columns offers a solid foundation for statistical modeling, allowing us to analyze how playing at home versus away and opponent strength interact to influence team run scoring rates.

Opponent strength, a one of two key predictors in this analysis, was derived from the Moneyline column, which reflects pre-game betting odds indicating a team's favorability to win. Lower Moneyline values represent stronger teams (favorites), while higher values indicate weaker teams (underdogs). To standardize this metric and make it suitable for statistical modeling, we calculated the z-score of the Moneyline variable using the formula:

$$Opponent.Strength = \frac{Moneyline - mean(Moneyline)}{sd(Moneyline)} \quad (1)$$

This standardization centers the variable at 0 and scales it by its standard deviation, making it easier to interpret and incorporate into our model. The standardized Opponent.Strength variable allows us to quantify how a team's relative strength, before the game, impacts their ability to score runs.

Model Selection

Our primary response variable, Team.R, represents the total runs scored by the batter's team in a game. To evaluate factors influencing team scoring, we focused on two key predictors: H.A (indicating whether the game was played at home or away) and Opponent.Strength (a standardized metric derived from the Moneyline to measure team favorability). Negative values of Opponent.Strength represent stronger opponents, while positive values represent weaker ones. These predictors are central to assessing the potential effects of playing at home and facing stronger or weaker opponents on a team's scoring performance.

Given that Team.R is count data, we initially considered using the Poisson distribution for modeling. However, baseball data often exhibit overdispersion, where the variance in runs scored exceeds the mean. The Poisson distribution assumes equal mean and variance, which makes it unsuitable for this

type of data. To address overdispersion, we opted for the Negative Binomial distribution, which introduces a dispersion parameter (ϕ) to better accommodate variability.

$$Team.R_i \sim NegBinom(\mu_i, \phi) \qquad (2)$$

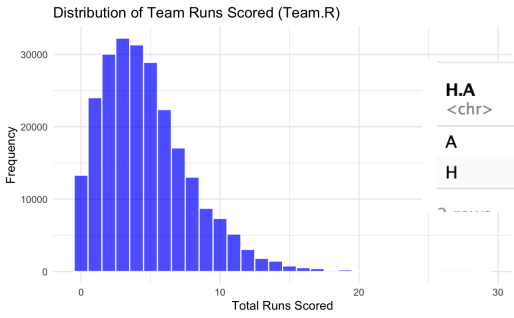
This choice ensures a more accurate representation of the data and accounts for the complex factors affecting team scoring. The expected number of runs, denoted as μ,is modeled on the log scale to ensure positivity and capture multiplicative relationships between predictors and the response variable. The formula for the log of the expected runs is:

$$log(\mu_i) = \beta_0 + \beta_1 \cdot H.A_i + \beta_2 + Opponent.Strength_i \qquad (3)$$

Here, β0 represents the baseline log-run rate for away games, β1 captures the additional effect of playing at home (expected to be positive for home-field advantage), and β2 measures the effect of opponent strength (expected to be negative, as stronger opponents suppress scoring). The inclusion of an overdispersion parameter ensures flexibility in modeling variability beyond what the predictors alone can explain. This structure allowed us to build a robust model.

Verifying Model Assumptions

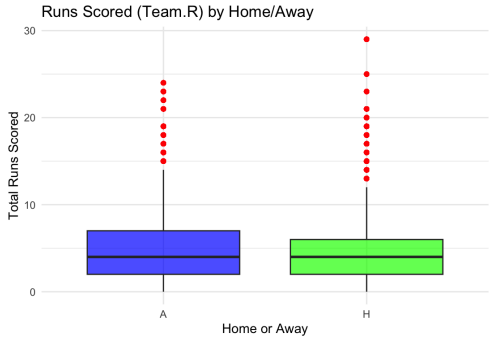
To ensure the appropriateness of our chosen Negative Binomial model, we began by examining the distribution of the response variable, Team.R. A histogram of Team.R revealed a right-skewed distribution, characteristic of count data, with most observations concentrated at lower run values and a long tail extending to higher counts. This aligns well with the assumptions of a Negative Binomial model, which accommodates such skewness and variability better than the Poisson distribution. Additionally, a summary of the mean and variance of Team.R confirmed overdispersion: the variance significantly exceeded the mean for both home and away games. This reinforced the need for the Negative Binomial model, which incorporates an overdispersion parameter to handle this variability. Both the histogram and the mean-variance summary can be seen below in figure 2.1.



H.A <chr>	Mean <dbl>	Variance <dbl>
A	4.659726	10.80537
H	4.743746	10.53404

Figure 2.1

Further validation came from visualizing the relationship between predictors and the response variable. A box-and-whisker plot of Team.R grouped by H.A (home vs. away) suggested a very modest difference in median runs scored, with home teams scoring slightly higher on average. This preliminary evidence supported the inclusion of H.A as a predictor in the model. Additionally, a scatter plot of Team.R against Opponent.Strength revealed a negative trend, indicating that stronger opponents (lower Opponent.Strength) tended to suppress scoring. This relationship justified the inclusion of Opponent.Strength as a meaningful predictor in the model. Both the box-and-whiskers plot and the scatter plot can be seen below in figures 2.2 and 2.3.



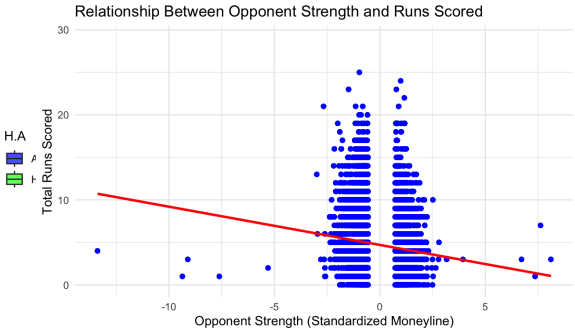


Figure 2.2 (left), Figure 2.3 (right)

These exploratory analyses provided strong evidence that our model assumptions were reasonable and worth investigating further. The alignment between the observed data patterns and the theoretical capabilities of the Negative Binomial distribution validated our choice. Moreover, the insights from the visualizations confirmed that our key predictors, H.A and Opponent.Strength, were relevant to explaining variability in team runs. These steps ensured that our model was grounded in the data's underlying structure and informed by meaningful relationships between predictors and outcomes.

Selecting Priors

We selected weakly informed priors to balance flexibility and structure, allowing the data to dominate and refine the estimates while grounding the parameters in plausible ranges based on general intuition and prior research. The prior for the baseline log-run rate, β_0, was chosen as $N(0,2^2)$ reflecting a mean of 0 and a standard deviation of 2. This wide prior acknowledges the potential variability in scoring tendencies across teams, independent of specific factors like home-field advantage or opponent strength. By selecting a standard deviation of 2, we allowed for greater uncertainty, enabling the model to explore a broad range of plausible baseline values. This choice balances flexibility with enough structure to anchor the model around realistic values for runs scored, aligning with general expectations in the data.

For the home-field advantage effect, β_1, we used a prior of $N(0,0.5^2)$, centering the distribution at 0 with a smaller standard deviation of 0.5. This reflects the understanding that the home-field advantage,

while generally positive, is expected to be modest in magnitude. The narrower standard deviation constrains the model to explore only a relatively small range of plausible values for B1, consistent with prior research on home-field effects in sports. The choice of a prior mean of 0 captures uncertainty about the exact size of this effect, allowing the data to refine the estimate while ensuring that it reflects realistic bounds.

The prior for β_2, the effect of opponent strength, was selected as N(0,1^2) with a mean of 0 and a standard deviation of 1. This choice balances uncertainty about the magnitude of the suppression effect while incorporating prior knowledge that stronger opponents typically suppress scoring. The prior reflects more variability than β_1 but less than β_0, given that opponent strength is more dynamic across games.For the overdispersion parameter (ϕ), we chose a Gamma(2, 0.5) distribution, which is strictly positive and flexible, capturing variability in team run counts. This prior was informed by studies on baseball run data, centering the distribution on plausible positive values while allowing moderate uncertainty. Together, these choices of priors balance flexibility, informed intuition, and the ability for data to refine estimates.

Posterior Inference

To conduct posterior inference, we began by initializing the parameters for our Bayesian model. Initial values for each parameter (β_0, β_1, β_2, and ϕ) were randomly drawn from their respective prior distributions to ensure proper sampling and exploration of the parameter space. The JAGS model was set up to use three independent Markov chains to enhance robustness and facilitate diagnostic testing. We implemented a burn-in phase of 500 iterations, allowing the chains to stabilize and converge toward the posterior distribution. After the burn-in, we sampled 3,000 iterations while thinning every five steps to reduce autocorrelation between consecutive samples, ensuring an effective sample size.

To evaluate convergence, we utilized multiple diagnostic methods. First, we visualized trace plots for each parameter to inspect the behavior of the Markov chains over time. The trace plots showed consistent mixing and overlapping across chains, a strong indication that the chains converged to the posterior distribution. We also assessed autocorrelation using autocorrelation plots for each parameter, which showed a rapid decline in correlation as the lag increased, indicating minimal autocorrelation in the posterior samples and effective mixing of the Markov chains. Lastly, we applied the Gelman-Rubin diagnostic, which compares within-chain and between-chain variance to assess convergence (Output in figure 2.4 below). For all parameters, the potential scale reduction factor was close to 1, confirming that the chains had converged to their stationary distributions (see **Appendix 6** for trace plots and autocorrelation plots for all three chains).

```
Potential scale reduction factors:

      Point est. Upper C.I.
alpha      0.999       1.00
beta0      1.003       1.01
beta1      1.004       1.01
beta2      1.001       1.01
```

Figure 2.4

After running MCMC, the posterior results gave us solid insights into the factors influencing a team's run rate. The overdispersion parameter, with a mean of 3.75 and a credible interval from 3.71 to 3.80, confirms moderate variability in runs scored. This finding supports our choice of using the Negative Binomial model over the Poisson model, which would have been inappropriate for handling overdispersed count data. The robustness of this parameter validates the model's ability to account for variability and ensures reliable inferences about our predictors.

For the baseline log-run rate (β_0), the mean estimate of 1.57 with a very narrow credible interval demonstrates high precision and reliability in understanding run rates during away games. This serves as a solid reference point for comparing the effects of playing at home or facing different opponents. The stability of β_0 indicates that the underlying factors affecting away games are well-captured by the model, providing confidence in its estimates.

The findings for β_1, representing the home-field effect, were unexpected. Contrary to the anticipated positive home-field advantage, β_1 yielded a mean of -0.051 with a 95% credible interval consistently below zero. This suggests a slight negative effect of playing at home on run rates, challenging traditional beliefs about home-field advantage in baseball. Potential explanations include confounding factors such as lineup changes, park effects, or overlapping influences between predictors like opponent strength and home-field advantage. These results prompt further investigation to understand why home-field advantage appears diminished or reversed in our analysis.

Finally, β_2 aligned well with expectations, confirming the importance of opponent strength in suppressing run rates. Its mean of -0.106 and narrow credible interval reinforce the notion that stronger opponents significantly impact performance. While β_2 results are consistent with prior assumptions, the counterintuitive negative β_1 necessitates deeper investigation. To address this, we will perform a sensitivity analysis, varying the prior specifications for β_1. If the negative sign persists across different prior settings, it would indicate that the result is robust and not an artifact of prior assumptions, underscoring the importance of this unexpected finding.

Sensitivity Analysis

To ensure the robustness of our findings, we performed a sensitivity analysis by testing our model under different prior specifications. This allowed us to evaluate how prior assumptions influenced the posterior results and whether the data-driven conclusions were consistent across varying levels of prior information.

First, we implemented completely non-informative priors to minimize prior influence and focus solely on the data's contribution to the results. For β_0, β_1, and β_2, we specified normal distributions with a mean of 0 and an extremely large variance of 10,000, representing substantial uncertainty about the parameters. For the overdispersion parameter, ϕ, we used a uniform distribution ranging from 0.01 to 10, which allowed for a wide range of possible overdispersion levels without imposing any specific expectations.

Next, we utilized informed priors derived from the posterior estimates of our partially informed model. These priors incorporated the mean and variance from the previous analysis, narrowing the distributions around plausible values. For example, β_0 had a mean of 1.5685 and a variance derived from its posterior standard deviation, while β_1 and β_2 followed similar patterns. The overdispersion parameter ϕ remained modeled as a Gamma(2, 0.5) distribution, reflecting our verified assumption about run variability. This approach allowed us to examine whether our prior beliefs aligned with the data and assess the consistency of our findings across prior frameworks.

The sensitivity analysis revealed that the posterior means for all parameters were remarkably consistent across the different prior specifications. For example, β_1, which represents the home-field effect, consistently remained around -0.0514, and β_2, reflecting opponent strength, remained near -0.1056. This demonstrates that the data strongly influenced the posterior estimates, minimizing the impact of prior assumptions. Even with the extremely broad non-informative priors, the posterior results aligned closely with those obtained under informed priors.

Similarly, the overdispersion parameter, phi, exhibited consistency across models regardless of the prior specification. The posterior means and credible intervals for phi were nearly identical, confirming that our prior distribution choice for this parameter had minimal impact on its final estimates. This further supports the robustness of our model to handle variability in team runs and affirms the suitability of the Negative Binomial distribution (see **Appendix 7** for density plots for all posteriors).

Overall, the results of the sensitivity analysis validate the appropriateness of our partially informed priors. The consistency in the posterior distributions across vastly different priors strengthens our conclusions about β_0, β_1, and β_2. Notably, the unexpected negative β_1 persisted under all prior assumptions, reinforcing the robustness of this surprising finding. These results lend greater credibility to our analysis, as they suggest that the observed effects are data-driven and not sensitive to prior specifications.

Final Verdict

Our analysis uncovered a slight negative effect of playing at home on a team's run rate. However, this does not definitively suggest a "home-field disadvantage" in baseball. While the data and modeling indicate a minor suppression of scoring for home teams, this finding should not be interpreted as evidence that home-field advantage does not exist. Instead, it points to the complexity of factors influencing performance, where home-field effects may not directly translate to increased run scoring.

One potential source of error in our analysis stems from the derivation of "Opponent Strength" from Moneyline, which inherently factors in home-field advantage. This overlap between predictors might have unintentionally suppressed the coefficient for playing at home (β_1), leading to a negative estimate. By attributing some of the home-field advantage to opponent strength, our model may have underestimated the actual effect of playing at home on run rates.

Alternatively, we may be overestimating the influence of home-field advantage specifically on run scoring. While home teams likely benefit from factors such as crowd support, morale, and familiarity

with their park, these advantages may manifest more prominently in defensive metrics, pitching effectiveness, or overall game strategy rather than raw run counts. Furthermore, home teams might experiment more with lineups or tactics, potentially lowering scoring in certain scenarios. In conclusion, our analysis suggests that playing at home does not significantly increase run rates, but this does not negate the broader concept of home-field advantage in baseball.

Players, Stadium, and Pitching Hands

Up till now we have addressed the relationship between stadiums, success, and their design, expanding that into the relationship between teams, success, and the stadium they played in. Let's broaden our scope once more.

To recap: the design of a stadium negatively impacts all players' performance by hindering the possibility of making Home Runs. Further, regardless of if a stadium is "Home" (that is, where a given team practices) or "Away", stadiums have an overall negative effect on teams themselves. This begs the question: what exactly *is* the relationship between Baseball and all those iconic Baseball Diamonds? Does every stadium have a negative impact upon every player? Are there some stadiums that are particularly damaging to player performance? And, lastly, are baseball stadiums even more influential on player success than their direct opponents?

Data Overview

We begin by reusing the same dataset from when we analyzed the relationship between teams and their home stadiums that detailed individual player statistics from 2017-2020. Of particular note for the current collection of questions are the *H.R.*, *Pitch.Hand*, and *Stadium* statistics, which respectively cover:

1. The Home Run count for every player in every game in each of the four represented seasons;
2. The hand each ball was pitched with (either Right or Left) in every game in each of the four represented seasons; and
3. The stadium for all the combinations of hitter/pitcher in every game in each of the four represented seasons

Pitch.Hand is a derived variable, coming from the *Starting.Pitcher* statistic given in the dataset that includes a byline denoting if a pitcher used their right of left hand to lob the ball. Both the *Pitch.Hand* and *Stadium* values were converted to numerics for ease of computation. For *Pitch.Hand*: 1 corresponds to the Right Hand, and 2 to the Left Hand. For *Stadium*, each stadium was given a representative number, 1 through 38, inclusive (see **Appendix 8** for more details). An example table is displayed below in Figure 3.1.

Player	Stadium	Pitch.Hand	HR
A.J. Cole	Citi Field	R	0
A.J. Cole	Citi Field	R	0
A.J. Cole	Citizens Bank Park	R	0
A.J. Cole	Citizens Bank Park	R	0
A.J. Cole	Marlins Park	L	0
A.J. Cole	Marlins Park	R	0
A.J. Cole	Marlins Park	R	0
A.J. Cole	Marlins Park	R	0
A.J. Cole	Nationals Park	L	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	Nationals Park	R	0
A.J. Cole	SunTrust Park	R	1

Figure 3.1

As we can see from the above, between the years of 2017 and 2020, player A.J. Cole faced a total of 17 different pitchers across 5 different stadiums, only ever scoring a home run once when pitched a right-handed ball at SunTrust Park. Similar data (with an emphasis on no home-runs hit) is seen for all active players.

Because our questions are so multifaceted, building off of each other and the like, we will begin by viewing relationships in isolation: home run count apropos stadiums and then home run counts apropos pitching hand using Hierarchical Bayesian models. To verify the last question posed, we will combine the pitching hand and stadium information into a Bayesian Regression model.

Poisson Model - Home Run Count v. Stadium

Below is figure 3.1, a density plot of the Home Run Count(s) for all players active and all games played in the 2017-2020 seasons.

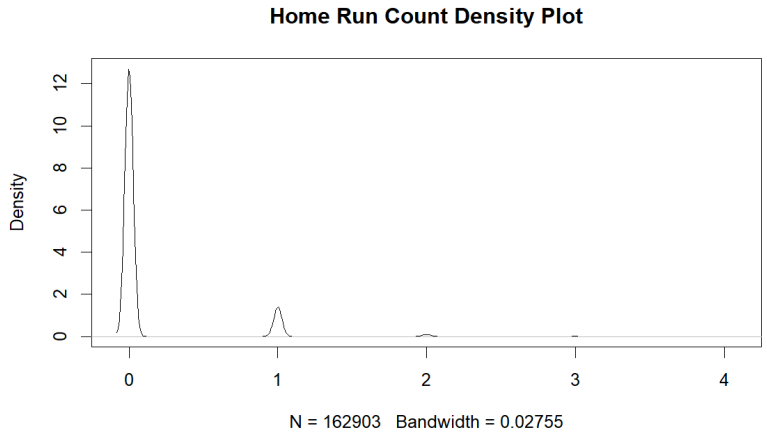


Figure 3.2

Although sampled data should not be exclusively used to determine the priors for Home Run Count, it is abundantly clear from Figure 3.1 that any continuous distribution would be ill-fit for the variable. Logically, this leads us to either using a Negative Binomial Prior or a Poisson prior for the Home Run Counts.

Constructing our initial model, we began with the former - a Poisson prior - and assume that overdispersion is negligible. We will confirm this hypothesis later. For simplicity's sake (and to provide us with a baseline for analysis), we select uninformative priors. These generates a hierarchical model such that:

$$HomeRuns \sim Poisson(\lambda); \qquad log(\lambda) = \alpha_i\,(stadium_i); \qquad \alpha_i \sim N(0,\ 1) \qquad \textbf{(4)}$$

When running a Monte-Carlo Markov Chain (MCMC) with the specified priors, 3 chains, and 5000 iterations, it's important to note that we see all the variables converge (see **Appendix 9** for example Trace Plots). The Gelman-Rubin Diagnostic confirms this as all 38 Stadiums (α_i) have a point estimate of 1. Further, the ACF plots (see **Appendix 10**) show low autocorrelation, confirming that the Poisson distribution is a clear thread of interest. Figure 3.3 shows the Posterior Predictive Check for these uninformative priors.

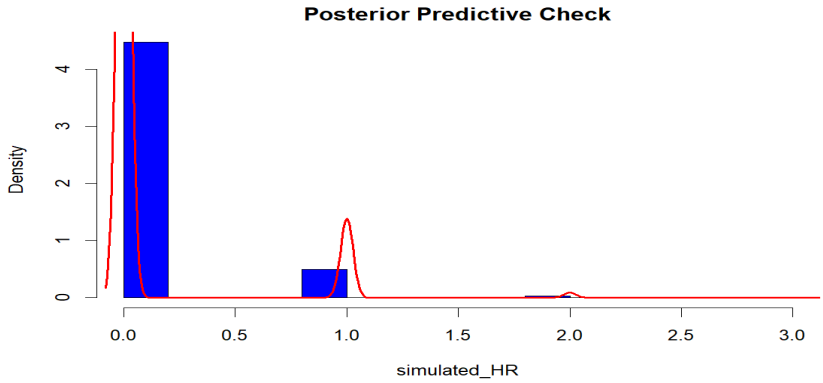


Figure 3.3

From the above plot, we can see that uninformative priors (in blue) actually underestimate the number of home runs we see in the observed data! In an effort to improve these results, let us run this same hierarchical Poisson model with different priors. Namely:

1. A low variance and a mean taken from the observed data ($\alpha_i \sim N(0.110974,\ 0.01)$);
2. A zero mean and a variance taken from the observed data ($\alpha_i \sim N(0,\ 0.1138833)$);
3. A model where both the variance and mean are taken from the observed data $\alpha_i \sim N(0.110974,\ \ 0.1138833)$; and
4. A broad model with a 0 mean and 100 valued variance ($\alpha_i \sim N(0,\ 100)$)

To directly compare all four of these models with the uninformative prior, we will use Deviance Information Criterion (DIC). As a baseline, from the DIC, we receive a penalized deviance of 116652 for the uninformative priors. For the respective models 1-4 above, the penalized deviances are 116652, 116650, 116650, and 118659. Consider now that the lowest penalized deviances were for the second and third model - to determine the overall best model, we analyze the penalty term. For those models, the penalty terms were 37.22 (zero mean, observed variance) and 37.46 (observed mean, observed variance). Ergo, our best Poisson model is one with a 0 mean and the observed variance. Figure 3.4 shows the Posterior Predictive Check for this model.

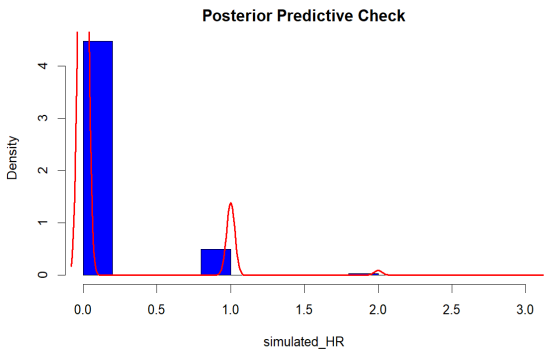


Figure 3.4

Note that figures 3.3 and 3.4 are nearly identical! What this indicates is that the effects of changing the priors are minimal at best; this Poisson model is considered "insensitive." However, because this model has a slightly better DIC, it's worth investigating the estimates of each α_i in an effort to answer some of our posed questions (a full list can be found in **Appendix 11**); of particular interest are "does every stadium have a negative impact upon every player?" and "are there some stadiums that are particularly damaging to player performance?" What we see from our results is yes, all stadiums have a negative effect on player performance. Interestingly, most of these values lie in 95% confidence intervals (see **Appendix 12**) ranging from roughly -3 to -2: this implies that all stadiums *could* prevent anywhere from 2 to 3 home runs per each player. Further, there is one stadium that is most influential: Stadium #34 - the Tokyo Dome - which could prevent anywhere between 2 and 5 home runs per each player (and on average prevents 3). Of course this model is still not very accurate. It's worth investigating a different prior in an attempt to improve our results.

Negative Binomial Model - Home Run Count v. Stadium

As mentioned earlier, based on the density plot of home run counts, instead of a Poisson prior we could have used a Negative Binomial prior which introduces an overdispersion parameter. Because we have already performed sensitivity analysis and determined semi-optimal hyperparameters for α_i, the only aspect of this secondary model that requires modification is the overdispersion parameter. Because the Poisson model is consistently underestimating the number of home runs, we will assume that overdispersion - or, an overestimation of the data that expands beyond the expected number of counts (ie. achieving 8 home runs in a single game) - is minimal at best.

$$HomeRuns \sim NegBinom(p,\ \phi);\ p\ = \frac{\phi}{\phi + \alpha_{i\,(stadium_i)}};\ \phi \sim gamma(5,\ 0.5);\ \alpha_i \sim N(0,\ 0.1138) \quad (\mathbf{5})$$

Where r is the overdispersion parameter. When running MCMC on this Hierarchical Negative Binomial model with 3 chains and a reduced number of 1,000 iterations, we again see that the model converges (see **Appendix 13** for example Trace Plots), something that we again verify using the

Gelman-Rubin Diagnostic (see **Appendix 14**). The Posterior Predictive Check, however, reveals disappointing results.

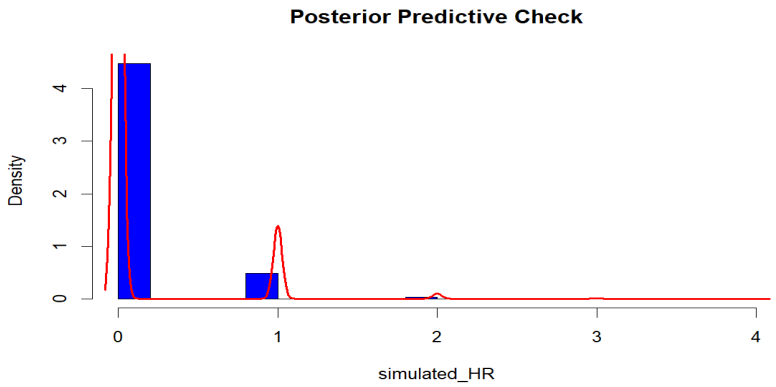


Figure 3.5

This is nearly identical to the Poisson distribution, a statement supported by the DIC for this model, 116619, which isn't statistically significantly different from the prior DIC for the optimal Poisson model. In this same vein, we find similar confidence intervals (see **Appendix 15**) for all α values, as well as similar means (see **Appendix 16**). It should be noted that the additional dispersion term - ϕ (denoted as r in diagnostic tests) - is only 5.974, which is quite low. This verifies the assumption that overdispersion is low as, for the converse to be true, we would expect to see $\phi > 10$. Now, because this additional analysis yielded nearly identical results, sensitivity analysis is forgone.

Poisson Model - Home Run Count v. Pitching Hand

Having exhausted our analysis of home run counts and stadiums in isolation, let us turn to our other variable of interest: the pitching hand of each ball thrown. Recalling that this is still modeling Home Runs, we will choose to reuse the Hierarchical Poisson model as the overdispersion parameter of the Hierarchical Negative Binomial model was non-influential, complexing the model unnecessarily.

$$HomeRuns \sim Poisson(\lambda); \qquad log(\lambda) = \beta_1 (hand); \qquad \beta_1 \sim N(0,\ 1) \qquad \textbf{(6)}$$

Again recall the last analysis of the Hierarchical Poisson Model and Baseball Stadiums where the most effective hyperparameters were the uninformative priors and a zero mean with observed variance priors
($\beta_1 \sim N(0,\ 0.1138833)$). For the sake of consistency, we will reuse both of those hyperparameter combinations along with using the DIC for comparison. For the uninformative hyperparameters, we achieve a penalized deviance of 125322; we also achieve a penalized deviance of 125322 for the other combination of priors! Again looking at the penalty term to "break the tie", we see a penalty of 0.953 for

the former hyperparameters and a penalty of 0.9906 for the latter. This implies that the uninformative hyperparameters are actually *more* accurate than its contemporary. Figure 3.6 shows the Posterior Predictive Check for this Hierarchical Poisson Model.

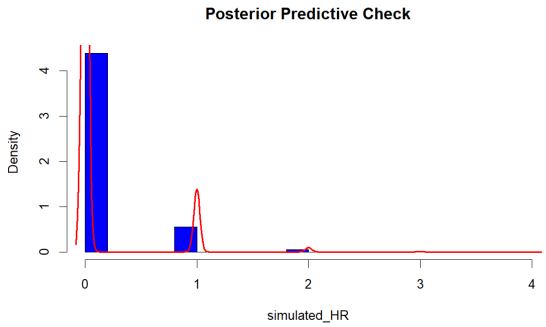


Figure 3.6

Again, we see that the variable on its own again underestimates the number of home runs! Shockingly, this is also near identical to the results of the Hierarchical Poisson Model from our stadium analysis. This implies that both variables, when viewed in isolation, are equally as effective in predicting home run counts. This gives credence to analyzing a regression model. Further, the pitching hand *also* has a negative effect on the number of home runs hit (see **Appendix 17**) with a mean value of -1.30. Recall that we defined hitting with the right hand as a one, the left a two. The mean value for β_1 - which does converge (see **Appendix 18** for Trace Plots) - is erring closer to 1 than to 2, implying that pitching with the right hand actually has more influence than pitching with the left, an interesting conclusion to draw.

Regression Model - Home Run Count v. Stadiums & Pitching Hand

To answer our final (and arguably most important) question posed - "are baseball stadiums even more influential on player success than their direct opponents?" - we need to create a regression model that combines both the Pitching Hand and stadium variables. Out of an abundance of caution, we will use a Negative Binomial Prior to simulate the home runs as combining the two terms could lead to overdispersion. Additionally, we will reuse the best performing hyperparameters for both the Stadium prior and the Pitching prior divined in the previous analyses.

$$HomeRuns \sim NegBinom(p,\ r); \qquad p\ = \frac{\phi}{\phi + \mu_i}; \qquad \mu_i = (\alpha_i \cdot stadium_i) + (\beta_1 \cdot hand)$$
$$\phi \sim gamma(5,\ 0.5); \qquad \beta_1 \sim N(0,\ 1); \qquad \alpha_i \sim N(0,\ 0.1138833); \qquad (7)$$

When running MCMC, again with 3 chains and 5,000 iterations, we see complete convergence of all variables (see **Appendix 19** for Trace Plots). It's worth noting that the negative association with the stadiums still holds in this combined model, with the Tokyo Dome being the most negatively correlated

stadium of all 38. However, the changes begin with β_1, the pitching hand variable. In the regression model, the mean value for β_1 is 0.0008151, which is effectively zero (see **Appendix 20** for all mean values). This implies that the influence stadiums have on player success is suffocating when viewed in tandem with the hand that a ball was pitched with, an assertion that is supported by the Posterior Predictive Plot, which is again near identical to the ones before (see **Appendix 21**).

Our answer, then, is clear. Stadiums *are* more influential on player success than the opponents they go against.

Conclusion

Predicting Baseball is a complicated beast, and one that cannot be fully captured in just a few variables. However, that does not mean certain aspects of the sport cannot be illuminated through specific analysis, particularly ones using a non-frequentist approach. Overall, we can confidently say that there do exist relationships between Home Run counts, the Stadiums a player (or team) performs in and how that stadium is designed.

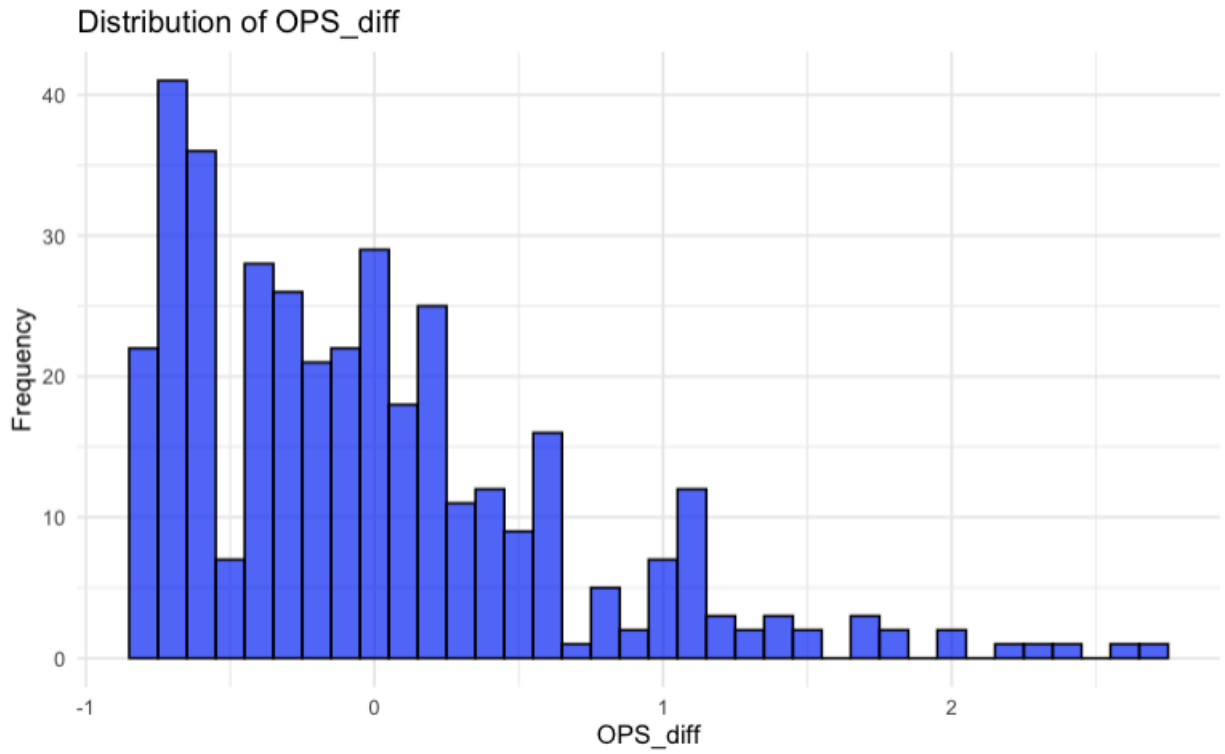
More specifically, there is a very strong correlation between *where* a baseball game is played and how much success a team or player sees. Through the evidence that all stadiums have a negative impact on players - while their opponents carry little influence in comparison - we bolster our hypothesis that individual aspects of stadiums are negatively influential as well: one cannot happen logically without the other. All stadiums, in some way, are designed (and effective) in hindering player performance.

This provides more insight into the importance of the home-field advantage: if all stadiums hinder all teams and all players, coaches want to minimize those detriments whenever possible. Although our analysis actually provided evidence for a “homefield disadvantage”, regardless of the accuracy in that conclusion, the relationship between stadium and player still holds strong. If we are correct in thinking that a homefield disadvantage may actually exist (when excusing other factors such as the crowd, team morale, etc.), then the influence of stadiums on player performance is very clearly a domineering factor in determining overall success. If we are wrong, it still supports the conclusion that teams are negatively correlated with stadiums; if the opposite were true, we would expect to see consistent performance across all fields.

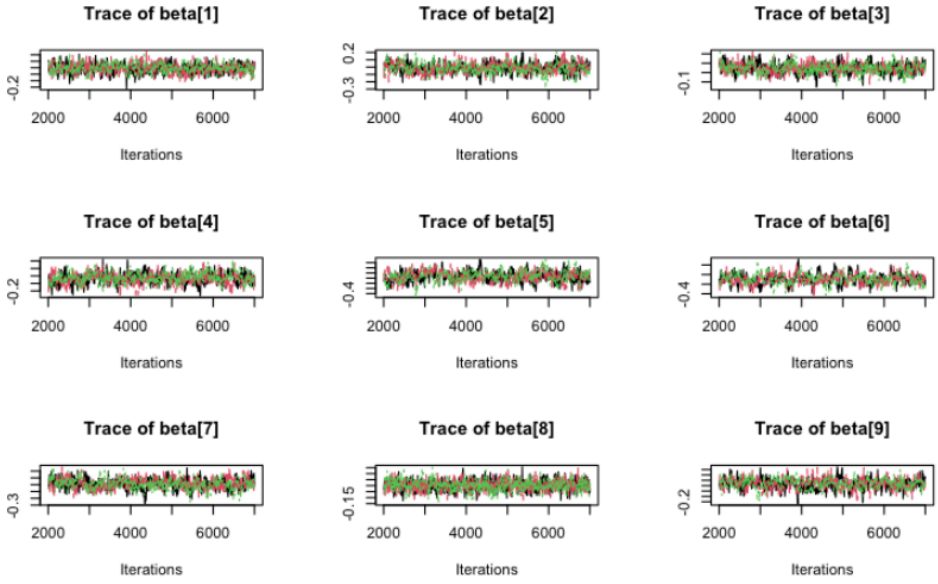
Regardless of if the question is focused on one team, all teams, or all players the answer is clear: baseball stadiums *do* have a clear, negative impact on baseball performance and should be included in all future analyses.

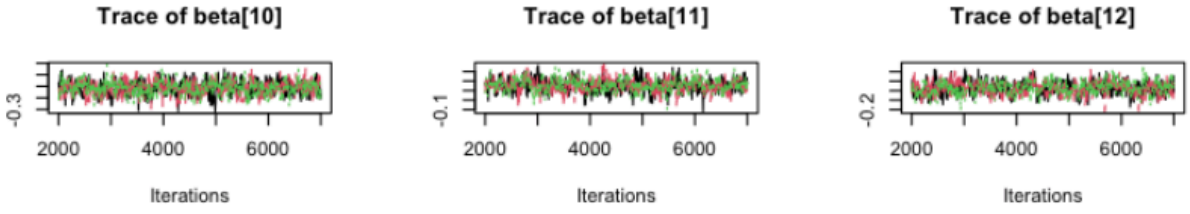
▮

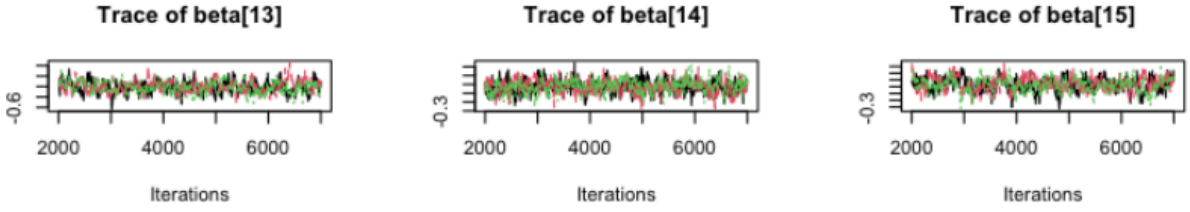
Appendix 1 - Distribution of *OPS_diff*

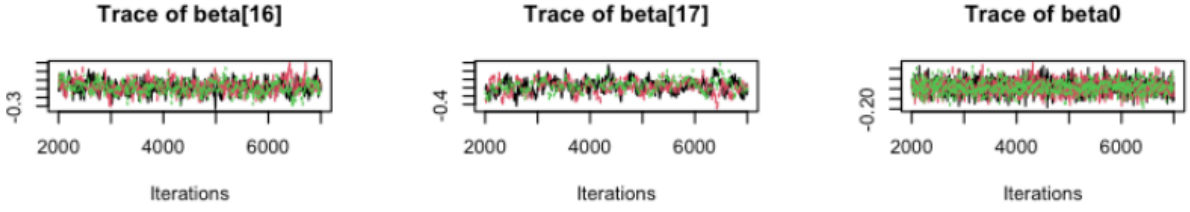


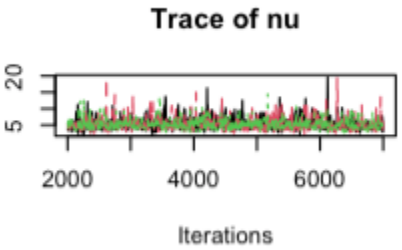
Appendix 2 - Traceplots, Autocorrelation plots, and Gelman-Rubin Diagnostic for Students T-dist model

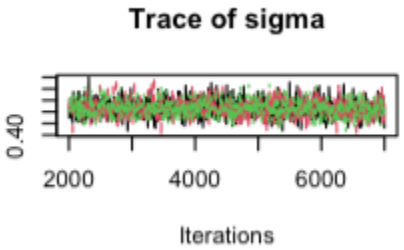






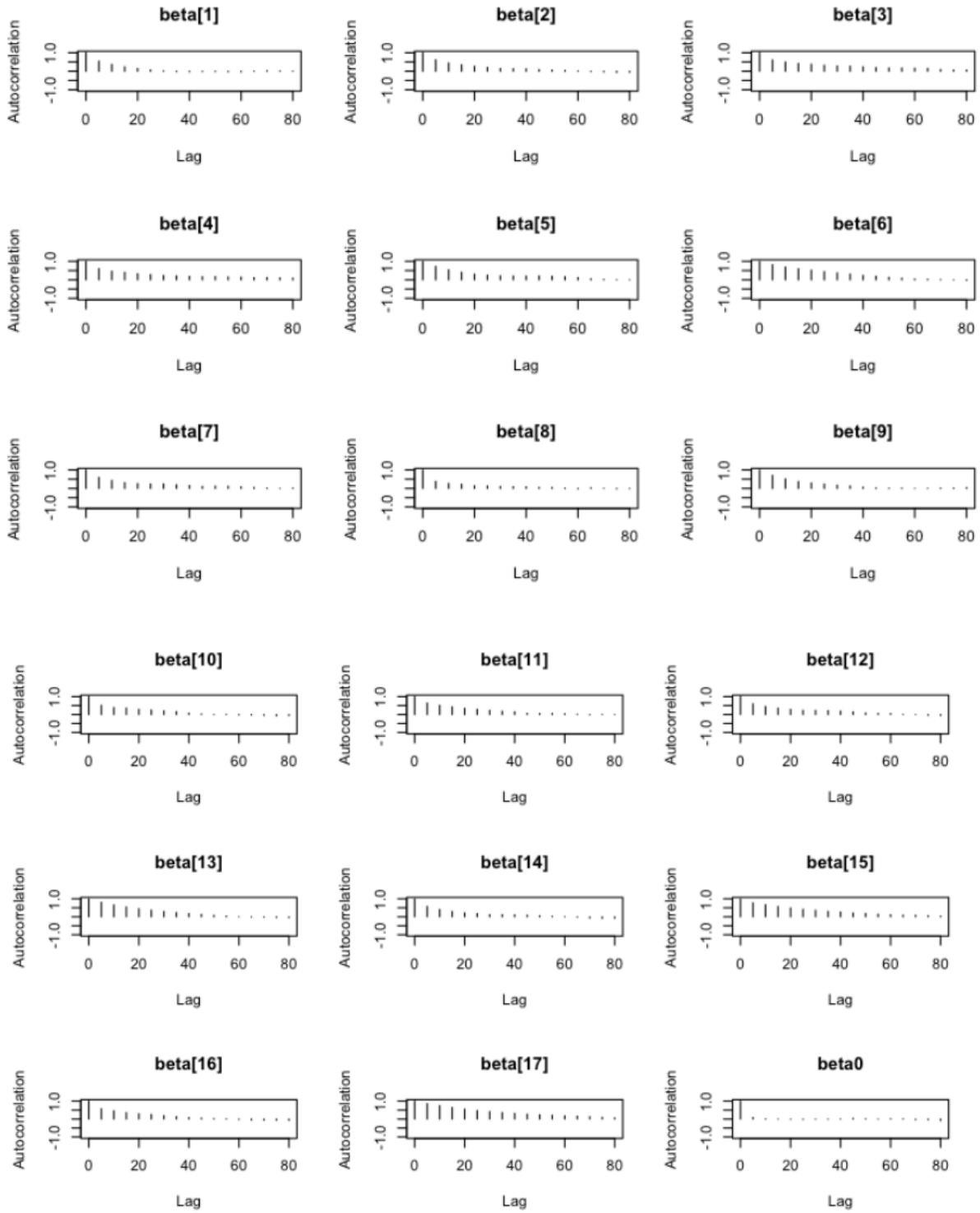




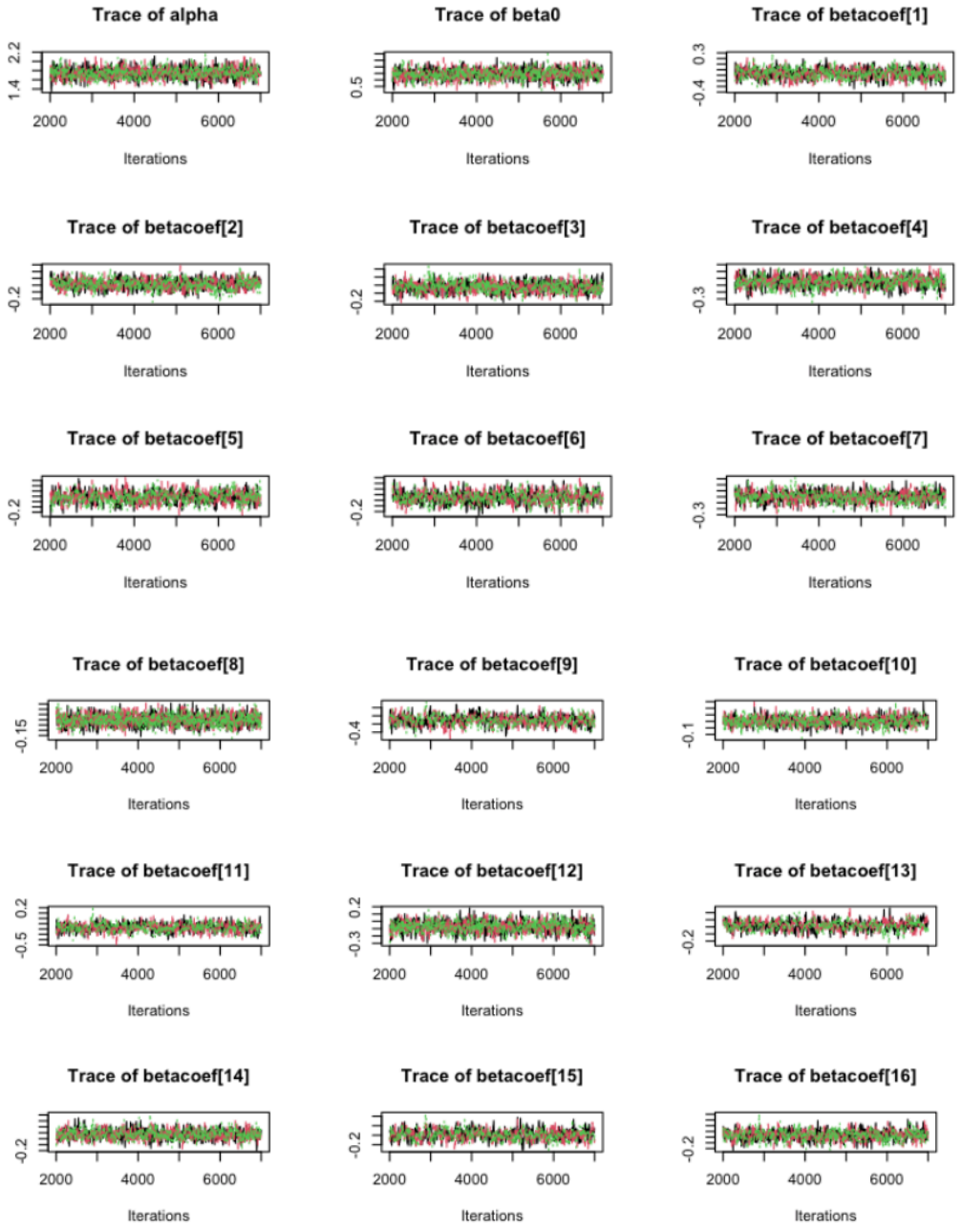


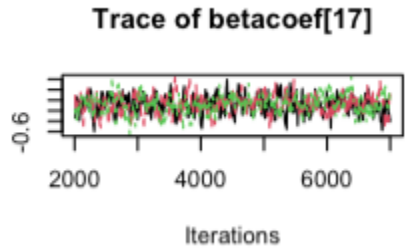
Gelman-Rubin Diagnostic Results

	Point est.	**Upper C.I.**
beta[1]	1.006323	1.020610
beta[2]	1.021574	1.076797
beta[3]	1.007290	1.018233
beta[4]	1.023093	1.081854
beta[5]	1.021869	1.078002
beta[6]	1.002754	1.006013
beta[7]	1.030770	1.107501
beta[8]	1.002302	1.010012
beta[9]	1.009776	1.025048
beta[10]	1.003426	1.013756
beta[11]	1.003915	1.011500
beta[12]	1.014301	1.051553
beta[13]	1.025933	1.090931
beta[14]	1.014798	1.053770
beta[15]	1.014630	1.046925
beta[16]	1.025100	1.085934
beta[17]	1.032369	1.112713
beta0	1.001127	1.002491
nu	1.012235	1.030525
sigma	1.001381	1.006819


beta[1]
beta[2]
beta[3]
beta[4]
beta[5]
beta[6]
beta[7]
beta[8]
beta[9]
beta[10]
beta[11]
beta[12]
beta[13]
beta[14]
beta[15]
beta[16]
beta[17]
beta0
Autocorrelation
-1.0 1.0
0 20 40 60 80
Lag

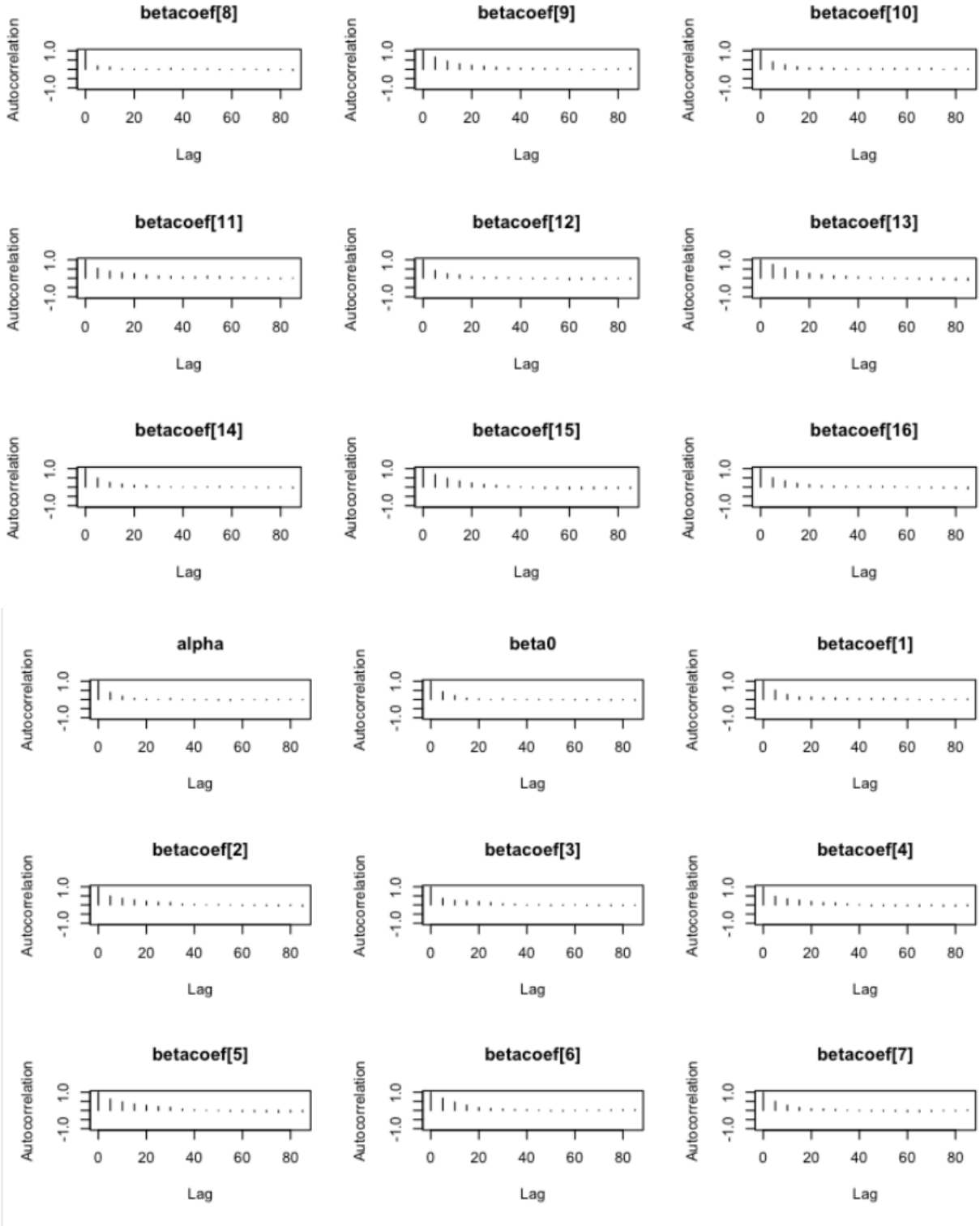
Appendix 3 - Traceplots, Autocorrelation plots, and Gelman-Rubin Diagnostic for Gamma Distribution Model



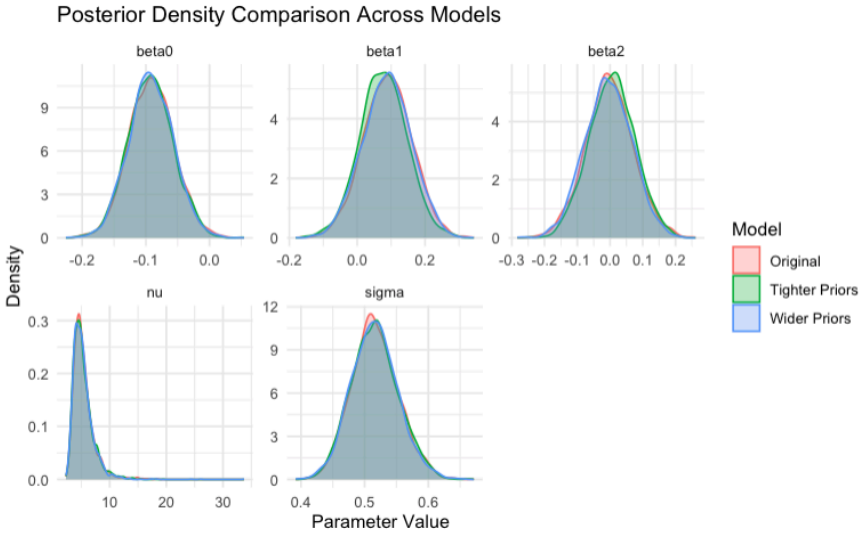


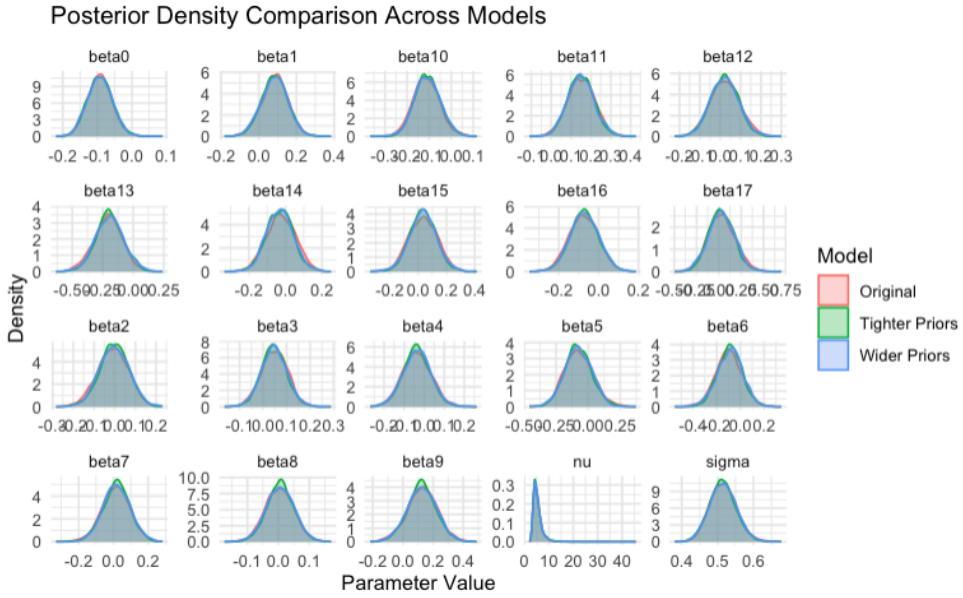
Gelman-Rubin Diagnostic Results

	Point est.	**Upper C.I.**
alpha	0.9997686	1.000635
beta0	0.9997059	1.000813
betacoef[1]	1.0006276	1.003881
betacoef[2]	1.0031638	1.011800
betacoef[3]	1.0083445	1.030751
betacoef[4]	1.0040500	1.015114
betacoef[5]	1.0072213	1.023159
betacoef[6]	1.0066189	1.024745
betacoef[7]	1.0087989	1.033429
betacoef[8]	1.0025426	1.011011
betacoef[9]	1.0042658	1.017384
betacoef[10]	1.0036778	1.013446
betacoef[11]	1.0033912	1.013911
betacoef[12]	1.0010962	1.004742
betacoef[13]	1.0008377	1.004500
betacoef[14]	1.0006597	1.003736
betacoef[15]	1.0102012	1.037617
betacoef[16]	1.0013139	1.005689
betacoef[17]	1.0064028	1.023368


betacoef[8]
betacoef[9]
betacoef[10]
betacoef[11]
betacoef[12]
betacoef[13]
betacoef[14]
betacoef[15]
betacoef[16]
alpha
beta0
betacoef[1]
betacoef[2]
betacoef[3]
betacoef[4]
betacoef[5]
betacoef[6]
betacoef[7]
Autocorrelation
-1.0 1.0
0 20 40 60 80
Lag

Appendix 4 - Results of the Student-t Distribution

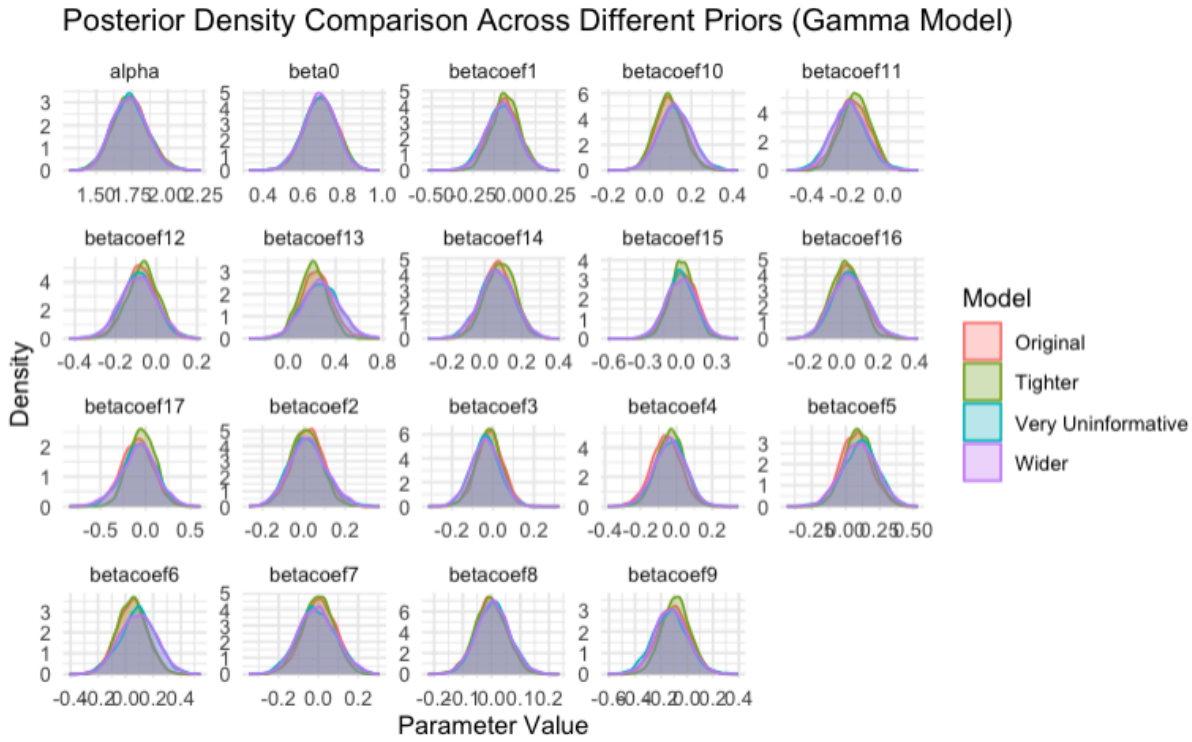




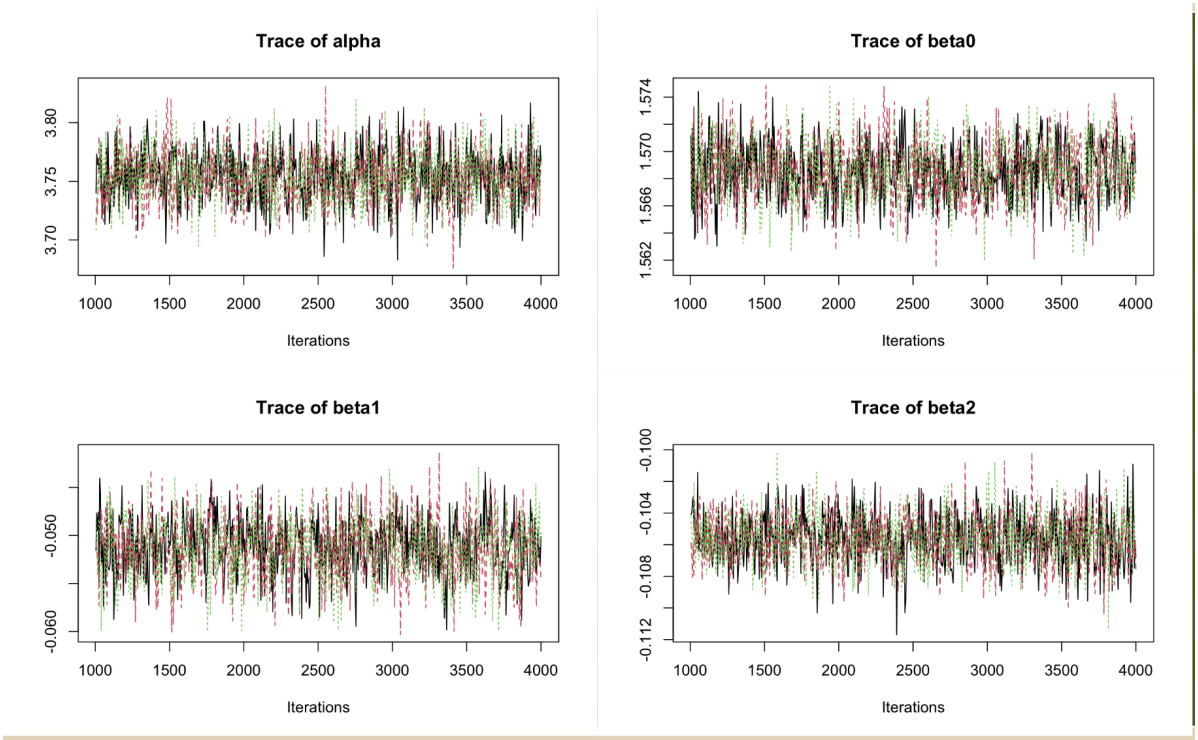
	Parameter	Original_Mean	Original_SD	Tighter_Mean	Tighter_SD	Wider_Mean	Wider_SD
beta[1]	beta[1]	0.0862055	0.0702448	0.0840564	0.0683013	0.0830265	0.0707067
beta[2]	beta[2]	-0.0090638	0.0744150	-0.0020217	0.0693554	-0.0026703	0.0721260
beta[3]	beta[3]	0.0476601	0.0554945	0.0456615	0.0524322	0.0471696	0.0552411
beta[4]	beta[4]	-0.0369013	0.0718595	-0.0375556	0.0668382	-0.0356385	0.0699247
beta[5]	beta[5]	-0.0724546	0.1116017	-0.0768168	0.1038729	-0.0774221	0.1056331
beta[6]	beta[6]	-0.0986621	0.1111829	-0.0879990	0.1006682	-0.0887536	0.1089877
beta[7]	beta[7]	0.0136754	0.0824153	0.0175885	0.0776614	0.0188480	0.0782070
beta[8]	beta[8]	0.0038687	0.0461745	0.0071890	0.0433895	0.0059437	0.0456003
beta[9]	beta[9]	0.1291942	0.1006865	0.1248316	0.0930106	0.1256180	0.0982074
beta[10]	beta[10]	-0.1127416	0.0598721	-0.1064745	0.0574840	-0.1082409	0.0590462
beta[11]	beta[11]	0.1519608	0.0691736	0.1494704	0.0656485	0.1501071	0.0677748
beta[12]	beta[12]	0.0248639	0.0730409	0.0202860	0.0671540	0.0223769	0.0694487
beta[13]	beta[13]	-0.2070501	0.1173187	-0.1939053	0.1099501	-0.1953063	0.1182458
beta[14]	beta[14]	-0.0243189	0.0794477	-0.0305638	0.0734133	-0.0306017	0.0732590
beta[15]	beta[15]	0.0339310	0.1014926	0.0337142	0.0925228	0.0346162	0.0972929
beta[16]	beta[16]	-0.0755835	0.0736075	-0.0715802	0.0714543	-0.0712874	0.0710063
beta[17]	beta[17]	0.0332756	0.1524570	0.0280073	0.1404049	0.0283168	0.1469773
beta0	beta0	-0.0926012	0.0357173	-0.0918645	0.0358829	-0.0924506	0.0364067
nu	nu	5.4105113	2.6760064	5.2913086	1.7498763	5.3384440	1.8831966
sigma	sigma	0.5157800	0.0377183	0.5138262	0.0366612	0.5156609	0.0369201

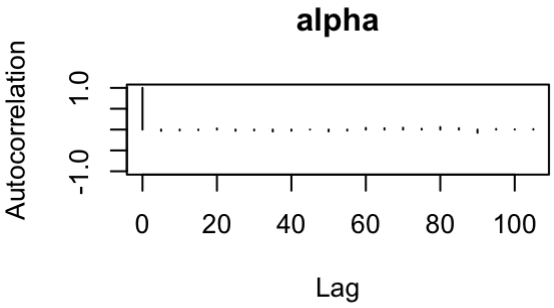
Appendix 5 - Results of the Gamma Distribution

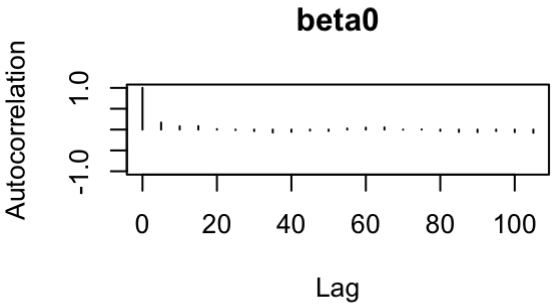
	Parameter	Original_Mean	Original_SD	Tighter_Mean	Tighter_SD	Wider_Mean	Wider_SD
beta[1]	beta[1]	0.0862055	0.0702448	0.0840564	0.0683013	0.0830265	0.0707067
beta[2]	beta[2]	-0.0090638	0.0744150	-0.0020217	0.0693554	-0.0026703	0.0721260
beta[3]	beta[3]	0.0476601	0.0554945	0.0456615	0.0524322	0.0471696	0.0552411
beta[4]	beta[4]	-0.0369013	0.0718595	-0.0375556	0.0668382	-0.0356385	0.0699247
beta[5]	beta[5]	-0.0724546	0.1116017	-0.0768168	0.1038729	-0.0774221	0.1056331
beta[6]	beta[6]	-0.0986621	0.1111829	-0.0879990	0.1006682	-0.0887536	0.1089877
beta[7]	beta[7]	0.0136754	0.0824153	0.0175885	0.0776614	0.0188480	0.0782070
beta[8]	beta[8]	0.0038687	0.0461745	0.0071890	0.0433895	0.0059437	0.0456003
beta[9]	beta[9]	0.1291942	0.1006865	0.1248316	0.0930106	0.1256180	0.0982074
beta[10]	beta[10]	-0.1127416	0.0598721	-0.1064745	0.0574840	-0.1082409	0.0590462
beta[11]	beta[11]	0.1519608	0.0691736	0.1494704	0.0656485	0.1501071	0.0677748
beta[12]	beta[12]	0.0248639	0.0730409	0.0202860	0.0671540	0.0223769	0.0694487
beta[13]	beta[13]	-0.2070501	0.1173187	-0.1939053	0.1099501	-0.1953063	0.1182458
beta[14]	beta[14]	-0.0243189	0.0794477	-0.0305638	0.0734133	-0.0306017	0.0732590
beta[15]	beta[15]	0.0339310	0.1014926	0.0337142	0.0925228	0.0346162	0.0972929
beta[16]	beta[16]	-0.0755835	0.0736075	-0.0715802	0.0714543	-0.0712874	0.0710063
beta[17]	beta[17]	0.0332756	0.1524570	0.0280073	0.1404049	0.0283168	0.1469773
beta0	beta0	-0.0926012	0.0357173	-0.0918645	0.0358829	-0.0924506	0.0364067
nu	nu	5.4105113	2.6760064	5.2913086	1.7498763	5.3384440	1.8831966
sigma	sigma	0.5157800	0.0377183	0.5138262	0.0366612	0.5156609	0.0369201

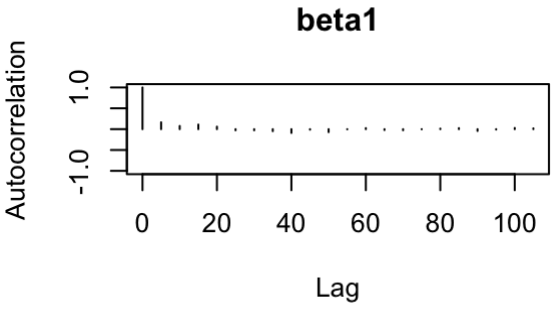


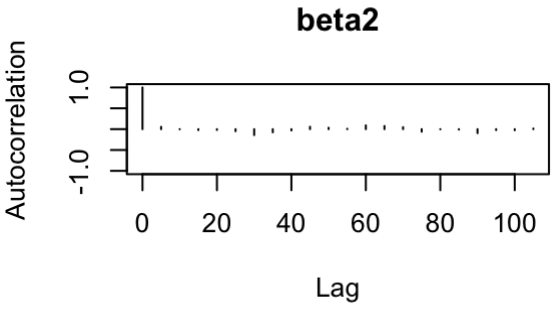
Appendix 6 - Trace and Autocorrelation plots for all MCMC chains analyzing Home-Field Advantage

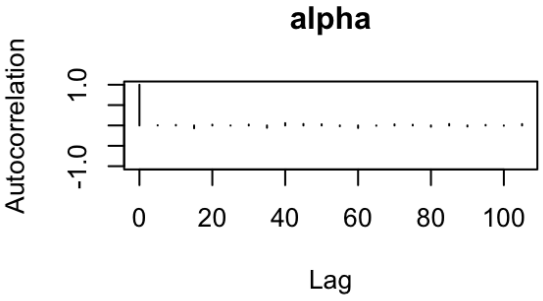


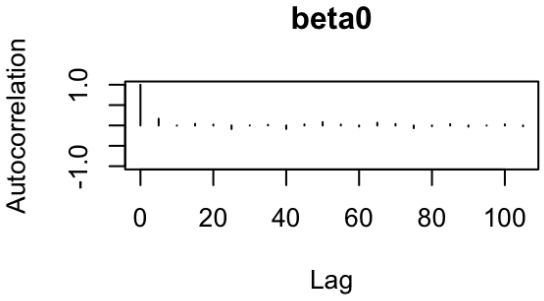

alpha
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag

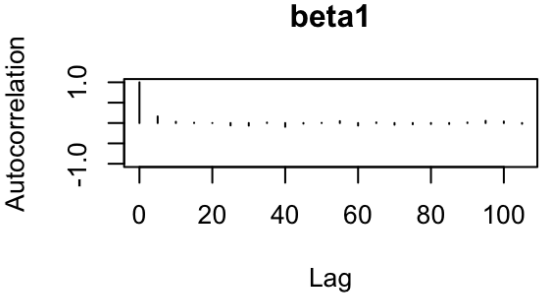

beta0
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag

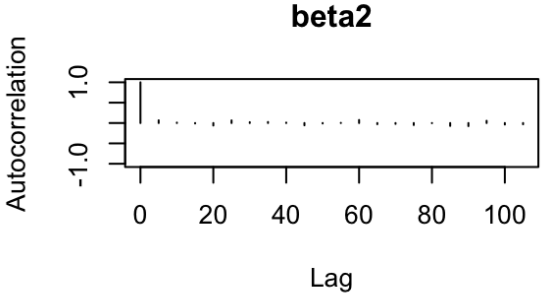

beta1
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag

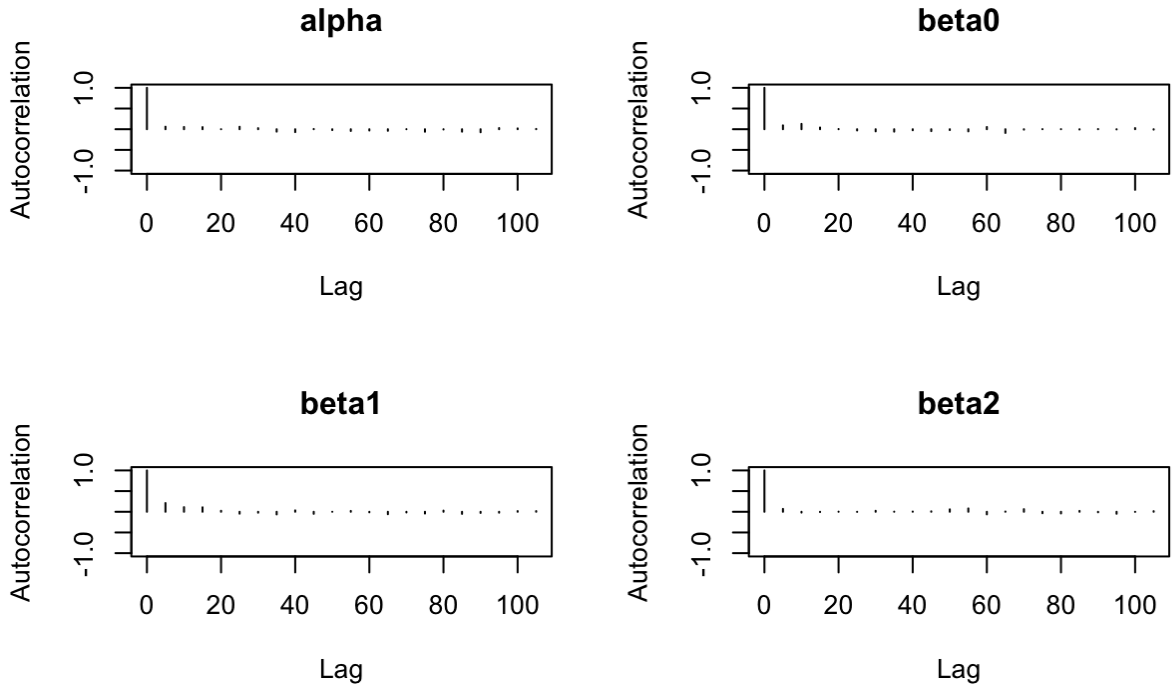

beta2
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag


alpha
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag

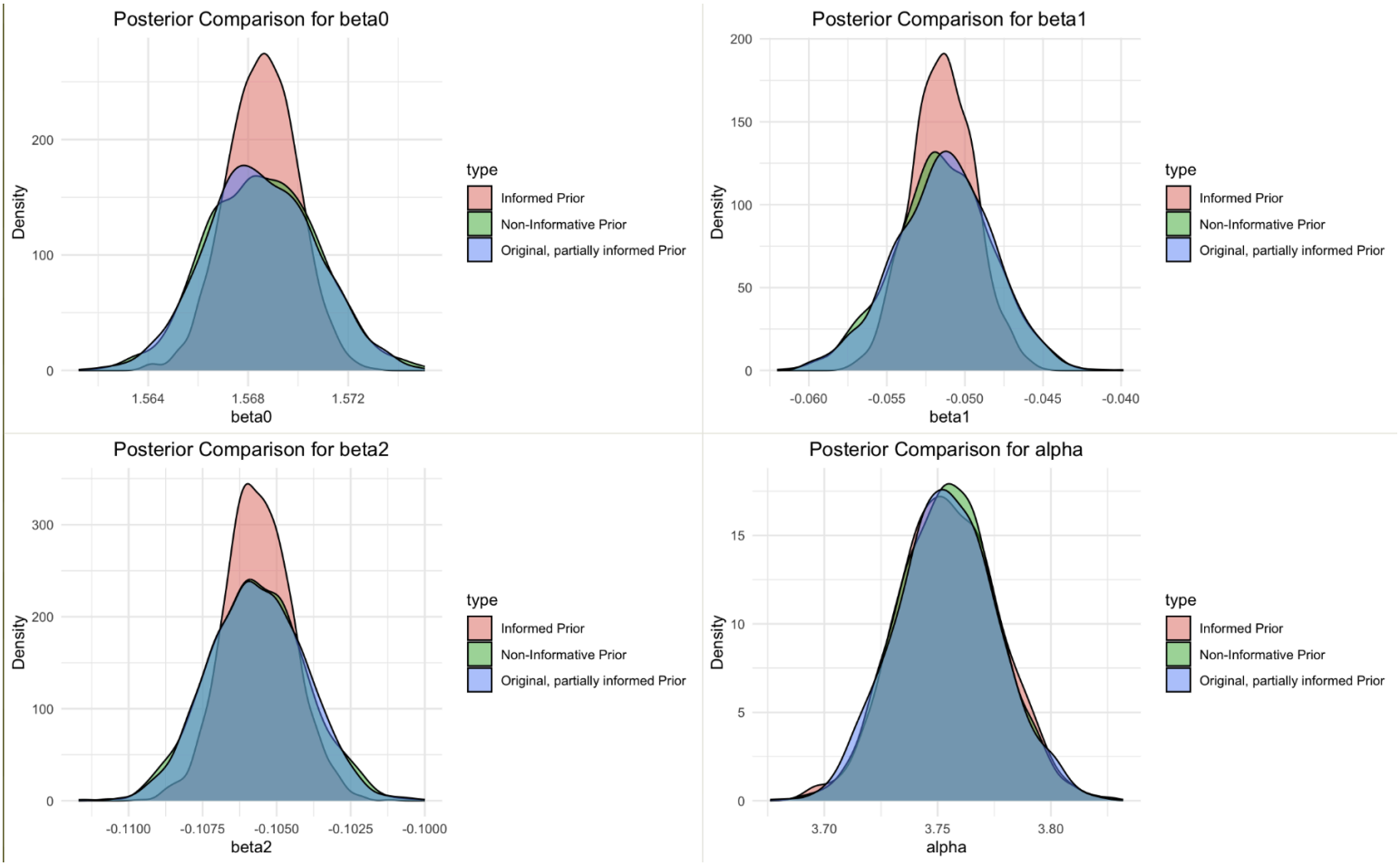

beta0
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag


beta1
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag


beta2
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag


alpha
beta0
beta1
beta2
Autocorrelation
1.0
-1.0
0
20
40
60
80
100
Lag

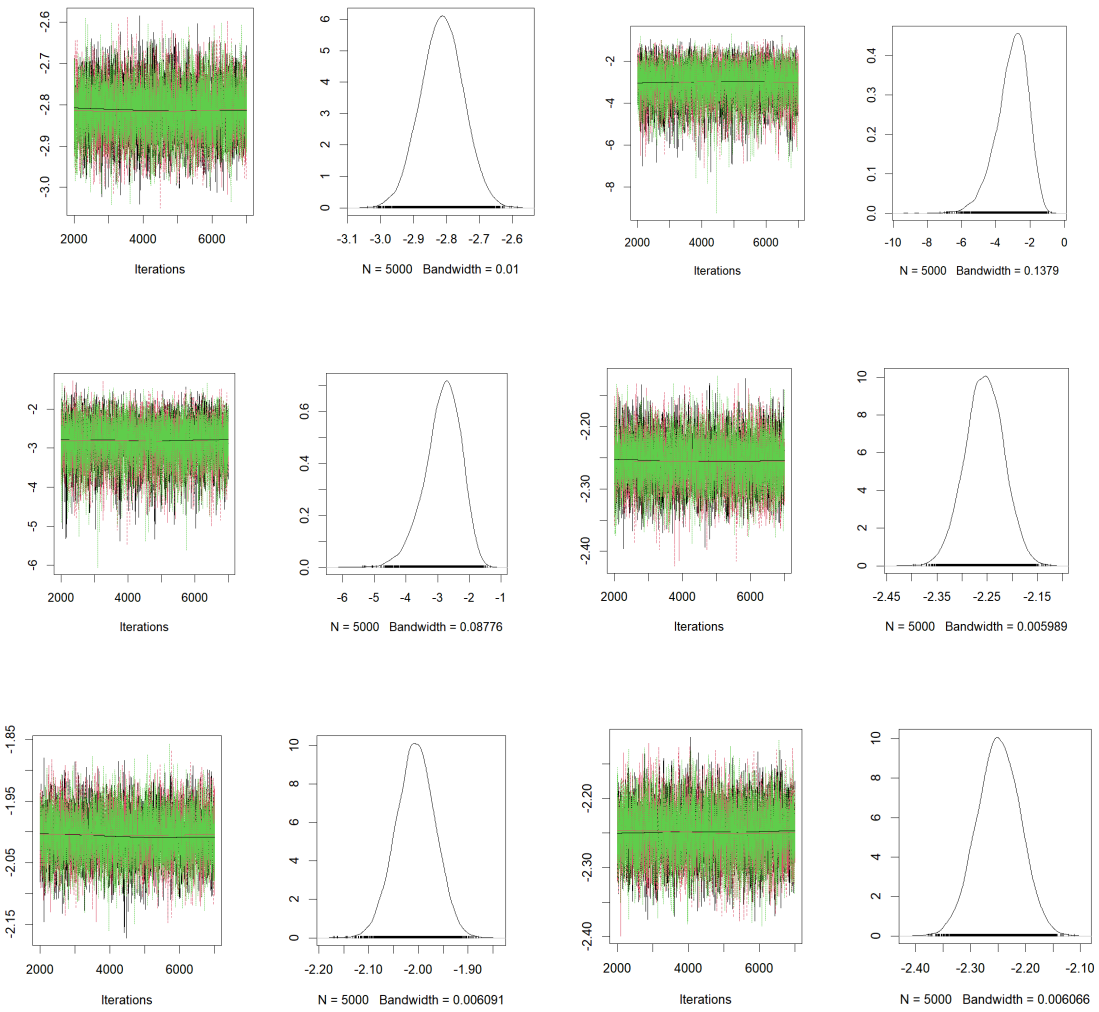
Appendix 7 - Density Plots for all Home-Field Advantage Posteriors



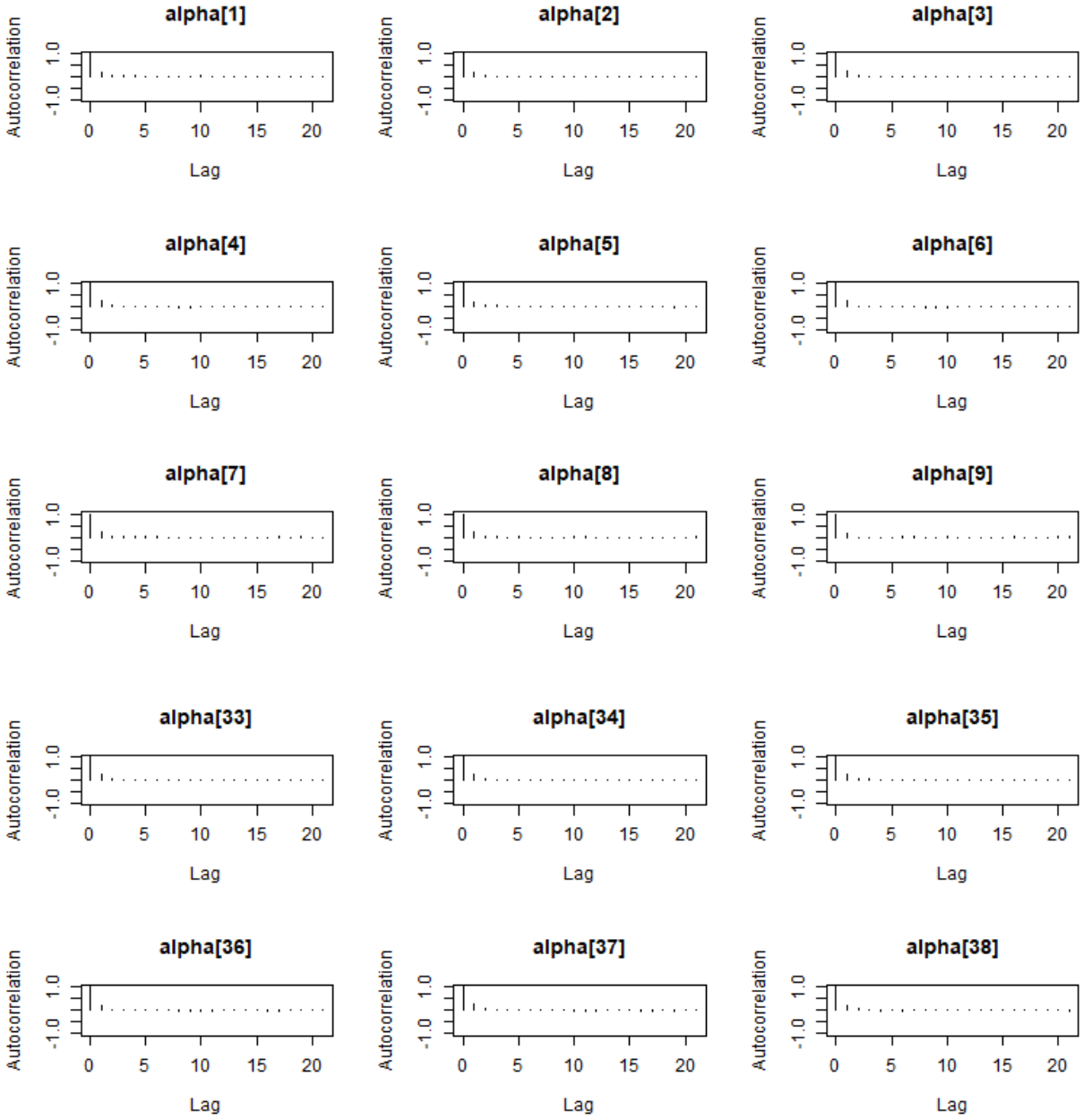
Appendix 8 - Classification of all 38 stadiums

1. Chase Field
2. Busch Stadium III
3. Tropicana Field
4. Oriole Park at Camden Yards
5. Fenway Park
6. Great American Ball Park
7. Minute Maid Park
8. Dodger Stadium
9. Miller Park
10. Target Field
11. Citi Field
12. Oakland-Alameda County Coliseum
13. Globe Life Park in Arlington
14. Nationals Park
15. Guaranteed Rate Field
16. Angel Stadium of Anaheim
17. Coors Field
18. Comerica Park
19. Citizens Bank Park
20. PNC Park
21. Petco Park
22. Wrigley Field
23. Kauffman Stadium
24. Yankee Stadium III
25. Safeco Field
26. AT&T Park
27. Progressive Field
28. Marlins Park
29. Rogers Centre
30. SunTrust Park
31. BB&T Park
32. Estadio Hiram Bithorn
33. Estadio de Beisbol Monterrey
34. Tokyo Dome
35. T-Mobile Park
36. Oracle Park
37. TD Ameritrade Park
38. London Stadium

Appendix 9 - Trace Plots for α_1; α_7; α_{13}; α_{19}; α_{25}; α_{31}; α_{37}



Appendix 10 - ACF plots for α_1 through α_9 and α_{33} through α_{38}



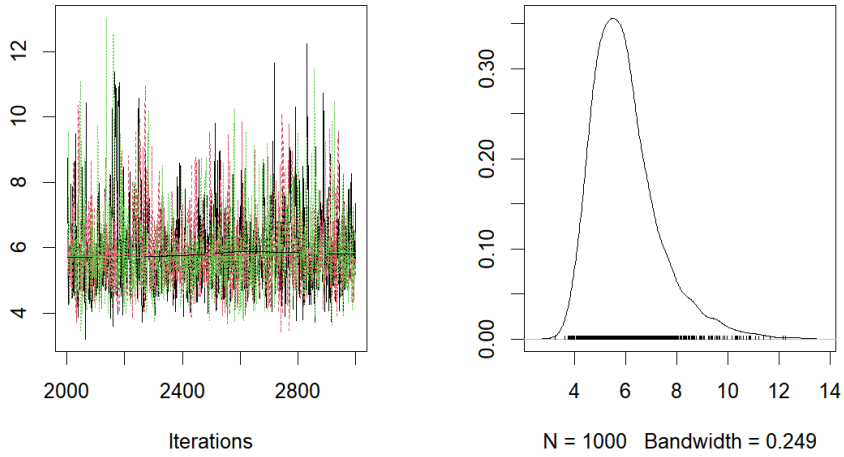
Appendix 11 - α_i estimates for the best Poisson Model

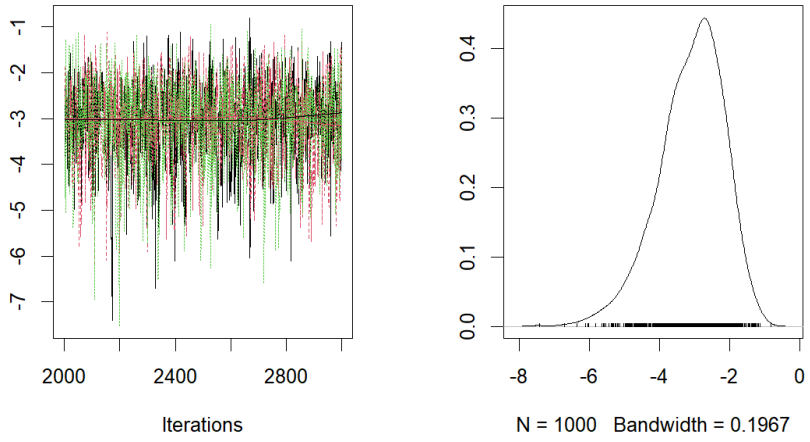
	Mean	SD	Naive SE	Time-series SE
alpha[1]	-2.006	0.03932	0.0003210	0.0004055
alpha[2]	-2.813	0.06464	0.0005278	0.0006505
alpha[3]	-2.842	0.57819	0.0047209	0.0065914
alpha[4]	-2.476	0.04445	0.0003630	0.0004593
alpha[5]	-2.328	0.04066	0.0003320	0.0004187
alpha[6]	-2.336	0.04207	0.0003435	0.0004364
alpha[7]	-2.187	0.03896	0.0003181	0.0004114
alpha[8]	-2.124	0.04164	0.0003400	0.0004402
alpha[9]	-2.159	0.03765	0.0003074	0.0004054
alpha[10]	-2.256	0.03868	0.0003158	0.0003966
alpha[11]	-1.984	0.21705	0.0017722	0.0021249
alpha[12]	-2.020	0.41198	0.0033638	0.0044009
alpha[13]	-2.186	0.04268	0.0003485	0.0004480
alpha[14]	-1.948	0.03800	0.0003103	0.0003853
alpha[15]	-2.190	0.03789	0.0003094	0.0003814
alpha[16]	-2.007	0.04042	0.0003300	0.0004386
alpha[17]	-2.255	0.04570	0.0003732	0.0004622
alpha[18]	-1.557	0.31922	0.0026064	0.0033272
alpha[19]	-2.553	0.04593	0.0003750	0.0004757
alpha[20]	-2.249	0.03916	0.0003197	0.0004081
alpha[21]	-1.961	0.03817	0.0003117	0.0003970
alpha[22]	-2.215	0.03896	0.0003181	0.0004042
alpha[23]	-2.070	0.04046	0.0003304	0.0004113
alpha[24]	-2.558	0.07972	0.0006509	0.0008354
alpha[25]	-1.874	0.03711	0.0003030	0.0003859
alpha[26]	-2.383	0.04246	0.0003467	0.0004406
alpha[27]	-2.552	0.04591	0.0003749	0.0004662
alpha[28]	-2.098	0.04078	0.0003330	0.0004138
alpha[29]	-1.950	0.03808	0.0003109	0.0003955
alpha[30]	-2.119	0.05019	0.0004098	0.0005153
alpha[31]	-2.411	0.04258	0.0003477	0.0004389
alpha[32]	-1.911	0.06415	0.0005238	0.0006734
alpha[33]	-2.038	0.04058	0.0003313	0.0004229
alpha[34]	-3.057	0.92262	0.0075332	0.0103737
alpha[35]	-1.830	0.38376	0.0031334	0.0041035
alpha[36]	-2.215	0.04278	0.0003493	0.0004394
alpha[37]	-2.386	0.04175	0.0003409	0.0004301
alpha[38]	-1.898	0.03762	0.0003072	0.0003884

Appendix 12 - α_i confidence intervals for the best Poisson Model

	2.5%	25%	50%	75%	97.5%
alpha[1]	-2.084	-2.032	-2.006	-1.979	-1.9302
alpha[2]	-2.940	-2.857	-2.813	-2.770	-2.6866
alpha[3]	-4.122	-3.189	-2.790	-2.429	-1.8647
alpha[4]	-2.563	-2.506	-2.475	-2.446	-2.3899
alpha[5]	-2.410	-2.354	-2.327	-2.300	-2.2495
alpha[6]	-2.419	-2.365	-2.336	-2.308	-2.2542
alpha[7]	-2.265	-2.213	-2.188	-2.161	-2.1116
alpha[8]	-2.207	-2.152	-2.124	-2.096	-2.0439
alpha[9]	-2.235	-2.185	-2.159	-2.134	-2.0864
alpha[10]	-2.333	-2.281	-2.255	-2.229	-2.1807
alpha[11]	-2.430	-2.126	-1.978	-1.834	-1.5794
alpha[12]	-2.911	-2.282	-1.992	-1.732	-1.2853
alpha[13]	-2.271	-2.214	-2.186	-2.157	-2.1039
alpha[14]	-2.025	-1.974	-1.948	-1.923	-1.8756
alpha[15]	-2.265	-2.215	-2.189	-2.164	-2.1161
alpha[16]	-2.086	-2.034	-2.006	-1.979	-1.9286
alpha[17]	-2.346	-2.286	-2.254	-2.223	-2.1674
alpha[18]	-2.231	-1.762	-1.539	-1.335	-0.9745
alpha[19]	-2.643	-2.584	-2.552	-2.522	-2.4627
alpha[20]	-2.326	-2.275	-2.249	-2.222	-2.1724
alpha[21]	-2.037	-1.987	-1.961	-1.935	-1.8884
alpha[22]	-2.293	-2.242	-2.215	-2.189	-2.1395
alpha[23]	-2.151	-2.098	-2.070	-2.043	-1.9928
alpha[24]	-2.719	-2.610	-2.556	-2.503	-2.4078
alpha[25]	-1.948	-1.899	-1.873	-1.848	-1.8023
alpha[26]	-2.465	-2.413	-2.383	-2.354	-2.3016
alpha[27]	-2.642	-2.583	-2.551	-2.520	-2.4623
alpha[28]	-2.178	-2.125	-2.098	-2.070	-2.0183
alpha[29]	-2.026	-1.976	-1.950	-1.924	-1.8764
alpha[30]	-2.217	-2.153	-2.119	-2.085	-2.0226
alpha[31]	-2.496	-2.440	-2.411	-2.382	-2.3290
alpha[32]	-2.040	-1.953	-1.910	-1.868	-1.7885
alpha[33]	-2.118	-2.065	-2.038	-2.011	-1.9587
alpha[34]	-5.127	-3.592	-2.946	-2.399	-1.5493
alpha[35]	-2.650	-2.071	-1.805	-1.561	-1.1498
alpha[36]	-2.298	-2.243	-2.215	-2.186	-2.1318
alpha[37]	-2.469	-2.414	-2.386	-2.358	-2.3060
alpha[38]	-1.972	-1.924	-1.898	-1.872	-1.8252

Appendix 13 - Trace plots for α_{34} and ϕ for the Negative Binomial Stadium Model





Appendix 14 - Gelman-Rubin Diagnostic Results for the Negative Binomial Stadium Model

```
Potential scale reduction factors:

          Point est. Upper C.I.
alpha[1]        1.00       1.00
alpha[2]        1.00       1.01
alpha[3]        1.00       1.01
alpha[4]        1.00       1.00
alpha[5]        1.00       1.00
alpha[6]        1.00       1.00
alpha[7]        1.00       1.01
alpha[8]        1.00       1.01
alpha[9]        1.01       1.02
alpha[10]       1.00       1.00
alpha[11]       1.00       1.01
alpha[12]       1.00       1.01
alpha[13]       1.00       1.01
alpha[14]       1.01       1.02
alpha[15]       1.00       1.00
alpha[16]       1.00       1.00
alpha[17]       1.00       1.00
alpha[18]       1.00       1.00
alpha[19]       1.00       1.00
alpha[20]       1.00       1.00
alpha[21]       1.00       1.00
alpha[22]       1.00       1.00
alpha[23]       1.00       1.00
alpha[24]       1.00       1.00
alpha[25]       1.00       1.00
alpha[26]       1.00       1.00
alpha[27]       1.00       1.00
alpha[28]       1.00       1.00
alpha[29]       1.00       1.00
alpha[30]       1.00       1.00
alpha[31]       1.00       1.00
alpha[32]       1.00       1.01
alpha[33]       1.00       1.00
alpha[34]       1.00       1.00
alpha[35]       1.00       1.01
alpha[36]       1.00       1.00
alpha[37]       1.00       1.00
alpha[38]       1.00       1.00
r               1.00       1.00

Multivariate psrf

1.02
```

Appendix 15 - α_i confidence intervals for the best Negative Binomial Stadium Model

2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
alpha[1]	-2.085	-2.032	-2.004	-1.977	-1.927
alpha[2]	-2.946	-2.858	-2.813	-2.771	-2.695
alpha[3]	-4.158	-3.237	-2.802	-2.425	-1.846
alpha[4]	-2.567	-2.506	-2.475	-2.445	-2.386
alpha[5]	-2.409	-2.355	-2.327	-2.298	-2.245
alpha[6]	-2.418	-2.364	-2.335	-2.304	-2.252
alpha[7]	-2.270	-2.215	-2.186	-2.158	-2.106
alpha[8]	-2.202	-2.151	-2.123	-2.095	-2.043
alpha[9]	-2.232	-2.185	-2.161	-2.134	-2.085
alpha[10]	-2.335	-2.283	-2.255	-2.227	-2.177
alpha[11]	-2.441	-2.131	-1.974	-1.838	-1.587
alpha[12]	-2.891	-2.265	-1.986	-1.724	-1.300
alpha[13]	-2.272	-2.215	-2.185	-2.156	-2.099
alpha[14]	-2.023	-1.974	-1.948	-1.924	-1.871
alpha[15]	-2.269	-2.216	-2.189	-2.162	-2.114
alpha[16]	-2.086	-2.034	-2.007	-1.979	-1.925
alpha[17]	-2.345	-2.287	-2.255	-2.224	-2.165
alpha[18]	-2.222	-1.767	-1.543	-1.343	-0.981
alpha[19]	-2.644	-2.585	-2.552	-2.521	-2.466
alpha[20]	-2.323	-2.274	-2.247	-2.221	-2.172
alpha[21]	-2.039	-1.987	-1.961	-1.935	-1.884
alpha[22]	-2.296	-2.242	-2.218	-2.191	-2.143
alpha[23]	-2.154	-2.099	-2.070	-2.042	-1.991
alpha[24]	-2.715	-2.612	-2.558	-2.503	-2.397
alpha[25]	-1.946	-1.898	-1.872	-1.847	-1.804
alpha[26]	-2.472	-2.414	-2.385	-2.355	-2.304
alpha[27]	-2.643	-2.583	-2.552	-2.520	-2.464
alpha[28]	-2.184	-2.126	-2.098	-2.070	-2.015
alpha[29]	-2.026	-1.976	-1.950	-1.923	-1.876
alpha[30]	-2.222	-2.155	-2.120	-2.084	-2.017
alpha[31]	-2.496	-2.441	-2.410	-2.381	-2.327
alpha[32]	-2.035	-1.949	-1.906	-1.864	-1.789
alpha[33]	-2.115	-2.064	-2.036	-2.010	-1.960
alpha[34]	-5.235	-3.661	-2.987	-2.428	-1.572
alpha[35]	-2.604	-2.074	-1.813	-1.560	-1.139
alpha[36]	-2.303	-2.245	-2.213	-2.185	-2.130
alpha[37]	-2.470	-2.411	-2.384	-2.358	-2.307
alpha[38]	-1.977	-1.924	-1.896	-1.871	-1.823
r	4.081	5.051	5.757	6.612	9.390

Appendix 16 - α_i estimates for the best Negative Binomial Stadium Model

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE
alpha[1]	-2.005	0.04039	0.0007374	0.0009722
alpha[2]	-2.816	0.06469	0.0011810	0.0016452
alpha[3]	-2.859	0.59853	0.0109276	0.0147825
alpha[4]	-2.476	0.04574	0.0008350	0.0011491
alpha[5]	-2.327	0.04158	0.0007591	0.0009555
alpha[6]	-2.334	0.04344	0.0007931	0.0009769
alpha[7]	-2.187	0.04160	0.0007596	0.0009938
alpha[8]	-2.123	0.04126	0.0007533	0.0009746
alpha[9]	-2.160	0.03796	0.0006930	0.0008590
alpha[10]	-2.255	0.04061	0.0007414	0.0009319
alpha[11]	-1.987	0.21383	0.0039040	0.0047359
alpha[12]	-2.010	0.40797	0.0074485	0.0097081
alpha[13]	-2.186	0.04402	0.0008037	0.0010502
alpha[14]	-1.948	0.03854	0.0007037	0.0009328
alpha[15]	-2.190	0.03932	0.0007179	0.0008850
alpha[16]	-2.007	0.03999	0.0007302	0.0010001
alpha[17]	-2.255	0.04613	0.0008422	0.0010901
alpha[18]	-1.561	0.32275	0.0058925	0.0071213
alpha[19]	-2.553	0.04623	0.0008440	0.0011014
alpha[20]	-2.247	0.03957	0.0007224	0.0008896
alpha[21]	-1.961	0.03869	0.0007063	0.0008769
alpha[22]	-2.217	0.03948	0.0007207	0.0008644
alpha[23]	-2.071	0.04167	0.0007607	0.0009484
alpha[24]	-2.557	0.08258	0.0015076	0.0019717
alpha[25]	-1.873	0.03682	0.0006722	0.0008614
alpha[26]	-2.385	0.04312	0.0007872	0.0009718
alpha[27]	-2.552	0.04546	0.0008300	0.0010229
alpha[28]	-2.098	0.04206	0.0007679	0.0010372
alpha[29]	-1.950	0.03847	0.0007024	0.0009069
alpha[30]	-2.120	0.05277	0.0009634	0.0013452
alpha[31]	-2.411	0.04391	0.0008016	0.0009942
alpha[32]	-1.907	0.06346	0.0011587	0.0014945
alpha[33]	-2.037	0.03928	0.0007171	0.0009263
alpha[34]	-3.100	0.93783	0.0171224	0.0233910
alpha[35]	-1.826	0.37734	0.0068893	0.0083216
alpha[36]	-2.215	0.04420	0.0008070	0.0009625
alpha[37]	-2.385	0.04092	0.0007470	0.0010140
alpha[38]	-1.898	0.03955	0.0007221	0.0009474
r	5.974	1.31893	0.0240802	0.0372926

Appendix 17 - Estimates for the Negative Binomial Pitching Hand Model

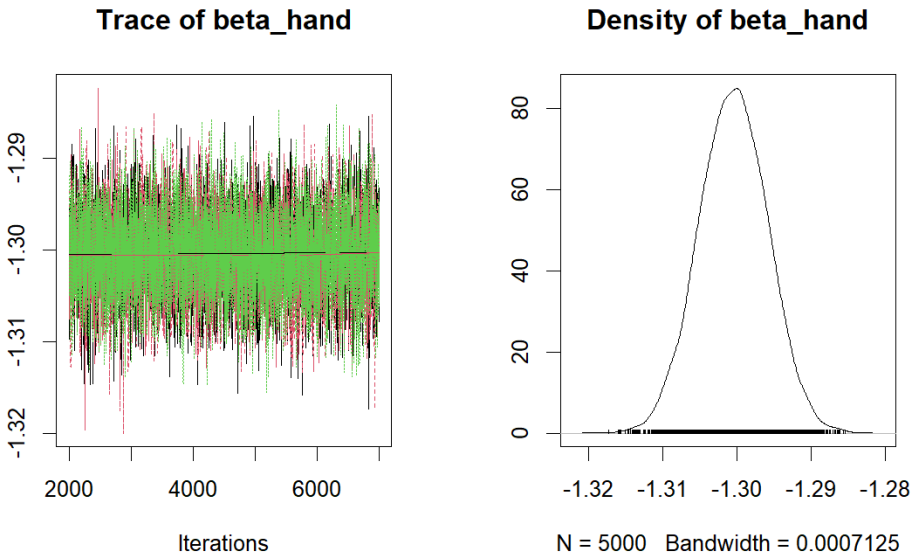
1. Empirical mean and standard deviation for each variable, plus standard error of the mean:

Mean	SD	Naive SE	Time-series SE
-1.300e+00	4.599e-03	3.755e-05	4.730e-05

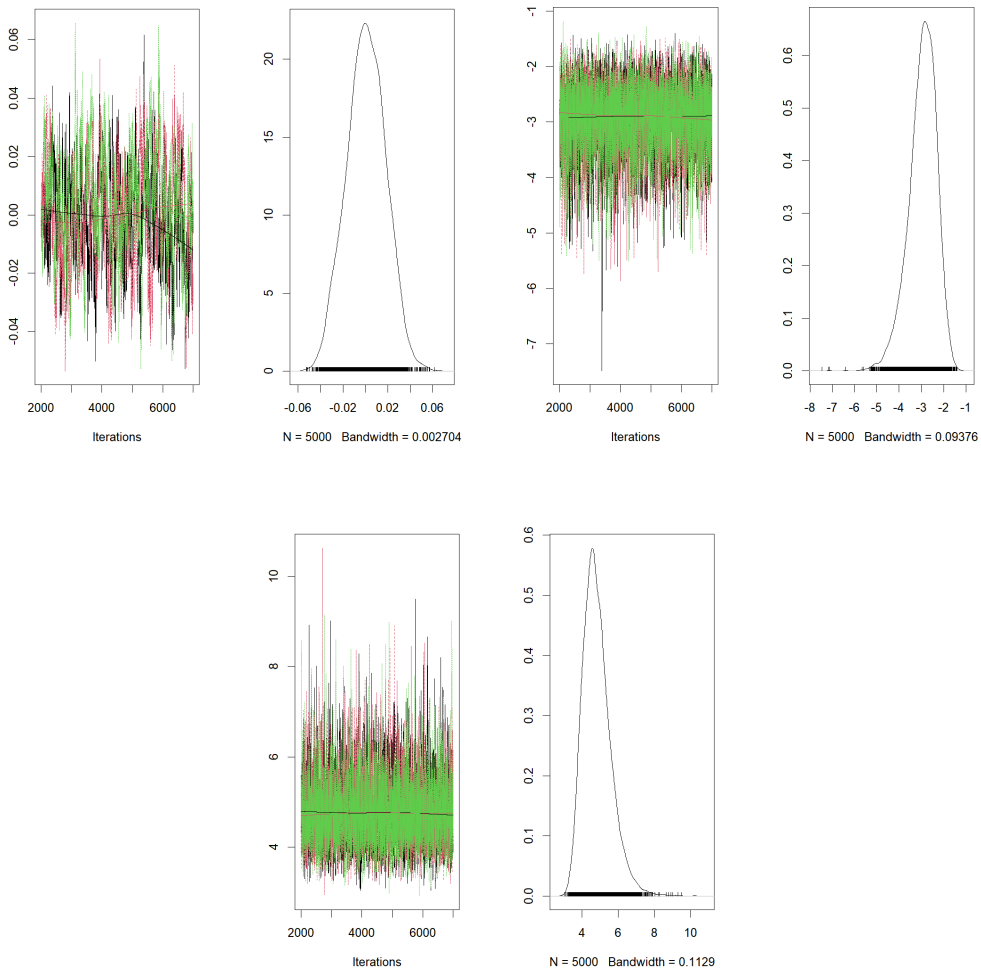
2. Quantiles for each variable:

2.5%	25%	50%	75%	97.5%
-1.309	-1.304	-1.300	-1.297	-1.291

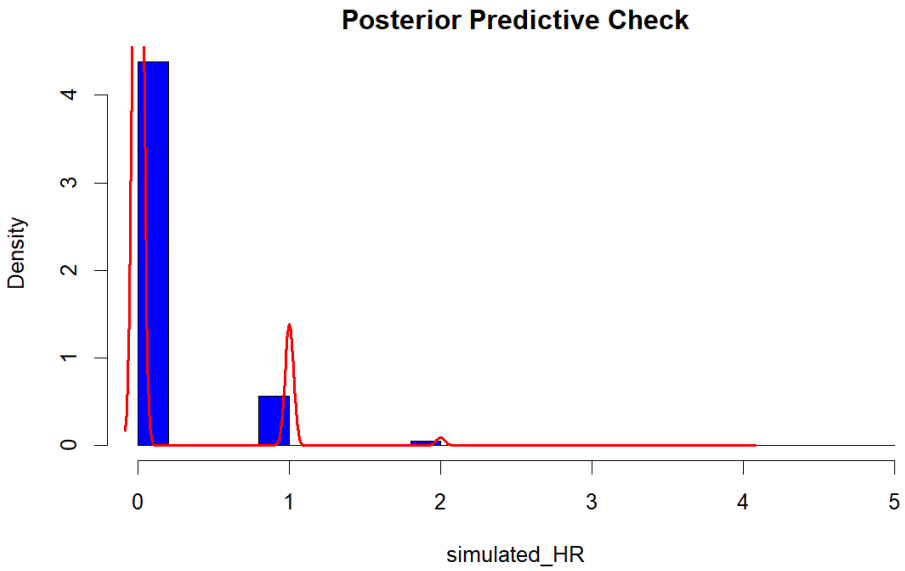
Appendix 18 - Trace Plots for the Negative Binomial Pitching Hand Model



Appendix 19 - Trace Plots for the Regression Model for Stadium + Pitching Hand



Appendix 20 - Posterior Predictive Check for the Regression Model for Stadium + Pitching Hand



Appendix 21 - α_i estimates for the Regression Model for Stadium + Pitching Hand

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

	Mean	SD	Naive SE	Time-series SE
alpha[1]	-2.0074002	0.04948	0.0004040	0.001801
alpha[2]	-2.8164048	0.07055	0.0005760	0.001526
alpha[3]	-2.9534968	0.62939	0.0051389	0.007325
alpha[4]	-2.4773487	0.05430	0.0004433	0.001828
alpha[5]	-2.3292242	0.05090	0.0004156	0.001845
alpha[6]	-2.3376787	0.05234	0.0004274	0.001823
alpha[7]	-2.1886465	0.05025	0.0004103	0.001701
alpha[8]	-2.1263583	0.05156	0.0004210	0.001818
alpha[9]	-2.1609534	0.04822	0.0003937	0.001787
alpha[10]	-2.2566501	0.04864	0.0003971	0.001738
alpha[11]	-1.9947817	0.22680	0.0018518	0.002535
alpha[12]	-2.0496992	0.42770	0.0034921	0.004527
alpha[13]	-2.1877729	0.05258	0.0004293	0.001701
alpha[14]	-1.9492816	0.04707	0.0003843	0.001696
alpha[15]	-2.1921529	0.04849	0.0003959	0.001845
alpha[16]	-2.0090899	0.05124	0.0004184	0.001839
alpha[17]	-2.2561751	0.05476	0.0004471	0.001986
alpha[18]	-1.5773761	0.33458	0.0027318	0.003824
alpha[19]	-2.5552752	0.05594	0.0004568	0.001808
alpha[20]	-2.2502865	0.04995	0.0004078	0.001824
alpha[21]	-1.9624932	0.04831	0.0003944	0.001853
alpha[22]	-2.2169840	0.04960	0.0004050	0.001902
alpha[23]	-2.0723045	0.05068	0.0004138	0.001816
alpha[24]	-2.5611425	0.08592	0.0007016	0.001889
alpha[25]	-1.8746523	0.04786	0.0003908	0.001841
alpha[26]	-2.3857858	0.05101	0.0004165	0.001564
alpha[27]	-2.5542863	0.05604	0.0004576	0.001963
alpha[28]	-2.1001276	0.05251	0.0004288	0.001892
alpha[29]	-1.9516731	0.04877	0.0003982	0.001821
alpha[30]	-2.1206706	0.05754	0.0004698	0.001508
alpha[31]	-2.4120294	0.05288	0.0004317	0.001761
alpha[32]	-1.9123253	0.07086	0.0005786	0.001491
alpha[33]	-2.0400766	0.05063	0.0004134	0.001837
alpha[34]	-3.4812716	1.21719	0.0099383	0.014935
alpha[35]	-1.8539688	0.39942	0.0032613	0.004449
alpha[36]	-2.2163069	0.05329	0.0004352	0.001788
alpha[37]	-2.3868639	0.04998	0.0004081	0.001567
alpha[38]	-1.8993165	0.04656	0.0003802	0.001601
beta_hand	0.0008151	0.01746	0.0001425	0.001035
r	4.8296495	0.78066	0.0063741	0.009581

Project Responsibilities

Andrew completed the first section of the analysis. He analyzed the influence of stadium dimensions on player performance, focusing on Tampa Bay Rays players. He combined and cleaned data from Clem's Baseball dataset and Statcast, linking player OPS differences to stadium characteristics. Through exploratory analysis, including histograms and scatter plots, he assessed data patterns and skewness. Using JAGS, Andrew implemented a Gamma regression model with informed priors, carefully initializing parameters and diagnosing convergence with trace plots, autocorrelation plots, and Gelman-Rubin diagnostics. To ensure robustness, he conducted a sensitivity analysis with various prior specifications. His findings, presented in the final paper, provided insights into how stadium features impact performance and can inform strategic stadium design.

Aditya completed the second section of this analysis, answering the question on whether playing at home significantly impacts a team's run rate. He began by consolidating game-by-game baseball data from 2017 to 2020 and preparing the dataset for modeling. Using visualizations like histograms, scatter plots, and boxplots, Aditya examined patterns in the data, assessed overdispersion, and explored relationships between the predictors and response variables. Aditya then formalized the analysis by specifying priors for each parameter based on intuition and empirical reasoning, opting for weakly informative distributions to maintain flexibility while incorporating realistic assumptions. Using JAGS, he implemented the model through MCMC methods, initializing parameters thoughtfully and diagnosing convergence with trace plots, autocorrelation plots, and diagnostics like Gelman-Rubin and Geweke tests. To ensure the robustness of the results, Aditya conducted a sensitivity analysis, testing the model with non-informative and fully informed priors alongside the partially informed priors initially used. Aditya summarized these findings in both his presentation and final paper, addressing potential errors and offering interpretation of the results.

As to be expected, Brandon completed the third question using the dataset that Aditya had previously developed. Initial preparations before modeling were minimal, as all that was required was an analysis of the distribution of home runs. Brandon began each Bayesian model by selecting uninformative priors, performing robust sensitivity analysis on almost all models from part three to draw supported and confident conclusions about the relationships represented. This involved testing both a Poisson and Negative Binomial prior, along with 4 distinct models. Like his teammates, he also used JAGS to develop his models, along with trace plots and the Gelman-Rubin diagnostics to confirm convergence. DIC was used to compare all variations of models, of which there were 10. Similarly to both Aditya and Andrew, his findings were presented by him in class and can be found in this paper, elucidating the general relationship between ballparks, opposing teams, and player effectiveness. He wrote the introduction and conclusion of the paper, along with most of the formatting of the final document.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Code for Question 1:

````
---
title: "425 Project"
author: "Andrew Sun"
date: "`r Sys.Date()`"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## R Markdown

Background:
Tampa Bay Rays have been attempting to build a new baseball stadium,
(for various reasons). I want to investigate whether or not stadium
factors are so significant that the Rays should intentionally design
their stadium in such a way to benefit their own players. To do this,
we'll select the 4 players on their team with above 350 at-bats
(Yandy Diaz, Brandon Lowe, Josh Lowe, and Jose Caballero), and
investigate whether the combination of these four players
historically play significantly better in some stadium designs than
others.

```{r cars}
file_path <- "/Users/andrewsun/Downloads/Clems Statistics -
Sheet1.csv"


clems_data <- read.csv(file_path, header = TRUE, stringsAsFactors =
FALSE)
head(clems_data)
```

```{r}
clems_data$Stadium..see.notes. <- gsub("*", "",
clems_data$Stadium..see.notes.)
clems_data$Stadium..see.notes. <-
trimws(clems_data$Stadium..see.notes.)
clems_data$Stadium..see.notes.
```
````

```{r}
colnames(clems_data)
head(clems_data, 10)
```

```{r}
# Remove rows where the Stadium column is empty
clems_data <- clems_data[clems_data$Stadium..see.notes. != "", ]

# Reset row indices
row.names(clems_data) <- NULL
```

```{r}
# Rename columns
colnames(clems_data) <- c(
  "Stadium", "Teams", "MLBLifetimeYears", "SeatingCapacity",
"SeatingRowsTypical",
  "Deck2", "UpperDeck", "LowerDeckShade", "UpperDeckShade",
"FairTerritory",
  "FoulTerritory", "FenceHeightLF", "FenceHeightCF", "FenceHeightRF",
  "CFOOrientation", "Backstop", "LeftField", "LeftCenter",
"CenterField",
  "RightCenter", "RightField"
)
```

```{r}
# Convert relevant columns to numeric
numeric_columns <- c(
  "FenceHeightLF", "FenceHeightCF", "FenceHeightRF", "LeftField",
  "LeftCenter", "CenterField", "RightCenter", "RightField"
)

clems_data[numeric_columns] <- lapply(clems_data[numeric_columns],
function(x) as.numeric(as.character(x)))

# Check for conversion issues
summary(clems_data)
```

```{r}
# Replace NAs with the mean (or median) for numeric columns
for (col in numeric_columns) {
  clems_data[[col]][is.na(clems_data[[col]])] <-
mean(clems_data[[col]], na.rm = TRUE)
}
```

```{r}
# Check the cleaned data
str(clems_data)

# Preview the data
head(clems_data)
```

```{r}
clems_data$Stadium
```

```{r}
library(baseballr)
library(dplyr)
library(lubridate)
library(tidyr)

```

```{r}
# Get player IDs
yandy_id <- 650490
brandon_id <- 666139
jose_id <- 676609

# Combine into a vector
player_ids <- c(yandy_id, brandon_id, jose_id)
names(player_ids) <- c("Yandy Diaz", "Brandon Lowe", "Jose
Caballero")

```

```{r}
# Retrieve the active roster for the Tampa Bay Rays (team ID: 139)
rays_roster <- mlb_rosters(team_id = 139, roster_type = "active")

print(rays_roster)

```

```{r}
# Define the date range
start_date <- "2024-04-01"
end_date <- "2024-10-05"
# Function to get pitch-by-pitch data for a player
get_player_statcast_data <- function(player_id, start_date, end_date)
{
  # Retrieve data
  statcast_data <- statcast_search(start_date = start_date, end_date
= end_date, playerid = player_id, player_type = "batter")
  return(statcast_data)
}

# Get statcast data for each player
player_statcast_list <- lapply(player_ids, get_player_statcast_data,
start_date = start_date, end_date = end_date)
names(player_statcast_list) <- names(player_ids)

```

```{r}
player_statcast_list$`Yandy Diaz`$home_team
```

```{r}
# Function to process statcast data and calculate game-level OPS
process_statcast_data <- function(statcast_data) {
  # Filter out irrelevant events
  statcast_data <- statcast_data %>%
    filter(!is.na(events))

  # Create variables for hits, at-bats, etc.
  statcast_data <- statcast_data %>%
    mutate(
```

```
      single = ifelse(events == "single", 1, 0),
      double = ifelse(events == "double", 1, 0),
      triple = ifelse(events == "triple", 1, 0),
      home_run = ifelse(events == "home_run", 1, 0),
      BB = ifelse(events %in% c("walk", "hit_by_pitch",
"intent_walk"), 1, 0),
      HBP = ifelse(events == "hit_by_pitch", 1, 0),
      SF = ifelse(events == "sac_fly", 1, 0),
      AB = ifelse(events %in% c("single", "double", "triple",
"home_run", "strikeout", "field_out", "grounded_into_double_play",
"force_out", "fielders_choice_out", "field_error", "double_play",
"triple_play"), 1, 0),
      H = single + double + triple + home_run,
      TB = single + 2 * double + 3 * triple + 4 * home_run
    )

  # Aggregate to game level
  game_data <- statcast_data %>%
    group_by(game_date, game_pk, player_name = paste(player_name),
home_team, away_team) %>%
    summarise(
      AB = sum(AB),
      H = sum(H),
      BB = sum(BB),
      HBP = sum(HBP),
      SF = sum(SF),
      TB = sum(TB),
      .groups = "drop"
    ) %>%
    mutate(
      OBP = ifelse((AB + BB + HBP + SF) > 0, (H + BB + HBP) / (AB +
BB + HBP + SF), NA),
      SLG = ifelse(AB > 0, TB / AB, NA),
      OPS = OBP + SLG
    )
  return(game_data)
}

# Process each player's statcast data
player_game_logs_processed <- lapply(player_statcast_list,
process_statcast_data)
```

```
```

```{r}
# Combine data for all players
combined_player_logs <- bind_rows(player_game_logs_processed)

```

```{r}
# Create a data frame with team abbreviations and stadium names
team_stadium_mapping <- data.frame(
  home_team = c("ARI", "ATL", "BAL", "BOS", "CHC", "CIN", "CLE",
"COL", "CWS", "DET",
                "HOU", "KC", "LAA", "LAD", "MIA", "MIL", "MIN",
"NYM", "NYY", "OAK",
                "PHI", "PIT", "SD", "SEA", "SF", "STL", "TB", "TEX",
"TOR", "WSH"),
  stadium_name = c("Chase Field", "Truist (ex-SunTrust) Park",
"Oriole Park at Camden Yards", "Fenway Park",
                   "Wrigley Field", "Great American Ballpark",
"Progressive Field", "Coors Field",
                   "Guaranteed Rate Field", "Comerica Park", "Minute
Maid Park", "Kauffman Stadium",
                   "Angel Stadium", "Dodger Stadium", "Marlins Park",
"Miller Park",
                   "Target Field", "Citi Field", "Yankee Stadium",
"Oakland Coliseum",
                   "Citizens Bank Park", "PNC Park", "Petco Park",
"Safeco Park",
                   "Oracle Park", "Busch Stadium III", "Tropicana
Field", "Globe Life Field",
                   "Rogers Centre", "Nationals Park")
)

# View the mapping
print(team_stadium_mapping)

```
```{r}
# Merge the team-stadium mapping with combined player logs
combined_player_logs <- combined_player_logs %>%
  left_join(team_stadium_mapping, by = "home_team")
```

```
```

```{r}
# Ensure stadium names are consistent
combined_player_logs <- combined_player_logs %>%
  mutate(stadium_name = as.character(stadium_name))

clems_data <- clems_data %>%
  mutate(stadium_name = as.character(Stadium))  # Adjust the column
name as per your dataset

# Merge with Clems dataset
final_player_data <- combined_player_logs %>%
  left_join(clems_data, by = "stadium_name")

```

```{r}
final_player_data
# need to include player historical OPS as well
# also need to get league avg OPS to compare
```
```{r}
# Example using season OPS
final_player_data <- final_player_data %>%
  group_by(player_name) %>%
  mutate(
    average_OPS = mean(OPS, na.rm = TRUE),
    OPS_diff = OPS - average_OPS
  ) %>%
  ungroup()

```

```{r}
final_player_data
```
```{r}
library(ggplot2)
ggplot(final_player_data, aes(x = OPS_diff)) +
```

```
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black",
alpha = 0.7) +
  labs(title = "Distribution of OPS_diff", x = "OPS_diff", y =
"Frequency") +
  theme_minimal()
```

```{r}
# Clean and convert to numeric
final_player_data <- final_player_data %>%
  mutate(SeatingCapacity = as.numeric(gsub("[^0-9]", "",
SeatingCapacity)))

# View the cleaned column
head(final_player_data)
```

```{r}
# Clean and convert to numeric
final_player_data <- final_player_data %>%
  mutate(MLBLifetimeYears = as.numeric(gsub("[^0-9]", "",
MLBLifetimeYears)))

# View the cleaned column
head(final_player_data$MLBLifetimeYears)
```

```{r}
# Specify the columns to clean
columns_to_clean <- c(
  "SeatingCapacity", "SeatingRowsTypical",
  "Deck2", "UpperDeck", "LowerDeckShade", "UpperDeckShade",
"FairTerritory",
  "FoulTerritory", "FenceHeightLF", "FenceHeightCF", "FenceHeightRF",
"Backstop", "LeftField", "LeftCenter", "CenterField",
  "RightCenter", "RightField"
)
```

```
# Clean and convert these specific columns to numeric
final_player_data <- final_player_data %>%
  mutate(across(
    all_of(columns_to_clean),
    ~ as.numeric(gsub("[^0-9.-]", "", .))
  ))

# View the cleaned data


final_player_data$CFOOrientation <- NULL

head(final_player_data)

```

```{r}
```

```{r}
# Load necessary libraries
library(dplyr)
library(ggplot2)


# List of predictor variables
predictor_vars <- c(
  "SeatingCapacity", "SeatingRowsTypical",
  "Deck2", "UpperDeck", "LowerDeckShade", "UpperDeckShade",
"FairTerritory",
  "FoulTerritory", "FenceHeightLF", "FenceHeightCF", "FenceHeightRF",
"Backstop", "LeftField", "LeftCenter", "CenterField",
  "RightCenter", "RightField"
)

# Ensure that all predictor variables and OPS_diff are numeric
final_player_data <- final_player_data %>%
  mutate(across(all_of(c("OPS_diff", predictor_vars)), as.numeric))

# Check for missing values
summary(final_player_data)
```

```
# Remove rows with missing values
final_player_data <- final_player_data %>%
  filter(complete.cases(.[, c("OPS_diff", predictor_vars)]))

# Standardize predictor variables
final_player_data <- final_player_data %>%
  mutate(across(all_of(predictor_vars), ~ (.-mean(.))/sd(.)))

# Preview the prepared data
head(final_player_data)
```

```{r}
colnames(final_player_data)
```

```{r}
# Prepare data list for JAGS
jags_data <- list(
  N = nrow(final_player_data),
  OPS_diff = final_player_data$OPS_diff,
  X = as.matrix(final_player_data[, predictor_vars]),
  P = length(predictor_vars)
)

library(rjags)

model_code <- "
model {
  for (i in 1:N) {
    # Likelihood
    OPS_diff[i] ~ dt(mu[i], tau, nu)
    mu[i] <- beta0 + inprod(beta[1:P], X[i,])
  }

  # Priors for coefficients
  beta0 ~ dnorm(0, 0.01)  # Precision is 1/variance
  for (j in 1:P) {
    beta[j] ~ dnorm(0, 0.01)
  }
```

```
  # Prior for scale parameter (sigma)
  tau <- pow(sigma, -2)
  sigma ~ dt(0, pow(2.5, -2), 1) T(0,)  # Half-Cauchy prior

  # Prior for degrees of freedom (nu)
  nu ~ dexp(1/30) T(2, 100)  # Constrained to be >2
}
"

```

```{r}
# Function to generate initial values
initialize_jags <- function() {
  list(
    beta0 = rnorm(1, 0, 1),
    beta = rnorm(jags_data$P, 0, 1),
    sigma = runif(1, 0.1, 5),
    nu = runif(1, 2, 100)
  )
}

```

```{r}
# Create JAGS model
jags_model <- jags.model(
  textConnection(model_code),
  data = jags_data,
  inits = initialize_jags,
  n.chains = 3,
  n.adapt = 1000
)

# Burn-in
update(jags_model, n.iter = 1000)

# MCMC sampling
mcmc_samples <- coda.samples(
  model = jags_model,
  variable.names = c("beta0", "beta", "sigma", "nu"),
  n.iter = 5000,
```

```
  thin = 5
)

```

```{r}
install.packages("ggmcmc")
```

```{r}
library(coda)
par(mfrow = c(3, 3))

# Trace plots
traceplot(mcmc_samples)

# Autocorrelation plots
autocorr.plot(mcmc_samples)

# Gelman-Rubin diagnostic
gelman_results <- gelman.diag(mcmc_samples, multivariate = FALSE)
print("gelman results")
print(gelman_results)

# Geweke diagnostic
geweke_results <- geweke.diag(mcmc_samples)
print("geweke results")
print(geweke_results)



library(knitr)

# Suppose gelman_results and geweke_results are your diagnostic
results
# Gelman-Rubin results often come as a list with an element named
$psrf
kable(gelman_results$psrf, caption = "Gelman-Rubin Diagnostic
Results")

# Geweke results may be a list of values per parameter
# If geweke_results is structured as a named vector or matrix:
```

```
kable(as.data.frame(geweke_results[[1]]), caption = "Geweke
Diagnostic Results")
```

```{r}
# Summarize posterior samples
posterior_summary <- summary(mcmc_samples)
print(posterior_summary)

# Extract posterior means and credible intervals
posterior_means <- posterior_summary$statistics[, "Mean"]
posterior_sds <- posterior_summary$statistics[, "SD"]
posterior_cis <- posterior_summary$quantiles[, c("2.5%", "97.5%")]

# Create a summary table
results_table <- data.frame(
  Parameter = rownames(posterior_summary$statistics),
  Mean = posterior_means,
  SD = posterior_sds,
  `2.5%` = posterior_cis[, "2.5%"],
  `97.5%` = posterior_cis[, "97.5%"]
)
print(results_table)

```

```{r}
# Convert mcmc_samples to a matrix and then to a data frame
mcmc_mat <- as.matrix(mcmc_samples)
mcmc_df <- as.data.frame(mcmc_mat)

# Rename columns by removing brackets
colnames(mcmc_df) <- gsub("\\[|\\]", "", colnames(mcmc_df))

# List of parameters to plot
params <- colnames(mcmc_df)

# Plot posterior densities using a loop
for (param in params) {
  plot <- ggplot(mcmc_df, aes(x = .data[[param]])) +
    geom_density(fill = "blue", alpha = 0.5) +
```

```
    labs(
      title = paste("Posterior Density of", param),
      x = param,
      y = "Density"
    ) +
    theme_minimal()

  print(plot)

  # Optionally save the plot
  # ggsave(filename = paste0("posterior_", param, ".png"), plot =
plot, width = 8, height = 6)
}

```


Sensitivity Analysis

```{r}
# Define alternative model code with tighter priors
model_code_tighter_priors <- "
model {
  for (i in 1:N) {
    # Likelihood
    OPS_diff[i] ~ dt(mu[i], tau, nu)
    mu[i] <- beta0 + inprod(beta[1:P], X[i,])
  }

  # Priors for coefficients (tighter)
  beta0 ~ dnorm(0, 0.25)    # Variance = 4 (Precision = 1/4)
  for (j in 1:P) {
    beta[j] ~ dnorm(0, 4)    # Variance = 0.25 (Precision = 4)
  }

  # Prior for scale parameter (sigma) - Half-Cauchy
  tau <- pow(sigma, -2)
  sigma ~ dt(0, pow(2.5, -2), 1) T(0,)  # Half-Cauchy prior

  # Prior for degrees of freedom (nu)
  nu ~ dexp(1/30) T(2, 100)           # Constrained to be >2
}
```

```
"

# Define alternative model code with wider priors
model_code_wider_priors <- "
model {
  for (i in 1:N) {
    # Likelihood
    OPS_diff[i] ~ dt(mu[i], tau, nu)
    mu[i] <- beta0 + inprod(beta[1:P], X[i,])
  }

  # Priors for coefficients (wider)
  beta0 ~ dnorm(0, 0.04)    # Variance = 25 (Precision = 1/25)
  for (j in 1:P) {
    beta[j] ~ dnorm(0, 1)    # Variance = 1 (Precision = 1)
  }

  # Prior for scale parameter (sigma) - Half-Cauchy
  tau <- pow(sigma, -2)
  sigma ~ dt(0, pow(2.5, -2), 1) T(0,)  # Half-Cauchy prior

  # Prior for degrees of freedom (nu)
  nu ~ dexp(1/30) T(2, 100)           # Constrained to be >2
}
"
# Prepare data list for JAGS (assuming final_player_data is already
prepared)
jags_data <- list(
  N = nrow(final_player_data),
  OPS_diff = final_player_data$OPS_diff,
  X = as.matrix(final_player_data[, predictor_vars]),
  P = length(predictor_vars)
)

# Function to generate initial values
initialize_jags <- function() {
  list(
    beta0 = rnorm(1, 0, 1),
    beta = rnorm(jags_data$P, 0, 1),
    sigma = runif(1, 0.1, 5),
    nu = runif(1, 2, 100)
  )
```

```
}

# Fit the original model
jags_model_original <- jags.model(
  textConnection(model_code),
  data = jags_data,
  inits = initialize_jags(),
  n.chains = 3,
  n.adapt = 1000
)

# Burn-in
update(jags_model_original, n.iter = 1000)

# MCMC sampling
mcmc_original <- coda.samples(
  model = jags_model_original,
  variable.names = c("beta0", "beta", "sigma", "nu"),
  n.iter = 5000,
  thin = 5
)

# Fit the model with tighter priors
jags_model_tighter <- jags.model(
  textConnection(model_code_tighter_priors),
  data = jags_data,
  inits = initialize_jags(),
  n.chains = 3,
  n.adapt = 1000
)

# Burn-in
update(jags_model_tighter, n.iter = 1000)

# MCMC sampling
mcmc_tighter <- coda.samples(
  model = jags_model_tighter,
  variable.names = c("beta0", "beta", "sigma", "nu"),
  n.iter = 5000,
  thin = 5
)
```

```
# Fit the model with wider priors
jags_model_wider <- jags.model(
  textConnection(model_code_wider_priors),
  data = jags_data,
  inits = initialize_jags(),
  n.chains = 3,
  n.adapt = 1000
)

# Burn-in
update(jags_model_wider, n.iter = 1000)

# MCMC sampling
mcmc_wider <- coda.samples(
  model = jags_model_wider,
  variable.names = c("beta0", "beta", "sigma", "nu"),
  n.iter = 5000,
  thin = 5
)


```

```{r}
# Summarize posterior samples for all models
summary_original <- summary(mcmc_original)
summary_tighter <- summary(mcmc_tighter)
summary_wider <- summary(mcmc_wider)

# Create a comparison table for prior sensitivity
compare_priors <- data.frame(
  Parameter = rownames(summary_original$statistics),
  Original_Mean = summary_original$statistics[, "Mean"],
  Original_SD = summary_original$statistics[, "SD"],
  Tighter_Mean = summary_tighter$statistics[, "Mean"],
  Tighter_SD = summary_tighter$statistics[, "SD"],
  Wider_Mean = summary_wider$statistics[, "Mean"],
  Wider_SD = summary_wider$statistics[, "SD"]
)

print("Comparison of Posterior Estimates Across Different Priors:")
print(compare_priors)
```

```
# Combine all MCMC samples into a single data frame with model
identifiers
mcmc_original_df <- as.data.frame(as.matrix(mcmc_original))
mcmc_original_df$Model <- "Original"

mcmc_tighter_df <- as.data.frame(as.matrix(mcmc_tighter))
mcmc_tighter_df$Model <- "Tighter Priors"

mcmc_wider_df <- as.data.frame(as.matrix(mcmc_wider))
mcmc_wider_df$Model <- "Wider Priors"

# Rename columns to remove brackets
colnames(mcmc_original_df) <- gsub("\\[|\\]", "",
colnames(mcmc_original_df))
colnames(mcmc_tighter_df) <- gsub("\\[|\\]", "",
colnames(mcmc_tighter_df))
colnames(mcmc_wider_df) <- gsub("\\[|\\]", "",
colnames(mcmc_wider_df))

# Combine data frames
combined_priors <- bind_rows(mcmc_original_df, mcmc_tighter_df,
mcmc_wider_df)
combined_models <- bind_rows(combined_priors)

# Reshape data for ggplot2
library(tidyr)
plot_params <- c("beta0", "betacoef1", "betacoef2",
"betacoef3","betacoef4","betacoef5","betacoef6","betacoef7","betacoef
8","betacoef9","betacoef10","betacoef11","betacoef12","betacoef13","b
etacoef14","betacoef15","betacoef16","betacoef17","alpha")

mcmc_long <- combined_models %>%
  pivot_longer(cols = c("beta0", "beta1",
"beta2","beta3","beta4","beta5","beta6","beta7","beta8","beta9","beta
10","beta11","beta12","beta13","beta14","beta15","beta16","beta17",
"sigma", "nu"),
               names_to = "Parameter",
               values_to = "Value")

# Plot density plots with faceting
```

```
ggplot(mcmc_long, aes(x = Value, fill = Model, color = Model)) +
  geom_density(alpha = 0.3) +
  facet_wrap(~ Parameter, scales = "free") +
  labs(
    title = "Posterior Density Comparison Across Models",
    x = "Parameter Value",
    y = "Density"
  ) +
  theme_minimal()

```
```{r}
kable(compare_priors)
```

```{r}
# First, we'll need to shift the OPS_diff data to ensure positivity
# Find minimum value and add offset to make all values positive
min_ops <- min(final_player_data$OPS_diff)
offset <- abs(min_ops) + 0.1  # Adding 0.1 to ensure strictly
positive
shifted_ops <- final_player_data$OPS_diff + offset

# Define the correct predictor variables
predictor_vars <- c(
  "SeatingCapacity", "SeatingRowsTypical",
  "Deck2", "UpperDeck", "LowerDeckShade", "UpperDeckShade",
"FairTerritory",
  "FoulTerritory", "FenceHeightLF", "FenceHeightCF", "FenceHeightRF",
"Backstop",
  "LeftField", "LeftCenter", "CenterField", "RightCenter",
"RightField"
)

min_ops <- min(final_player_data$OPS_diff)
offset <- abs(min_ops) + 0.1  # ensure strictly positive
shifted_ops <- final_player_data$OPS_diff + offset

jags_data_gamma <- list(
  N = nrow(final_player_data),
  OPS_diff_shifted = shifted_ops,
  X = as.matrix(final_player_data[, predictor_vars]),
```

```
  P = length(predictor_vars),
  offset = offset
)

### Original (Reference) Model Code (Informed priors) ###
model_code_original <- "
model {
  alpha ~ dgamma(2, 0.5)
  for (i in 1:N) {
    OPS_diff_shifted[i] ~ dgamma(alpha, beta[i])
    log(beta[i]) <- beta0 + inprod(betacoef[1:P], X[i,])
  }

  # Intercept
  beta0 ~ dnorm(0, 0.1)

  # Informed priors as per your original code
  betacoef[1] ~ dnorm(-0.1, 4)
  betacoef[2] ~ dnorm(-0.05, 16)
  betacoef[3] ~ dnorm(-0.1, 4)
  betacoef[4] ~ dnorm(-0.15, 4)
  betacoef[5] ~ dnorm(-0.2, 9)
  betacoef[6] ~ dnorm(-0.2, 9)
  betacoef[7] ~ dnorm(0.1, 4)
  betacoef[8] ~ dnorm(0.1, 4)
  betacoef[9] ~ dnorm(-0.15, 4)
  betacoef[10] ~ dnorm(-0.2, 4)
  betacoef[11] ~ dnorm(-0.15, 4)
  betacoef[12] ~ dnorm(-0.05, 16)
  betacoef[13] ~ dnorm(-0.1, 4)
  betacoef[14] ~ dnorm(-0.1, 4)
  betacoef[15] ~ dnorm(-0.15, 4)
  betacoef[16] ~ dnorm(-0.1, 4)
  betacoef[17] ~ dnorm(-0.1, 4)
}
"

### Less Informative (Wider) Priors ###
model_code_wider <- "
model {
  alpha ~ dgamma(2, 0.5)
  for (i in 1:N) {
```

```
    OPS_diff_shifted[i] ~ dgamma(alpha, beta[i])
    log(beta[i]) <- beta0 + inprod(betacoef[1:P], X[i,])
  }

  beta0 ~ dnorm(0, 0.01)    # Var = 100
  for (j in 1:P) {
    betacoef[j] ~ dnorm(0, 0.01) # Var = 100
  }
}
"

### More Informative (Tighter) Priors ###
model_code_tighter <- "
model {
  alpha ~ dgamma(2, 0.5)
  for (i in 1:N) {
    OPS_diff_shifted[i] ~ dgamma(alpha, beta[i])
    log(beta[i]) <- beta0 + inprod(betacoef[1:P], X[i,])
  }

  beta0 ~ dnorm(0, 1)     # Var = 1
  for (j in 1:P) {
    betacoef[j] ~ dnorm(0, 10) # Var = 0.1
  }
}
"

### Very Uninformative Priors ###
# Here we use extremely flat priors, e.g. dnorm(0, 0.0001) ~ Normal
with Var = 10,000
# This is likely so broad that the data should dominate entirely.
model_code_very_uninform <- "
model {
  alpha ~ dgamma(2, 0.5)
  for (i in 1:N) {
    OPS_diff_shifted[i] ~ dgamma(alpha, beta[i])
    log(beta[i]) <- beta0 + inprod(betacoef[1:P], X[i,])
  }

  beta0 ~ dnorm(0, 0.0001)   # Var = 10,000
  for (j in 1:P) {
    betacoef[j] ~ dnorm(0, 0.0001) # Var = 10,000
```

```
  }
}
"

# Initialization function
initialize_jags_gamma <- function() {
  list(
    beta0 = rnorm(1, 0, 1),
    betacoef = rnorm(jags_data_gamma$P, 0, 1),
    alpha = rgamma(1, 2, 0.5)
  )
}

# Fit original model
jags_model_original <- jags.model(
  textConnection(model_code_original),
  data = jags_data_gamma,
  inits = initialize_jags_gamma,
  n.chains = 3,
  n.adapt = 1000
)
update(jags_model_original, n.iter = 1000)
mcmc_original <- coda.samples(
  model = jags_model_original,
  variable.names = c("beta0", "betacoef", "alpha"),
  n.iter = 5000,
  thin = 5
)

# Fit wider model
jags_model_wider <- jags.model(
  textConnection(model_code_wider),
  data = jags_data_gamma,
  inits = initialize_jags_gamma,
  n.chains = 3,
  n.adapt = 1000
)
update(jags_model_wider, n.iter = 1000)
mcmc_wider <- coda.samples(
  model = jags_model_wider,
  variable.names = c("beta0", "betacoef", "alpha"),
  n.iter = 5000,
```

```
  thin = 5
)

# Fit tighter model
jags_model_tighter <- jags.model(
  textConnection(model_code_tighter),
  data = jags_data_gamma,
  inits = initialize_jags_gamma,
  n.chains = 3,
  n.adapt = 1000
)
update(jags_model_tighter, n.iter = 1000)
mcmc_tighter <- coda.samples(
  model = jags_model_tighter,
  variable.names = c("beta0", "betacoef", "alpha"),
  n.iter = 5000,
  thin = 5
)

# Fit very uninformative model
jags_model_very_uninform <- jags.model(
  textConnection(model_code_very_uninform),
  data = jags_data_gamma,
  inits = initialize_jags_gamma,
  n.chains = 3,
  n.adapt = 1000
)
update(jags_model_very_uninform, n.iter = 1000)
mcmc_very_uninform <- coda.samples(
  model = jags_model_very_uninform,
  variable.names = c("beta0", "betacoef", "alpha"),
  n.iter = 5000,
  thin = 5
)

# Summaries
summary_original <- summary(mcmc_original)
summary_wider <- summary(mcmc_wider)
summary_tighter <- summary(mcmc_tighter)
summary_very_uninform <- summary(mcmc_very_uninform)

# Create comparison table
```

```
parameter_names <- c("beta0", paste0("betacoef[",
1:jags_data_gamma$P, "]"), "alpha")

compare_priors <- data.frame(
  Parameter = parameter_names,
  Original_Mean = summary_original$statistics[,"Mean"],
  Original_SD = summary_original$statistics[,"SD"],
  Wider_Mean = summary_wider$statistics[,"Mean"],
  Wider_SD = summary_wider$statistics[,"SD"],
  Tighter_Mean = summary_tighter$statistics[,"Mean"],
  Tighter_SD = summary_tighter$statistics[,"SD"],
  VeryUninform_Mean = summary_very_uninform$statistics[,"Mean"],
  VeryUninform_SD = summary_very_uninform$statistics[,"SD"]
)

print("Comparison of Posterior Estimates Across Different Priors:")
print(compare_priors)

# Combine MCMC samples into a single data frame
mcmc_original_df <- as.data.frame(as.matrix(mcmc_original))
mcmc_original_df$Model <- "Original"

mcmc_wider_df <- as.data.frame(as.matrix(mcmc_wider))
mcmc_wider_df$Model <- "Wider"

mcmc_tighter_df <- as.data.frame(as.matrix(mcmc_tighter))
mcmc_tighter_df$Model <- "Tighter"

mcmc_very_uninform_df <- as.data.frame(as.matrix(mcmc_very_uninform))
mcmc_very_uninform_df$Model <- "Very Uninformative"

# Clean column names
colnames(mcmc_original_df) <- gsub("\\[|\\]", "",
colnames(mcmc_original_df))
colnames(mcmc_wider_df) <- gsub("\\[|\\]", "",
colnames(mcmc_wider_df))
colnames(mcmc_tighter_df) <- gsub("\\[|\\]", "",
colnames(mcmc_tighter_df))
colnames(mcmc_very_uninform_df) <- gsub("\\[|\\]", "",
colnames(mcmc_very_uninform_df))

# Combine all MCMC samples
```

```
combined <- bind_rows(mcmc_original_df, mcmc_wider_df,
mcmc_tighter_df, mcmc_very_uninform_df)


```

```{r}
# Select some parameters to plot
plot_params <- c("beta0", "betacoef1", "betacoef2",
"betacoef3","betacoef4","betacoef5","betacoef6","betacoef7","betacoef
8","betacoef9","betacoef10","betacoef11","betacoef12","betacoef13","b
etacoef14","betacoef15","betacoef16","betacoef17","alpha")

mcmc_long <- combined %>%
  pivot_longer(cols = plot_params, names_to = "Parameter", values_to
= "Value")

# Plot the densities
ggplot(mcmc_long, aes(x = Value, fill = Model, color = Model)) +
  geom_density(alpha = 0.3) +
  facet_wrap(~ Parameter, scales = "free") +
  labs(
    title = "Posterior Density Comparison Across Different Priors
(Gamma Model)",
    x = "Parameter Value",
    y = "Density"
  ) +
  theme_minimal()

kable(compare_priors)

```

```{r}
par(mfrow = c(3, 3))
# Trace plots
traceplot(mcmc_samples_informed)

# Autocorrelation plots
autocorr.plot(mcmc_samples_informed)

# Gelman-Rubin diagnostic
```

```
gelman_results <- gelman.diag(mcmc_samples_informed, multivariate =
FALSE)
print("gelman results")
kable(gelman_results$psrf, caption = "Gelman-Rubin Diagnostic
Results")

# Geweke diagnostic
geweke_results <- geweke.diag(mcmc_samples_informed)
print("geweke results")
print(geweke_results)
```

Code for Question 2:

````
---
title: "STAT425FINALPROJECT"
author: "Aditya Daga"
format: pdf
editor:
  markdown:
    wrap: 72
---

#1

```{r}
library(dplyr)
library(readr)
library(ggplot2)

file_paths <-
c("/Users/adityadaga/Downloads/adity/Downloads/batting_2017.csv",
"/Users/adityadaga/Downloads/adity/Downloads/batting_2018.csv",
"/Users/adityadaga/Downloads/adity/Downloads/batting_2019.csv",
"/Users/adityadaga/Downloads/adity/Downloads/batting_2020.csv")

combined_data <- file_paths %>%
  lapply(read_csv) %>%
  bind_rows()

# Preview the combined dataframe
head(combined_data)

```

The question:

Does a team’s home run rate change significantly when playing at
their
home stadium versus away stadiums?

#3

Response Variable:
````

- Team.R (Total runs scored by the batter's team in the game).

Predictor Variables:

- H.A: Home or Away (binary: 0 for Away, 1 for Home).

- Opponent.Strength: Standardized Moneyline column to measure how favored or underdog the team was

    - Opponent Strength = (Moneyline - mean(Moneyline))/sd(Moneyline)

    - Lower values (e.g., negative moneyline) imply stronger teams (favored)

    - Higher values (e.g., positive moneyline) imply weaker teams (underdogs).

We will choose the Negative Binomial distribution for the Bayesian model because it is well-suited for the type of data and the underlying characteristics of the problem

The outcome variable, Team.R (total runs scored), is a count variable that can take on non-negative integer values (0, 1, 2, ...). This makes the Poisson family of distributions (which models count data) a natural choice. However, the Poisson distribution assumes the mean equals the variance, which may not hold in real-world data, particularly in baseball.

In baseball data, there is often overdispersion, meaning that the variance of the count data is greater than the mean. This happens because team performance can vary widely based on factors such as player skill, park effects, and opponent strength. Also, external conditions, such as weather or game strategy, may also introduce variability.

The expected number of runs is modeled on the log scale as:

```
$$
\log(\mu_i) = \beta_0 + \beta_1 \cdot H.A_i + \beta_2 \cdot
\text{Opponent.Strength}_i
$$

Where B0 is Baseline log-run rate for Away games, B1 is the
additional
effect of playing at Home, and B2 is the effect of Opponent Strength
(positive/negative impact on run rate).

Our assumption is that home advantage increases the run rate (a
positive
B1) and that stronger opponents supress the run rate (a negative B2)

Additionally, our likelihood is given by:

$$
\text{Team.R}_i \sim \text{NegativeBinomial}(\mu_i, \phi)
$$

Where:

$$
\mu_i = e^{\log(\mu_i)}: \text{Mean runs for team } i.
$$

And:

$$
\phi: \text{Dispersion parameter to model overdispersion.}
$$

#4

Let's begin verifying our model assumptions:

```{r}
# Ensure Moneyline is numeric
combined_data <- combined_data %>%
  mutate(Moneyline = as.numeric(as.character(Moneyline)))
```
```

```
# Check if there are any NAs introduced during conversion
if (any(is.na(combined_data$Moneyline))) {
  warning("Some Moneyline values could not be converted to numeric.
Please check for invalid entries.")
}

# Standardize Moneyline to create Opponent.Strength predictor
combined_data <- combined_data %>%
  mutate(Opponent.Strength = (Moneyline - mean(Moneyline, na.rm =
TRUE)) / sd(Moneyline, na.rm = TRUE))
```

We have standarized the Moneyline column above.

Since the Negative Binomial distribution assumes count data with overdispersion, it is important to verify if Team.R matches this pattern.

```{r}
# Plot the distribution of Team.R
ggplot(combined_data, aes(x = Team.R)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "white", alpha
= 0.7) +
  labs(
    title = "Distribution of Team Runs Scored (Team.R)",
    x = "Total Runs Scored",
    y = "Frequency"
  ) +
  theme_minimal()
```

While the Poisson distribution is often used for count data, the observed variance in the data (evident from the long tail) suggests overdispersion (variance exceeding the mean). This makes the Negative Binomial distribution a better fit, as it can accommodate overdispersion
by introducing an additional parameter for variance.

Next, we will evaluate the relationship Between Predictors and Response.
We will examine whether H.A (Home vs. Away) and Opponent.Strength are related to Team.R. We will use boxplots and scatterplots to visualize these relationships.

```{r}
# Boxplot of Team.R by Home/Away
ggplot(combined_data, aes(x = H.A, y = Team.R, fill = H.A)) +
  geom_boxplot(alpha = 0.7, outlier.color = "red") +
  labs(
    title = "Runs Scored (Team.R) by Home/Away",
    x = "Home or Away",
    y = "Total Runs Scored"
  ) +
  theme_minimal() +
  scale_fill_manual(values = c("blue", "green"))

```

```{r}
# Scatterplot of Team.R vs Opponent.Strength
ggplot(combined_data, aes(x = Opponent.Strength, y = Team.R)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(
    title = "Relationship Between Opponent Strength and Runs Scored",
    x = "Opponent Strength (Standardized Moneyline)",
    y = "Total Runs Scored"
  ) +
  theme_minimal()

```

The boxplot of Team.R (Total Runs Scored) grouped by H.A (Home vs. Away)
reveals that there is a slight difference in the median runs scored between home and away games, with teams generally scoring marginally

more runs at home. This supports the inclusion of H.A as a predictor in
the model, as it suggests a potential relationship between playing location and scoring outcomes. The scatterplot of Team.R against Opponent.Strength (standardized Moneyline) shows a slight negative trend, indicating that as the opponent's strength increases (lower Moneyline values), the runs scored by the team decrease. This aligns with the hypothesis that stronger opponents suppress team performance,
justifying the inclusion of Opponent.Strength as a predictor in the model. Together, these visualizations provide preliminary evidence supporting the relevance of the chosen predictors and the assumptions underlying the model structure.

Lastly, we will assess overdispersion in the Response Variable by comparing the variance and mean of Team.R. Oversdispertion is a part of
our assumption of the Negative Binomial model. Additionally, we will plot the mean-variance relationship.

```{r}
# Calculate mean and variance of Team.R
mean_var_summary <- combined_data %>%
  group_by(H.A) %>%
  summarise(
    Mean = mean(Team.R, na.rm = TRUE),
    Variance = var(Team.R, na.rm = TRUE)
  )

# Print mean and variance summary
print(mean_var_summary)

```

The mean-variance relationship of the Team.R variable for home (H) and
away (A) games shows that the variance is greater than the mean in both
cases, with a variance of 10.805 for away games and 10.534 for home games, compared to means of 4.660 and 4.744, respectively. This observation indicates overdispersion, where the variability in the data

exceeds what would be expected under a Poisson distribution, which assumes equality of mean and variance. The negative binomial model accounts for this overdispersion by introducing an additional parameter
to model variability beyond the mean. Therefore, the negative binomial
model is an appropriate choice for modeling the total runs scored (Team.R), as it better captures the observed variability in the data.

#5

We will set priors for the coefficients (B0, B1, B2) and the overdispersion parameter (α) based on the context of our data. Here's the plan:

- Baseline Log-Run Rate (B0)
    - We assume the baseline log-run rate to have a weakly informative
      prior, as we do not have strong prior knowledge about the average runs scored per game.
    - Prior: B0~N(0,2^2)(mean = 0, standard deviation = 2)
- Effect of Playing at Home (B1)
    - The effect of playing at home is expected to be small but positive, as home-field advantage in sports is generally small.
    - Prior: B1~N(0,0.5^2)(mean = 0, standard deviation = 0.5)
- Effect of Opponent Strength (B2):
    - Opponent strength is standardized, and we assume a weak prior centered around zero since its effect on runs scored is uncertain
    - Prior: B2~N(0,1^2)(mean = 0, standard deviation = 1)
- Overdispersion Parameter (α):

- The overdispersion parameter is constrained to be positive and reflects the variability in the run count. A weakly informative prior will be used to allow flexibility.

- Prior: α~Gamma(2,0.5) (shape = 2, rate = 0.5), allowing a broad range of values with higher likelihood near small positive values.

We will plot the prior densities for each parameter to confirm that they
align with our expectations and assumptions about the model.

```{r}
library(ggplot2)

# Priors for coefficients
beta_prior <- data.frame(
  beta0 = rnorm(10000, mean = 0, sd = 2),
  beta1 = rnorm(10000, mean = 0, sd = 0.5),
  beta2 = rnorm(10000, mean = 0, sd = 1)
)

ggplot(beta_prior, aes(x = beta0)) +
  geom_density(color = "blue", fill = "blue", alpha = 0.3) +
  labs(title = "Prior for β0 (Baseline Log-Run Rate)", x = "β0", y = "Density")

ggplot(beta_prior, aes(x = beta1)) +
  geom_density(color = "green", fill = "green", alpha = 0.3) +
  labs(title = "Prior for β1 (Effect of Playing at Home)", x = "β1", y = "Density")

ggplot(beta_prior, aes(x = beta2)) +
  geom_density(color = "purple", fill = "purple", alpha = 0.3) +
  labs(title = "Prior for β2 (Effect of Opponent Strength)", x = "β2", y = "Density")

# Prior for alpha
```

```
alpha_prior <- data.frame(alpha = rgamma(10000, shape = 2, rate =
0.5))

ggplot(alpha_prior, aes(x = alpha)) +
  geom_density(color = "red", fill = "red", alpha = 0.3) +
  labs(title = "Prior for α (Overdispersion)", x = "α", y =
"Density")
```

```
#6

Plan for posterior inference:

We plan to implement posterior inference using Markov Chain Monte
Carlo
methods, specifically utilizing Gibbs sampling through the rjags
package
in R. Our goal will be to estimate the posterior distributions of the
model parameters: the intercept (beta0), the effect of playing at
home
versus away (beta1), the effect of opponent strength (beta2), and the
overdispersion parameter (alpha) for a negative binomial regression
model. By using the MCMC algorithm, we will approximate the joint
posterior distributions of these parameters by generating samples
from
their probability distributions conditional on the observed data.

To ensure efficient and reliable posterior inference, we will use
three
chains with random initial values and an adaptive burn-in phase to
allow
the chains to reach their stationary distribution. After the burn-in,
we
will run the MCMC chains for 3,000 iterations per chain with a
thinning
interval of 5 to reduce autocorrelation between consecutive samples.
To
diagnose convergence and ensure the accuracy of our posterior
estimates,
we will examine trace plots, autocorrelation plots, and use
diagnostic
```

tests, such as the Geweke diagnostic and the Gelman-Rubin diagnostic. These steps will help us verify that the chains have converged and provide robust posterior estimates.

#7

```{r}
library(rjags)
library(coda)
library(ggplot2)
library(dplyr)

# Filter out rows with NA in Opponent_Strength
filtered_data <- combined_data %>%
  filter(!is.na(Opponent.Strength))

# Prepare the data for JAGS
jags_data <- list(
  Team_R = filtered_data$Team.R,
  Home_Away = as.numeric(filtered_data$H.A == "H"),
  Opponent_Strength = filtered_data$Opponent.Strength,
  N = nrow(filtered_data) # Number of observations
)

# Define the JAGS model with updated priors
model_code <- "
model {
  for (i in 1:N) {
    Team_R[i] ~ dnegbin(prob[i], alpha)
    prob[i] <- alpha / (alpha + mu[i])
    log(mu[i]) <- beta0 + beta1 * Home_Away[i] + beta2 *
Opponent_Strength[i]
  }

  # Updated priors for the regression coefficients
  beta0 ~ dnorm(0, 0.25)    # Weakly informative prior: mean = 0, sd
= 2
  beta1 ~ dnorm(0, 4)       # Weakly informative prior: mean = 0, sd
= 0.5
  beta2 ~ dnorm(0, 1)       # Weakly informative prior: mean = 0, sd
= 1
```

```
  # Updated prior for the overdispersion parameter
  alpha ~ dgamma(2, 0.5)    # Weakly informative Gamma prior: shape =
2, rate = 0.5
}
"

# Initialize the parameters
initialize_jags <- function() {
  list(
    beta0 = rnorm(1, 0, 1),
    beta1 = rnorm(1, 0, 0.2),
    beta2 = rnorm(1, 0, 0.5),
    alpha = rgamma(1, 2, 0.5)
  )
}

# Run the JAGS model
jags_model <- jags.model(
  textConnection(model_code),
  data = jags_data,
  inits = initialize_jags,
  n.chains = 3,      # Use 3 chains for better diagnostics
  n.adapt = 500      # Reduced adaptation phase for speed
)

# Burn-in phase
update(jags_model, 500)  # Reduced burn-in to 500 iterations

# Sampling from the posterior
mcmc_samples <- coda.samples(
  jags_model,
  variable.names = c("beta0", "beta1", "beta2", "alpha"),
  n.iter = 3000,     # Further reduced iterations
  thin = 5           # Increased thinning to reduce autocorrelation
)

# Plot trace plots to assess convergence
traceplot(mcmc_samples)

# Plot autocorrelation for each parameter
autocorr.plot(mcmc_samples)
```

```
# Geweke diagnostic for convergence
geweke_results <- geweke.diag(mcmc_samples)
print(geweke_results)

# Gelman-Rubin diagnostic for convergence
gelman_results <- gelman.diag(mcmc_samples, multivariate = FALSE)
print(gelman_results)

# Summarize the posterior samples
summary(mcmc_samples)

# Optional: Create ggplot-based trace plots for visual inspection
mcmc_df <- as.data.frame(as.matrix(mcmc_samples))
mcmc_df$Iteration <- 1:nrow(mcmc_df)

ggplot(mcmc_df, aes(x = Iteration)) +
  geom_line(aes(y = beta0, color = "beta0")) +
  geom_line(aes(y = beta1, color = "beta1")) +
  geom_line(aes(y = beta2, color = "beta2")) +
  geom_line(aes(y = alpha, color = "alpha")) +
  labs(
    title = "Trace Plots for MCMC Chains",
    x = "Iteration",
    y = "Parameter Value",
    color = "Parameter"
  ) +
  theme_minimal()


```

The trace plots for the parameters alpha, beta0, beta1, and beta2 show
good mixing, with the chains exploring the parameter space without significant trends or drifts. This indicates that the MCMC sampling is
converging, and the chains have likely reached their stationary distribution. The absence of distinct upward or downward trends in the
traces further supports convergence.

The autocorrelation plots reveal that all the parameters exhibit minimal autocorrelation across lags. This suggests that the chains are sampling effectively with reduced dependence between successive iterations. Lower autocorrelation is desirable as it means the samples are more independent, leading to better posterior estimation.

Geweke Diagnostic: The Geweke diagnostic results for the parameters show that the z-scores are within an acceptable range (close to zero), indicating no significant difference between the means of the first and last portions of the chains. This further supports the conclusion that the chains have converged.

alpha: z-scores range around ±0.5
beta0: Mixed but within reasonable bounds
beta1 and beta2: Also within acceptable ranges. The Geweke diagnostic does not suggest any issues with convergence.

The Gelman-Rubin diagnostic shows that all parameters have Potential Scale Reduction Factor values close to 1:

alpha: Point estimate = 1.00, Upper CI = 1.01 beta0: Point estimate = 1.01, Upper CI = 1.02 beta1: Point estimate = 1.01, Upper CI = 1.03 beta2: Point estimate = 1.00, Upper CI = 1.02

These values indicate that the chains have converged across the three chains, with no significant variance between them. A PSRF value close to 1 signifies good convergence.

The summary statistics show small standard errors, narrow credible intervals, and consistent means across chains. For example:

alpha: Mean = 3.70674, 2.5% quantile = 3.6556, 97.5% quantile = 3.75785.
beta0: Mean = 1.54287, 2.5% quantile = 1.5374, 97.5% quantile = 1.54799.

```
beta1: Mean = -0.04024, 2.5% quantile = -0.0475, 97.5% quantile =
-0.03242.
beta2: Mean = -0.10238, 2.5% quantile = -0.1063, 97.5%
quantile = -0.09840.

This shows that the posterior distributions are well-defined with no
apparent anomalies.

#8

```{r}
# Load necessary library
library(ggplot2)

# Summarize posterior distributions
posterior_summary <- summary(mcmc_samples)

# Extract posterior means, standard deviations, and credible
intervals
posterior_means <- posterior_summary$statistics[, "Mean"]
posterior_sd <- posterior_summary$statistics[, "SD"]
posterior_ci <- apply(as.matrix(mcmc_samples), 2, function(x)
quantile(x, probs = c(0.025, 0.975)))

# Print summary statistics
print("Posterior Means:")
print(posterior_means)
print("Posterior Standard Deviations:")
print(posterior_sd)
print("95% Credible Intervals:")
print(posterior_ci)

# Plot histograms of the posterior densities
posterior_df <- as.data.frame(as.matrix(mcmc_samples))
params <- colnames(posterior_df)

for (param in params) {
  plot <- ggplot(posterior_df, aes_string(x = param)) +
    geom_histogram(aes(y = ..density..), bins = 30, fill = "blue",
alpha = 0.7) +
    geom_density(color = "red", size = 1) +
    labs(
```
```

```
      title = paste("Posterior Density of", param),
      x = param,
      y = "Density"
    ) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))

  # Display the plot
  print(plot)

  # Save the plot
  ggsave(filename = paste0("posterior_", param, ".png"), plot = plot,
width = 8, height = 6)
}

# Optional: Print out detailed summaries for interpretation
for (param in params) {
  cat("Posterior summary for", param, ":\n")
  cat("  Mean:", mean(posterior_df[[param]]), "\n")
  cat("  Median:", median(posterior_df[[param]]), "\n")
  cat("  Mode (approx):",
density(posterior_df[[param]])$x[which.max(density(posterior_df[[para
m]])$y)], "\n")
  cat("  95% CI:", quantile(posterior_df[[param]], probs = c(0.025,
0.975)), "\n\n")
}


```

Alpha (Overdispersion Parameter):

The posterior density of alpha is unimodal, symmetric, and appears approximately Gaussian. The mean, median, and mode of alpha are closely aligned (mean ≈ 3.754), indicating strong agreement between central tendency measures. The 95% credible interval (3.7117, 3.7997) suggests that the variability is well constrained. Given the prior assumption of a Gamma(2, 0.5) distribution, the posterior density

concentrates in a realistic range, confirming that the data supports a moderate overdispersion in team runs.

Beta0 (Baseline Log-Run Rate for Away Games):
The posterior density for beta0 is sharply peaked, symmetric, and Gaussian-like. The mean, median, and mode are almost identical (mean ≈ 1.569), indicating stability.The 95% credible interval (1.5644, 1.5728) is narrow, suggesting high certainty in the baseline away log-run rate. Compared to the prior, the posterior concentrates near the empirical average for runs scored, aligning well with prior expectations and the data.

Beta1 (Home Effect on Log-Run Rate):
The posterior density of beta1 is negative, symmetric, and tightly concentrated (mean ≈ -0.051), with a narrow 95% credible interval (-0.0576, -0.0455).This suggests a small but statistically significant negative effect of being the home team on run rates.
The result is surprising. While prior beliefs might expect a positive beta1 (home-field advantage), this negative value may indicate that run production is relatively suppressed for the home team, potentially due to league-wide biases, park effects, or other game-level factors. This negative beta1 merits further investigation.

Beta2 (Opponent Strength Effect on Log-Run Rate):
The posterior density for beta2 is symmetric, negative, and narrow (mean ≈ -0.106), with a 95% credible interval (-0.1088, -0.1025).This indicates a small but significant negative effect of opponent strength (stronger opponents decrease the team's run rate).
This finding aligns with prior intuition that facing a stronger opponent (lower Opponent.Strength) suppresses a team's scoring performance. The tight credible interval indicates a strong signal in the data.

#9

Sensitivity Analysis

```{r}
library(rjags)
library(coda)
library(ggplot2)
```

```
library(dplyr)

# Filter out rows with NA in Opponent_Strength
filtered_data <- combined_data %>%
  filter(!is.na(Opponent.Strength))

# Prepare the data for JAGS
jags_data <- list(
  Team_R = filtered_data$Team.R,
  Home_Away = as.numeric(filtered_data$H.A == "H"),
  Opponent_Strength = filtered_data$Opponent.Strength,
  N = nrow(filtered_data) # Number of observations
)

# Define non-informative priors model
model_code_non_informative <- "
model {
  for (i in 1:N) {
    Team_R[i] ~ dnegbin(prob[i], alpha)
    prob[i] <- alpha / (alpha + mu[i])
    log(mu[i]) <- beta0 + beta1 * Home_Away[i] + beta2 *
Opponent_Strength[i]
  }

  # Non-informative priors
  beta0 ~ dnorm(0, 0.0001)
  beta1 ~ dnorm(0, 0.0001)
  beta2 ~ dnorm(0, 0.0001)
  alpha ~ dunif(0.01, 10)
}
"

# Define informed priors model
model_code_informed <- "
model {
  for (i in 1:N) {
    Team_R[i] ~ dnegbin(prob[i], alpha)
    prob[i] <- alpha / (alpha + mu[i])
    log(mu[i]) <- beta0 + beta1 * Home_Away[i] + beta2 *
Opponent_Strength[i]
  }
```

```
  # Informed priors
  beta0 ~ dnorm(1.5685, 1/(0.0021^2))  # Based on partially informed
prior mean and variance
  beta1 ~ dnorm(-0.0513, 1/(0.0031^2)) # Based on partially informed
prior mean and variance
  beta2 ~ dnorm(-0.1056, 1/(0.0016^2)) # Based on partially informed
prior mean and variance
  alpha ~ dgamma(2, 0.5)               # Based on prior information
about dispersion
}
"

# Run MCMC for a given model
run_mcmc <- function(model_code) {
  # Initialize JAGS model
  jags_model <- jags.model(
    textConnection(model_code),
    data = jags_data,
    n.chains = 3,      # Use 3 chains for better diagnostics
    n.adapt = 500      # Adaptation phase
  )

  # Burn-in phase
  update(jags_model, 500)

  # Sampling from the posterior
  mcmc_samples <- coda.samples(
    jags_model,
    variable.names = c("beta0", "beta1", "beta2", "alpha"),
    n.iter = 3000,
    thin = 5
  )

  return(mcmc_samples)
}

# Run MCMC for both models
mcmc_non_informative <- run_mcmc(model_code_non_informative)
mcmc_informed <- run_mcmc(model_code_informed)
```

```{r}
```

```
library(ggplot2)

# Convert samples to data frames for visualization
posterior_non_informative <-
as.data.frame(as.matrix(mcmc_non_informative))
posterior_informed <- as.data.frame(as.matrix(mcmc_informed))

# Combine data for plotting
posterior_non_informative$type <- "Non-Informative Prior"
posterior_informed$type <- "Informed Prior"
posterior_df$type <- "Original, partially informed Prior"
combined_posterior <- rbind(posterior_non_informative,
posterior_informed, posterior_df)

# Create comparison plots
params <- c("beta0", "beta1", "beta2", "alpha")

for (param in params) {
  plot <- ggplot(combined_posterior, aes_string(x = param, fill =
"type")) +
    geom_density(alpha = 0.5) +
    labs(
      title = paste("Posterior Comparison for", param),
      x = param,
      y = "Density"
    ) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))

  # Display the plot
  print(plot)

  # Save the plot
  ggsave(filename = paste0("sensitivity_", param, ".png"), plot =
plot, width = 8, height = 6)
}

# Print summaries for both models
cat("\nSummary for Non-Informative Priors:\n")
print(summary(mcmc_non_informative))

cat("\nSummary for Informed Priors:\n")
```

```
print(summary(mcmc_informed))
```

Code for Question 3:

````
---
title: "425 Analysis 3"
author: "Brandon Fantine"
output: pdf_document
date: "2024-12-01"
---

```{r setup, include=FALSE}

# Load necessary libraries
library(dplyr)
library(rjags)
library(knitr)
library(stringr)

# Set a seed for reproducibility
set.seed(123)

```

```{r}

# Load and clean data
hitter_data <- as.data.frame(read.csv("combineddata.csv"))

hitter_data$Pitch.Hand <- str_extract(hitter_data$Starting.Pitcher,
'([RL])')

# Select relevant variables
cleaned_data <- hitter_data %>%
  filter(!is.na(BA) & !is.na(Pitch.Hand) & !is.na(Stadium) &
           !str_detect(Stadium, '[0-9]'))

# Convert categorical variables
cleaned_data$Stadium <- as.factor(cleaned_data$Stadium)
cleaned_data$H.A <- as.factor(cleaned_data$H.A)
cleaned_data$Pitch.Hand <- as.factor(cleaned_data$Pitch.Hand)

```
````

Let us create a knitr table for brief analysis.

```{r}

knitr::kable(cleaned_data %>%
               select(Player, Stadium, Pitch.Hand, HR) %>%
               group_by(Player, Stadium, Pitch.Hand, HR) %>%
               arrange(Player, Stadium, Pitch.Hand, HR), "simple")

```

Bayesian Analysis 1: HR v. Stadiums, Normal Prior

Let us begin our Bayesian analysis by identifying the relationship between HR counts (NOT RATES) and the stadiums themselves.

```{r}

# Prepare data for JAGS
N <- nrow(cleaned_data)
stadium_levels <- as.integer(factor(cleaned_data$Stadium))
pitch_hand <- as.integer(factor(cleaned_data$Pitch.Hand))
K <- length(unique(stadium_levels))
HR <- cleaned_data$HR

```

Bayesian Analysis 2: HR v. Stadiums, Poisson Prior

That wasn't very accurate, but was a worthy practice! Since sigma wasn't incredibly far from 0, we can assume that there reasonably exists a relationship between HR and Stadiums; if sigma was 1, the relationship would be purely random (and if it was 0, it would be contradictory to our posterior predictive plot). Our endavor has worth.

Consider that the data is dispersed across the values of 0, 1, 2, 3, and 4 discretely: they are non-continuous. We should choose a prior accordingly. This directs us towards a negative-binomial distribution *or* a poisson distribution. A poisson prior assumes that the mean

and variance are the same for HR rates. Let us look at our given data for reference.

```{r}

abs(sd(cleaned_data$HR)^2 - mean(cleaned_data$HR))

```

The variance and mean of our observed are shockingly close, only differing by 0.0029! This gives us grounds to investigate using a poisson prior.

```{r}

# Define the JAGS model with Poisson likelihood
poisson_uninformative_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0, 1)  # Weakly informative normal prior
  }
}
"

# Initialize data for JAGS
poisson_data <- list(
  N = N,
  K = K,
  HR = HR,
  stadium = stadium_levels
)

# Define initial values
```

```
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1)
  )
}

# Run the Gibbs sampler
poisson_uninformative <- jags.model(
  textConnection(poisson_uninformative_code),
  data = poisson_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(poisson_uninformative, 1000)

# Sample from the posterior
poisson_uninformative_results <- coda.samples(
  poisson_uninformative,
  variable.names = c("alpha"),
  n.iter = 5000
)

autocorr.plot(poisson_uninformative_results)

poisson_uninformative_dic <- dic.samples(poisson_uninformative,
n.iter = 1000)
print(poisson_uninformative_dic)

# Analysis of results
summary(poisson_uninformative_results)

# All converged (can double check with gelman.diag() but that isn't
necessary)
plot(poisson_uninformative_results)

gelman.diag(poisson_uninformative_results)

# Least influential stadium #2 - Busch Stadium
# Mean = -2.802
```

```
# All stadiums have negative influence on HR rates!
plot(poisson_uninformative_results[, "alpha[2]"])

# Posterior Predictive distribution mapping for a Poisson prior
posterior_means <- colMeans(as.matrix(poisson_uninformative_results))
alpha_means <- posterior_means[1:K]
simulated_HR <- numeric(N)
for (i in 1:N) {
  stadium_idx <- stadium_levels[i]
  mu <- exp(alpha_means[stadium_idx])
  simulated_HR[i] <- rpois(1, lambda = mu)
}

# Plot the posterior predictive check
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)

```

We see decent success from the above posterior predictive mapping, but it's not perfect.

Sensitivity Analysis

```{r}

sd(stadium_levels)^2

# Define the JAGS model with Poisson likelihood
poisson_informative_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
```

```
    alpha[j] ~ dnorm(0.110974, 0.01)
  }
}
"

# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1)
  )
}

# Run the Gibbs sampler
poisson_informative <- jags.model(
  textConnection(poisson_informative_code),
  data = poisson_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(poisson_informative, 1000)

# Sample from the posterior
poisson_informative_results <- coda.samples(
  poisson_informative,
  variable.names = c("alpha"),
  n.iter = 5000
)
poisson_informative_dic <- dic.samples(poisson_informative, n.iter =
1000)
print(poisson_informative_dic)


# Analysis of results
summary(poisson_informative_results)

# All converged (can double check with gelman.diag() but that isn't
necessary)
plot(poisson_informative_results)
```

```
# Least influential stadium #3 - Tropicana Field
# Mean = -2.950
# All stadiums have negative influence on HR rates!
plot(poisson_informative_results[, "alpha[3]"])

# Posterior Predictive distribution mapping for a Poisson prior
posterior_means <- colMeans(as.matrix(poisson_informative_results))
alpha_means <- posterior_means[1:K]
simulated_HR <- numeric(N)
for (i in 1:N) {
  stadium_idx <- stadium_levels[i]
  mu <- exp(alpha_means[stadium_idx])
  simulated_HR[i] <- rpois(1, lambda = mu)
}

# Plot the posterior predictive check
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)

```


```{r}

# poisson, sample variance, 0 mean
p_ssigma_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0, 0.1138833)
  }
}
"
```

```
# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1)
  )
}

# Run the Gibbs sampler
p_ssigma_model <- jags.model(
  textConnection(p_ssigma_code),
  data = poisson_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(p_ssigma_model, 1000)

# Sample from the posterior
p_ssigma_results <- coda.samples(
  p_ssigma_model,
  variable.names = c("alpha"),
  n.iter = 5000
)
p_ssigma_dic <- dic.samples(p_ssigma_model, n.iter = 1000)
print(p_ssigma_dic)


par(mfrow = c(1, 1))  # Single plot
# Adjust margins
par(mar = c(5, 4, 2, 2))

plot(p_ssigma_results[,"alpha[20]"])

# Posterior Predictive distribution mapping for a Poisson prior
posterior_means <- colMeans(as.matrix(p_ssigma_results))
alpha_means <- posterior_means[1:K]
simulated_HR <- numeric(N)
for (i in 1:N) {
  stadium_idx <- stadium_levels[i]
```

```
  mu <- exp(alpha_means[stadium_idx])
  simulated_HR[i] <- rpois(1, lambda = mu)
}

gelman.diag(p_ssigma_results)

# Plot the posterior predictive check
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)


```

```{r}

# poisson broad
p_broad_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0, 100)
  }
}
"

# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1)
  )
}

# Run the Gibbs sampler
```

```
p_broad_model <- jags.model(
  textConnection(p_broad_code),
  data = poisson_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(p_broad_model, 1000)

# Calculate DIC
p_broad_dic <- dic.samples(p_broad_model, n.iter = 1000)
print(p_broad_dic)

# Sample from the posterior
p_broad_results <- coda.samples(
  p_broad_model,
  variable.names = c("alpha"),
  n.iter = 5000
)


```

```{r}

# poisson, sample variance, sample mean
p_smu_ssigma_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0.110974, 0.1138833)
  }
```

```
}
"

# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1)
  )
}

# Run the Gibbs sampler
p_smu_ssigma_model <- jags.model(
  textConnection(p_smu_ssigma_code),
  data = poisson_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(p_smu_ssigma_model, 1000)

# Calculate DIC
p_smu_ssigma_dic <- dic.samples(p_smu_ssigma_model, n.iter = 1000)
print(p_smu_ssigma_dic)

# Sample from the posterior
p_smu_ssigma_results <- coda.samples(
  p_smu_ssigma_model,
  variable.names = c("alpha"),
  n.iter = 5000
)


```


Bayesian Analysis 3: HR v. Stadiums, Negative Binomial Prior

Let us investigate a negative binomial prior instead.

```{r}
```

```
mean_HR <- mean(cleaned_data$HR)
var_HR <- var(cleaned_data$HR)
var_HR / mean_HR

# Moderate dispersion
nbin_gamma_code <- "
model {
  for (i in 1:N) {
    # Negative binomial likelihood
    HR[i] ~ dnegbin(p[i], r)

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
    p[i] <- r / (r + mu[i])
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0, 0.1138833)
  }

  # Priors for dispersion parameter (r)
  r ~ dgamma(5, .5)
}
"

# Initialize data for JAGS
nbin_data <- list(
  N = N,
  K = K,
  HR = HR,
  stadium = stadium_levels
)

# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1),
    r = runif(1, 1, 5)
  )
}
```

```
# Run the Gibbs sampler
nbin_gamma_model <- jags.model(
  textConnection(nbin_gamma_code),
  data = nbin_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(nbin_gamma_model, 1000)

# Sample from the posterior
nbin_gamma_results <- coda.samples(
  nbin_gamma_model,
  variable.names = c("alpha", "r"),
  n.iter = 1000
)


# Analysis of results
summary(nbin_gamma_results)
plot(nbin_gamma_results)

# r w/ Mean = 4.862, SD = 0.76874; low overdispersion.
plot(nbin_gamma_results[, "alpha[34]"])

nbin_gamma_dic <- dic.samples(nbin_gamma_model, n.iter = 1000)
print(nbin_gamma_dic)

# Extract posterior means of alpha and r
posterior_means <- colMeans(as.matrix(nbin_gamma_results))
alpha_means <- posterior_means[1:K]
r_mean <- posterior_means["r"]

# Simulate HR based on the posterior predictive distribution
simulated_HR <- numeric(N)
for (i in 1:N) {
  stadium_idx <- stadium_levels[i]
  mu <- exp(alpha_means[stadium_idx])
```

```
  p <- r_mean / (r_mean + mu)
  simulated_HR[i] <- rnbinom(1, size = r_mean, prob = p)
}

# Plot the posterior predictive check
# It's really close!
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)
```

```{r}

# Moderate dispersion
nbin_exp_code <- "
model {
  for (i in 1:N) {
    # Negative binomial likelihood
    HR[i] ~ dnegbin(p[i], r)

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]]
    p[i] <- r / (r + mu[i])
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0, 0.1138833)
  }

  # Priors for dispersion parameter (r)
  r ~ dexp(0.1)
}
"

# Initialize data for JAGS
nbin_data <- list(
  N = N,
  K = K,
  HR = HR,
```

```
  stadium = stadium_levels
)

# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1),
    r = runif(1, 1, 5)
  )
}


# Run the Gibbs sampler
nbin_exp_model <- jags.model(
  textConnection(nbin_exp_code),
  data = nbin_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(nbin_exp_model, 1000)

unique(cleaned_data$Stadium)

# Sample from the posterior
nbin_exp_results <- coda.samples(
  nbin_exp_model,
  variable.names = c("alpha", "r"),
  n.iter = 1000
)


# Analysis of results
summary(nbin_exp_results)
plot(nbin_exp_results)

# r w/ Mean = 4.862, SD = 0.76874; low overdispersion.
plot(nbin_exp_results[, "alpha[34]"])

nbin_exp_dic <- dic.samples(nbin_exp_results, n.iter = 1000)
```

```
print(nbin_exp_dic)

# Extract posterior means of alpha and r
posterior_means <- colMeans(as.matrix(nbin_gamma_results))
alpha_means <- posterior_means[1:K]
r_mean <- posterior_means["r"]

# Simulate HR based on the posterior predictive distribution
simulated_HR <- numeric(N)
for (i in 1:N) {
  stadium_idx <- stadium_levels[i]
  mu <- exp(alpha_means[stadium_idx])
  p <- r_mean / (r_mean + mu)
  simulated_HR[i] <- rnbinom(1, size = r_mean, prob = p)
}

# Plot the posterior predictive check
# It's really close!
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)
```

```

## Bayesian Analysis 4: HR v. Pitcher Handedness, Poisson

Time to answer our central question: what's the relationship between HR counts, the stadiums' played in, and the handedness of the pitcher. Is there a relationship between if a ball was lobbed with the right or left hand and it becoming a home run-winning hit? Because the accuracy of both the Poisson and Negative Binomial Priors are roughly equal (when assuming an uninformative hyperparameter), but the runtimes were greatly different, for ease of access we will use a poisson prior for this subsequent model.

```{r}

plot(density(as.numeric(factor(cleaned_data$Pitch.Hand))),
main="Density Plot of Pitching Hand(s)")
```
```

```
mean(as.numeric(factor(cleaned_data$Pitch.Hand)))
var(as.numeric(factor(cleaned_data$Pitch.Hand)))

# Define the JAGS model focusing on handedness, uninformative
uninformative_ph_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- beta_hand * pitch_hand[i]
  }

  # Prior for beta_hand (effect of handedness)
  beta_hand ~ dnorm(0, 1)
}
"

# Initialize data for JAGS
ph_data <- list(
  N = N,
  HR = HR,
  pitch_hand = pitch_hand
)

# Define initial values
inits <- function() {
  list(
    beta_hand = rnorm(1, 0, 1)
  )
}

# Run the Gibbs sampler
uninformative_ph_model <- jags.model(
  textConnection(uninformative_ph_code),
  data = ph_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)
```

```
# Update the model (burn-in period)
update(uninformative_ph_model, 1000)

# Sample from the posterior
uninformative_ph_results <- coda.samples(
  uninformative_ph_model,
  variable.names = c("beta_hand"),
  n.iter = 5000
)

uninformative_ph_dic <- dic.samples(uninformative_ph_model, n.iter =
1000)
print(uninformative_ph_dic)


# Analysis of results
# beta_hand w/ Mean = -1.300451, SD = 0.0004532 (4.532e-03); Some
negative effect
summary(uninformative_ph_results)

# It has converged!
plot(uninformative_ph_results)

# Extract posterior mean of beta_hand
posterior_means <- colMeans(as.matrix(uninformative_ph_results))
beta_hand_mean <- posterior_means["beta_hand"]

# Simulate HR based on the posterior predictive distribution
simulated_HR <- numeric(N)
for (i in 1:N) {
  # Calculate the mean (mu) based on beta_hand and pitch_hand
  mu <- exp(beta_hand_mean * pitch_hand[i])

  # Simulate HR using Poisson distribution
  simulated_HR[i] <- rpois(1, lambda = mu)
}

# Plot the posterior predictive check
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)
```

```
```

```{r}

ssigma_ph_code <- "
model {
  for (i in 1:N) {
    # Poisson likelihood
    HR[i] ~ dpois(mu[i])

    # Link function for the mean
    log(mu[i]) <- beta_hand * pitch_hand[i]
  }

  # Prior for beta_hand (effect of handedness)
  beta_hand ~ dnorm(0, 0.2148009)
}
"

# Initialize data for JAGS
ph_data <- list(
  N = N,
  HR = HR,
  pitch_hand = pitch_hand
)

# Define initial values
inits <- function() {
  list(
    beta_hand = rnorm(1, 0, 1)
  )
}

# Run the Gibbs sampler
ssigma_ph_model <- jags.model(
  textConnection(ssigma_ph_code),
  data = ph_data,
  inits = inits,
  n.chains = 3,
```

```
  n.adapt = 1000
)

# Update the model (burn-in period)
update(ssigma_ph_model, 1000)

# Sample from the posterior
ssigma_ph_results <- coda.samples(
  ssigma_ph_model,
  variable.names = c("beta_hand"),
  n.iter = 1000
)

ssigma_ph_dic <- dic.samples(ssigma_ph_model, n.iter = 1000)
print(ssigma_ph_dic)


# Analysis of results
# beta_hand w/ Mean = -1.300451, SD = 0.0004532 (4.532e-03); Some
negative effect
summary(uninformative_ph_results)

# It has converged!
plot(uninformative_ph_results)

# Extract posterior mean of beta_hand
posterior_means <- colMeans(as.matrix(uninformative_ph_results))
beta_hand_mean <- posterior_means["beta_hand"]

# Simulate HR based on the posterior predictive distribution
simulated_HR <- numeric(N)
for (i in 1:N) {
  # Calculate the mean (mu) based on beta_hand and pitch_hand
  mu <- exp(beta_hand_mean * pitch_hand[i])

  # Simulate HR using Poisson distribution
  simulated_HR[i] <- rpois(1, lambda = mu)
}

# Plot the posterior predictive check
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
```

```
lines(density(HR), col = "red", lwd = 2)
```

```

From the above, we can see that the pitching hand does have a negative effect on HR count Recall that we denoted the Right hand with "1" and the left with "2". From the negative effect, we know that the more "influential" of the two pitching styles is the right hand as that is closer to the mean. Ergo, a right-handed pitch will negative impact a players HR count (per game).

For the posterior-predictive check, we see virtually the same results as before when operating with exclusively stadium data. This establishes that both the stadium and pitching hand are equally as accurate when predicting HR counts.

## Bayesian Analysis 5: HR v. Stadiums + Pitcher handedness, Negative Binomial Prior

The next logical step is to complicate that last question and incorporate *both* variables. Is there a relationship between Home Runs and the stadium it was in, as well as the hand that threw it?

```{r}

# Define the JAGS model w/ negative binomial prior & pitching hand
jags_model_code <- "
model {
  for (i in 1:N) {
    # Negative binomial likelihood
    HR[i] ~ dnegbin(p[i], r)

    # Link function for the mean
    log(mu[i]) <- alpha[stadium[i]] + beta_hand * pitch_hand[i]
    p[i] <- r / (r + mu[i])
  }

  # Priors for alpha (stadium effects)
  for (j in 1:K) {
    alpha[j] ~ dnorm(0, 0.01)
  }
```
```

```
  # Prior for beta_hand (effect of Pitch.Hand)
  beta_hand ~ dnorm(0, 0.01)

  # Priors for dispersion parameter (r)
  r ~ dgamma(1, 1)
}
"


jags_data <- list(
  N = N,
  K = K,
  HR = HR,
  stadium = stadium_levels,
  pitch_hand = pitch_hand
)

# Define initial values
inits <- function() {
  list(
    alpha = rnorm(K, 0, 1),
    beta_hand = rnorm(1, 0, 1),
    r = runif(1, 1, 5)
  )
}

# Run the Gibbs sampler
jags_model <- jags.model(
  textConnection(jags_model_code),
  data = jags_data,
  inits = inits,
  n.chains = 3,
  n.adapt = 1000
)

# Update the model (burn-in period)
update(jags_model, 1000)

# Sample from the posterior
test_results <- coda.samples(
  jags_model,
```

```
  variable.names = c("alpha", "beta_hand", "r"),
  n.iter = 5000
)

# Analysis of results
summary(test_results)
plot(test_results)

# r w/ Mean = 4.83, SD = 0.78066; low overdispersion.
plot(test_results[, "r"])

# beta_hand w/ Mean = 0.0008151, SD = 0.01746
# Basically no influence on HR rates when also incorporating the
stadium
# Convergence doesn't look great
plot(test_results[, "beta_hand"])

# Verify convergence with Gelman-Rubin diagnostic
# R_hat = 1.02, Upper C.I. = 1.08
# Erring on convergence issues, but is still within reasonable ranges
gelman.diag(test_results)

# All stadiums still have negative influence on HR rates!
plot(test_results[, "alpha[3]"])

# Extract posterior means of alpha and r
posterior_means <- colMeans(as.matrix(test_results))
alpha_means <- posterior_means[1:K]
r_mean <- posterior_means["r"]

# Simulate HR based on the posterior predictive distribution
simulated_HR <- numeric(N)
for (i in 1:N) {
  stadium_idx <- stadium_levels[i]
  mu <- exp(alpha_means[stadium_idx])
  p <- r_mean / (r_mean + mu)
  simulated_HR[i] <- rnbinom(1, size = r_mean, prob = p)
}

# Plot the posterior predictive check
# It's not any better than before. Ergo, Handedness has no effect on
Stadium performance
```

```
hist(simulated_HR, probability = TRUE, col = "blue", main =
"Posterior Predictive Check")
lines(density(HR), col = "red", lwd = 2)
```

```

We can conclude there is no relationship; handedness is negligible when stadiums are factored in. The stadium is a trump factor.
```