Montana State University

Department of Mathematical Sciences

Writing Project

Predicting the PITCHf/x Pitch Classifier

Author:
Christian Stratton

Supervisors:
Dr. Andrew Hoegh
Dr. Jennifer Green

Spring 2018

A writing project submitted in partial fulfillment
of the requirements for the degree

Master's of Science in Statistics

APPROVAL

of a writing project submitted by

Christian Stratton

This writing project has been read by the writing project advisor and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Statistics Faculty.

Date	Andrew Hoegh Writing Project Advisor
Date	Jennifer Green Writing Project Advisor
Date	Mark C. Greenwood Writing Projects Coordinator

Contents

Abstract

In Major League Baseball (MLB), the pitcher is often regarded as one of the most influential players on the team; a dominant pitcher has the ability to shutdown an opposing team and control the game. As a result, the ability to predict the next pitch can give a team a considerable competitive advantage; so much so, that controversy has recently developed in the MLB around teams attempting to identify signals exchanged between the opposing pitcher and catcher that relay information regarding the type of the next pitch to be thrown. And while the MLB has officially prohibited attempts to steal signs from opposing teams, their recent decision to allow iPads in the dugout has opened the door for the use of predictive modeling techniques in baseball. The MLB tracks data on every game played, which is stored in an online database known as the PITCHf/x database. The aim of this research is to build a model capable of predicting the pitch type classifier in the PITCHf/x database. A Bayesian hierarchical nominal multinomial regression model was fit to pitch data on Clayton Kershaw scraped from the PITCHf/x database; pitches were clustered based on the catcher calling the pitch. Predicted pitch type probabilities were compared to those from this model's non-hierarchical counterpart using a conditional Bayes factor. The analyses showed that the hierarchical model provided a better description of the data. The utility of this model is discussed in this article and practical applications of the results are explored. A simulation study on the advantages of hierarchical modeling is also presented.

1 Introduction

In the fall of 2017, controversy rocked Major League Baseball (MLB) as the number one seeded Boston Red Sox were accused of using electronic devices to "steal" signals exchanged between the opposing pitcher and catcher that relayed information regarding the type of the next pitch to be thrown; this act, known as "sign stealing," has a rich heritage in American baseball. Since the advent of the league in 1869, teams have been attempting to steal signs from the opposing team to gain a competitive edge (Schmidt, 2017). In American baseball, the pitcher is often regarded as the most influential player on the field; a dominant pitcher single-handedly has the ability to prevent an opposing team from scoring, an ability unique to that position. Therefore, knowing what pitch could be thrown next, and thereby negating some of that pitcher's dominance, can give a team a serious competitive edge.

The purpose of this research is to build a model capable of predicting the next pitch thrown by an MLB pitcher based on covariates that describe the current state of the game. In this section, we begin by motivating a model-based approach to pitch prediction and conclude with a description of the data collection and cleaning process.

1.1 Motivating Model-Based Pitch Prediction

The "sign stealing" stealing controversy discussed above suggests that there is a desire in the league to predict the next pitch, but is there a means to do so? Up until late in the 2017 season, teams were not allowed to use electronics to gain a competitive edge. However, in a recent decision by the MLB, each team is allowed an Apple iPad in the dugout (Novy-Williams, 2016). And while the MLB still expressly prohibits the use of electronics for communication (i.e., sign stealing), access to such a computing device opens the door for model-based pitch prediction, if it can be accomplished in real time. Therefore, there is a practical motivation for the development of a model capable of predicting the next pitch to be thrown.

Furthermore, there is a lot of money in the MLB. The 2017 World Series winners made over \$27.6 million in post season earnings. So there is not only a desire for the development of a pitch predicting model and a means to implement it, but also a very practical motivator for the creation of such a model.

1.1.1 A Motivating Example

To further motivate the need for a pitch predicting model, we consider a specific example where such a model could have been of use. On October 13th of 2016, the Washington Nationals met the Los Angeles Dodgers for game five of the National League Divisional Series (NLDS). The NLDS is the quarterfinal series of the World Series tournament, and is a best of five series in which the team that wins three games advances to the semifinals.

We pick up the story in the bottom of the ninth inning, the last inning of the game. The Dodgers lead the Nationals 4-3, and the Nationals were up to bat fighting to keep their World Series hopes alive. The first batter, Trea Turner, struck out swinging, giving the Dodgers the first of three required outs to end the game. The next two batters walked, putting the tying run on second base and the winning run on first base. At this point in the game, the Dodgers decided to bring in their ace pitcher, Clayton Kershaw, who is widely regarded as one of the most dominant pitchers of the modern era (Crasnick, 2018).

The first batter, Daniel Murphy, popped out to second base giving the

Dodgers the second out of the inning, just one away from ending the game. And so, with the game and the Nationals' World Series aspirations on the line, Wilmer Difo found himself up to bat. He fought Kershaw to a 1-1 count, before taking a strike on Kershaw's slider, bringing Difo to a 1-2 count, just a single strike away from ending the game. And so for Difo, the pivotal question and the focus of this research, is: "What is the next pitch?"

This is a question we will attempt to answer over the course of this paper through the development of a pitch prediction model. However, to fit such a model, we first need data to build it upon.

1.2 Data Collection

American baseball has a reputation for meticulously tracking descriptive statistics; a reputation that is embodied in their PITCHf/x database. In this section, we describe the PITCHf/x database, how we scraped the data used in our analysis from the database, and the data cleaning process.

1.2.1 The PITCHf/x Database

PITCHf/x is a term for the system of cameras Major League Baseball Advanced Media (MLBAM) uses to track the flight pattern of pitches at every MLB game (based on metrics like curvature, vertical drop, horizontal and vertical location over home plate, and many others). These measurements, and a wealth of other information including, but not limited to, longitudinal data on each player in the league, box-scores for every game played each season, commentary by announcers, etc., are stored in .xml format on the publicly available PITCHf/x database website. The data are structured in a hierarchical framework, where there exist clear parent-to-child relationships. For example, individual player stats are nested within teams, which are nested within games, which are nested within days, etc. One particularly useful identifier in the database is the game identification link (GID), which references the web page in the database that contains the pitch-by-pitch information for that particular game (Sievert, 2014).

Within the database are variables denoting the type of pitch thrown and the count associated with that pitch, which form the basis of the model we chose to fit. These variables, and others, are discussed in greater detail in section 3.2, where we provide the details of the exact pitch prediction model we fit. Next, we consider how we scraped the data from the PITCHf/x database into a format suitable for model building.

1.2.2 The *pitchRx* Package and Data Cleaning

To scrape the data into a usable format, we used a modified version of the *scrape* function in the *pitchRx* R package, developed by Carson Sievert (Sievert, 2014). By default, this function references a GID, and scrapes the pitch-by-pitch information into an R data frame object. In addition to this pitch-by-pitch information, we were interested in including batter information in our model. To do so, we modified the *scrape* source code to also pull batter information into the data frame. After modifying the function, we scraped data on every game played between August 2008 and August 2014.

Once we had scraped all these data into an SQL database, we decided to focus our pitch predicting model on Clayton Kershaw, the pitcher described in the motivating example. We chose to do so for a number of reasons. As we mentioned above, Kershaw is often regarded as one of the most prolific pitchers still active in the MLB, so we were interested in trying to gain insight into what makes him such a dominant player. Furthermore, it seems reasonable to believe his dominance may arise through thoughtful pitch sequencing, making him a prime candidate for a model such as this. And finally, he played a large role in the 2017 World Series, making him one of the more recognizable pitchers for this project.

Once we restricted our focus to Kershaw, there was a large amount of data cleaning to be done. We decided to throw out any all-star games he played in; in these games, Kershaw throws to a myriad of different catchers that he does not see in the regular season. Because we chose to consider a hierarchical model based on the catcher to whom Kershaw was throwing (see section 3.2), we did not want to include these catchers as members of the hierarchy. Over the course of his career, he may have only thrown 20 or less pitches to some of them, providing limited information about the relationship between pitch type probabilities and the covariates for those particular catchers.

We then identified the catcher to whom each pitch was thrown in the dataset. This information was not available directly through the *scrape* function, so we built a data frame that provided the catcher by GID and then merged that data frame with the pitches data frame available through the *scrape* function.

Next, we removed any observations in which the pitch type classifier was missing, or classified as an intentional or unintentional walk. While these cases could themselves make for an interesting analysis, they made up less than one percent of the data, and were not relevant to the goal of this research.

Finally, we created additional predictor variables, such as a previous pitch type classifier. This particular predictor presented its own challenges, as we did not want to carry over the previous pitch type between at-bats, so care had to be taken when creating this variable to ensure that the previous pitch type for the first pitch of each at-bat was recorded as "none." With the data scraped and appropriately cleaned, we turned our attention to fitting the model itself.

In section 2, we outline background modeling techniques and describe some relevant Bayesian computational methods. In section 3, we describe in detail the particular model we fit to these data, and discuss the sampler used to fit that model. The fourth section of this paper provides a simulation study to demonstrate the advantages of hierarchical modeling and in sections 5 and 6, we discuss the results of the Kershaw analysis.

2 Background Modeling Techniques

In this section, we begin by outlining some of the more common computational methods used in Bayesian statistics, methods upon which we developed our sampler for the Kershaw model. We then discuss a number of common Bayesian models that provide a foundation for the model we fit to the Kershaw data.

2.1 Bayesian Computational Methods

In Bayesian statistics, inference is driven by the posterior distribution of the unknown parameters; this posterior distribution is a function of the likelihood of a sampling model for the observed data and prior understanding of the unknown parameters. That is, if we let $p(\boldsymbol{y}|\boldsymbol{\theta})$ denote the likelihood for some sampling model and $p(\boldsymbol{\theta})$ denote the prior distribution on the unknown vector of parameters $\boldsymbol{\theta}$, then, by Bayes' rule, the posterior distribution is given by:

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\Theta} p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{1}$$

In special cases, often when conjugate priors are placed on the unknown parameters, a closed form solution for this posterior distribution is available (Hoff, 2010).

Definition 2.1. Conjugate

A class $\mathcal{P}$ of prior distributions $p(\theta)$ is called conjugate for a class $\mathcal{F}$ of *sampling distributions* $p(y|\theta)$ *if* $p(\theta) \in \mathcal{P}$ *and* $p(\theta|y) \in \mathcal{P}\ \forall\ p(y|\theta) \in \mathcal{F}$.

A classical example of a model for which a closed form posterior distribution exists is the beta-binomial model; in this setting, the sampling model is binomial with an unknown probability parameter θ. If we place a beta prior distribution on θ, it can be shown the posterior distribution of θ is also of the beta family. That is, given the following complete description of the model:

$$\begin{gathered} y_1, ..., y_n \sim \text{binomial}(1, \theta) \\ \theta \sim \text{beta}(\alpha, \beta) \end{gathered} \tag{2}$$

It can be shown that

$$\theta | y_1, ..., y_n \sim \text{beta}\left(\sum_{i=1}^{n} y_i + \alpha, n - \sum_{i=1}^{n} y_i + \beta\right) \tag{3}$$

Because the prior and posterior distributions are of the same family, the beta distribution is considered a conjugate prior for this sampling model. This relationship leads to straightforward calculations of the posterior distribution. However, in many cases, a closed form solution is not available. In such cases, we resort to computational methods that approximate the posterior distribution by sampling from this unknown distribution using iterative procedures.

2.1.1 The Gibbs Sampler

The Gibbs sampler is one of the most common choices of sampling algorithms, as it is one of the more efficient sampling algorithms currently available. The Gibbs sampler is a method of iteratively sampling from the full conditional distributions of each unknown parameter, which results in a sequence of parameter values that converge to the true target joint posterior distribution of the unknown parameters. This method requires calculation of those full conditional posterior distributions. The full conditional posterior distribution of a parameter is the posterior probability distribution of that parameter, conditioned on the other unknown parameters, data and priors. For computational ease, semi-conjugate prior distributions are often placed on each parameter which allow for closed form solutions for each full conditional posterior.

Definition 2.2. Semi-Conjugate

If $\mathcal{F}$ is a class of sampling distributions $p(y|\theta, \phi)$, and $\mathcal{P}$ is a class of prior distributions for θ conditional on ϕ, then the class $\mathcal{P}$ is semi-conjugate for $\mathcal{F}$ if $p(\theta|y, \phi) \in \mathcal{P} \ \forall \ p(y|\theta, \phi) \in \mathcal{F}$ and $p(y|\phi) \in \mathcal{P}$.

Once full conditional posterior distributions are derived for each parameter, the Gibbs sampler can be used to approximate samples from the joint posterior distribution, using the following algorithm:

The Gibbs Sampler

Let $\boldsymbol{\theta} = \{\theta_1, ..., \theta_p\}$ denote the vector of unknown parameters, $p(\theta_i|.)$ denote the full conditional distribution of θ_i and initialize $\boldsymbol{\theta}^{(0)} = \{\theta_1^{(0)}, ..., \theta_p^{(0)}\}$. The Gibbs sampler generates $\boldsymbol{\theta}^{(s)}$ from $\boldsymbol{\theta}^{(s-1)}$ in the following manner:

```
1) sample $\theta_1^{(s)} \sim p(\theta_1|\theta_2^{(s-1)}, \theta_3^{(s-1)}, ..., \theta_p^{(s-1)}, y_1, ..., y_n)$

2) sample $\theta_2^{(s)} \sim p(\theta_2|\theta_1^{(s)}, \theta_3^{(s-1)}, ..., \theta_p^{(s-1)}, y_1, ..., y_n)$
   ⋮
p) sample $\theta_p^{(s)} \sim p(\theta_p|\theta_1^{(s)}, \theta_2^{(s)}, ..., \theta_{p-1}^{(s)}, y_1, ..., y_n)$
```

This process generates a dependent sequence of vectors $\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(S)}$ (for convenience, this sequence is often referred to as a Markov-Chain Monte Carlo sequence, or MCMC chain) that, for large enough S, converge to the joint posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$. For a thorough discussion of why this algorithm converges to the target distribution, see Geman and Geman (1984). The required S to achieve convergence depends upon the complexity of the problem and the particular combination of sampling model and priors used; convergence diagnostics, discussed in section 2.1.4 below, are often used to assess whether a large enough S was considered. Furthermore, $\boldsymbol{\theta}^{(s)}$ depends only upon $\boldsymbol{\theta}^{(s-1)}$, and so this sequence is a Markov chain and benefits from all the properties of such chains (Hoff, 2010).

A classical example of a situation where Gibbs sampling is used is the normal sampling model with unknown mean and variance. Given the following complete description of the model,

$$
\begin{aligned}
y_1, ..., y_n|\theta, \sigma^2 &\sim \text{normal}(\theta, \sigma^2) \\
\theta &\sim \text{normal}(\mu_0, \tau_0^2) \\
\sigma^2 &\sim \text{inverse-gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right)
\end{aligned} \tag{4}
$$

It can be shown that the full conditional posterior distributions of θ and σ^2, $p(\theta|.)$ and $p(\sigma^2|.)$ respectively, are

$$
\begin{aligned}
\theta|\sigma^2, y_1, ..., y_n &\sim \text{normal}(\mu_n, \tau_n^2) \\
\text{where } \mu_n &= \frac{\frac{\mu_0}{\tau_0^2} + \frac{n\bar{y}}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}} \text{ and } \tau_n^2 = \left(\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}\right)^{-1} \\
\sigma^2|\theta, y_1, ..., y_n &\sim \text{inverse-gamma}\left(\frac{\nu_n}{2}, \frac{\nu_n\sigma_n^2}{2}\right) \\
\text{where } \nu_n = \nu_0 + n &\text{ and } \sigma_n^2 = \frac{1}{\nu_n}\left(\nu_0\sigma_0^2 + \sum_{i=1}^{n}(y_i - \theta)^2\right)
\end{aligned} \tag{5}
$$

Given these full conditional distributions, a Gibbs sampler can be implemented to sample from the joint posterior distribution of (θ, σ^2) (Hoff, 2010). In some cases, such as when fitting generalized linear models, semi-conjugate priors or closed form solutions for the full conditional distributions of the unknown parameters do not exist. In such cases, the Gibbs sampler cannot be used to approximate the joint posterior distribution and other methods must be considered.

2.1.2 The Metropolis-Hastings Algorithm

When closed form solutions for the full conditional distributions are not available, the Metropolis-Hastings algorithm is often used to sample from the target distribution. This algorithm does not require specification of the full conditional distributions; instead, a new set of parameter estimates is proposed by sampling from a proposal distribution, denoted $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s)})$, where $\boldsymbol{\theta}^{(s)}$ denotes the current set of parameter estimates and $\boldsymbol{\theta}^*$ denotes the proposed set of parameter estimates. Often, this proposal distribution is a random walk function, such as $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s)}) \sim \text{mvnormal}(\boldsymbol{\theta}^{(s)}, \gamma^2 \boldsymbol{I}_p)$, where γ^2 is thought of as the *step size*, or the value that determines the distance the proposed set of parameter estimates falls from the current set, on average.

Once proposed, the new set of estimates is assessed by evaluating the posterior likelihood at both the current and proposed set of parameter estimates; if the proposed set results in a larger likelihood, the proposal is accepted. If it results in a smaller likelihood, it is accepted with some probability r, defined below. Allowing the algorithm to accept proposals that result in smaller likelihoods enables it to escape local maxima and spend time in each region of the parameter space proportional to the target density. Formally, the Metropolis-Hastings algorithm is defined by the following:

The Metropolis-Hastings Algorithm

Let $\boldsymbol{\theta}^{(s)}$ denote the current set of parameter estimates, $\boldsymbol{\theta}^*$ denote the proposed set of estimates and $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s)})$ denote the proposal distribution.

1) sample $\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s)} \sim J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s)})$.

2) compute the acceptance ratio $r = \frac{p(\boldsymbol{\theta}^*|\boldsymbol{y})}{p(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*)}{p(\boldsymbol{y}|\boldsymbol{\theta}^{(s)})p(\boldsymbol{\theta}^{(s)})}$.

3) set $\boldsymbol{\theta}^{(s+1)} = \begin{cases} \boldsymbol{\theta}^* & \text{with probability } min(r,1) \\ \boldsymbol{\theta}^{(s)} & \text{with probability } 1 - min(r,1) \end{cases}$.

This process generates a dependent sequence of vectors $\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(S)}$ that, for large enough S, converge to the joint posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$. Again, the required S depends upon the complexity of the model; convergence diagnostics should be used to assess whether the algorithm converged to the true joint posterior distribution. Note that in the above definition, we assume the proposal distribution is symmetric; that is, we assume that $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s)}) = J(\boldsymbol{\theta}^{(s)}|\boldsymbol{\theta}^*)$. The Metropolis-Hastings algorithm allows for non-symmetric proposals by adjusting for values that are proposed too often (Hoff, 2010), but that is not discussed here.

2.1.3 Mixed Sampling Methods

In more complicated settings, such as hierarchical generalized linear models, the Metropolis-Hastings algorithm can be integrated into the Gibbs sampler to allow for more efficient sampling from the target distribution; this is discussed in greater detail in section 3.2.

2.1.4 Convergence Diagnostics

The theory behind the above computational methods assures us that the MCMC chains must converge to the target distribution, if allowed to run long enough. It does not, however, tell us how long the algorithms must run to converge to that distribution. Therefore, convergence diagnostics must be used to determine if the algorithm has run long enough. While there are many tools used to assess the convergence of MCMC chains, most of them rely upon the same fundamental concept; if different MCMC chains are initialized from random starting locations overdispersed relative to the target distribution, and all converge to the same location in the parameter space, then we can have confidence that the chains have converged to the true target distribution. In this section, we discuss two tools that use this ideology: a graphical tool known as a trace plot and the Gelman-Rubin statistic.

For each individual parameter in the target distribution, we can think about tracing its path through the parameter space by graphing its value indexed by the iteration number. This is the idea behind a trace plot. Once a trace plot has settled into a location, such that the remaining MCMC iterations bounce around that value, we can have confidence that the parameter estimate has converged to its proper value. Graphically, trace plots that signify a parameter estimate has converged to its true value are often described as "fuzzy caterpillar" like in appearance, which is perhaps evident from figure 1 below.

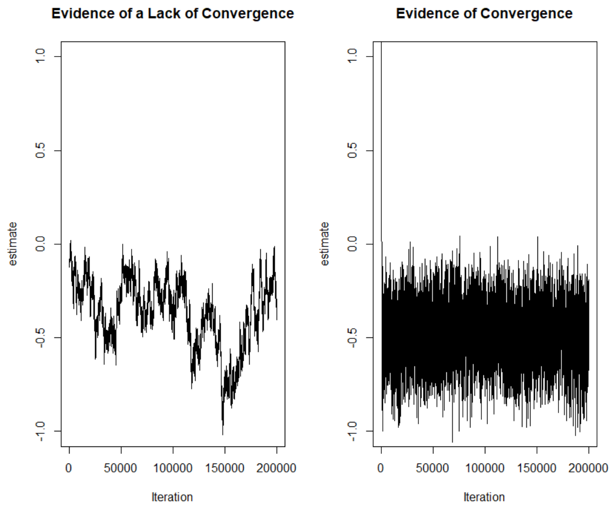

Figure 1: Examples of trace plots that display a lack of convergence (left) and evidence of convergence (right).

As the number of estimated parameters increases, it becomes difficult to assess the convergence of the MCMC with trace plots alone, due to the large number of plots that must be considered. In such cases, a numerical summary measure assessing the convergence of the sampler is desired; the Gelman-Rubin statistic is such a measure. This statistic, developed by Andrew Gelman and Donald Rubin in 1992, provides a means of determining whether multiple chains initialized at different starting locations have converged to the same location. It does so by using an ANOVA-like procedure, comparing the between-chain variability to the within-chain variability. If this ratio, known as a potential scale reduction factor (PSRF) is small, there is evidence that the chains are similar and have converged to the same distribution. Conversely, if the variability between the chains is large relative to the variability within each chain, there is evidence that the chains have not yet converged to the same distribution (Gelman and Rubin, 1992). Five years later, Gelman and Stephen Brooks proposed a correction to that statistic, but the corrected statistic still uses the same ANOVA-like approach (Brooks and Gelman, 1998). Due to this ANOVA-like approach, PSRFs close to one signify convergence across the chains.

2.2 Bayesian Hierarchical Modeling

The computational methods discussed above allow us to fit more complicated models from a Bayesian perspective. One common setting in which these sampling methods are used is the hierarchical model. In Bayesian statistics, the hierarchical model is often used to model data that arise from clusters. It allows us to account for both the within group and between group variability that we expect to see in such a setting, as well as account for the dependence in observations that we expect to see within a group. We begin by describing this relationship in the normal hierarchical framework and then extend these ideas to the multinomial framework in which we are working with the Kershaw data.

2.2.1 The Normal Hierarchical Model

The normal hierarchical model is most easily understood through an analogy. Consider the classical case of modeling exam scores of students at a school. We expect similarities within each class and some differences across classes, but in general all classes to behave similarly relative to each other because they are in the same school. To model data such as these, we think of each group (class) as having its own mean that comes from a distribution describing the means across classes. Graphically, the normal hierarchical model looks like the following:

$$\begin{tikzcd}
 & & \mu, \tau^2 \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drr] & & \\
\theta_1 \arrow[d] & \theta_2 \arrow[d] & \cdots & \theta_{m-1} \arrow[d] & \theta_m \arrow[d] \\
\boldsymbol{Y}_1 & \boldsymbol{Y}_2 & \cdots & \boldsymbol{Y}_{m-1} & \boldsymbol{Y}_m \\
\sigma_1^2 \arrow[u] & \sigma_2^2 \arrow[u] & \cdots & \sigma_{m-1}^2 \arrow[u] & \sigma_m^2 \arrow[u] \\
 & & \nu_0, \sigma_0^2 \arrow[ull] \arrow[ul] \arrow[ur] \arrow[urr] & &
\end{tikzcd}$$

Figure 2: Graphical representation of the normal hierarchical model from *A First Course in Bayesian Data Analysis* (Hoff, 2010).

In this graphic, μ and τ^2 describe the mean and variance of the normal distribution at the top level of the hierarchy, respectively. The θ_i's represent each individual class mean, which arise from the normal(μ, τ^2) distribution. And finally, ν_0 and σ_0^2 are the parameters of the prior distribution placed on σ_i^2, the variance parameter for a particular class. The θ_i and σ_i^2 parameters then define the normal distributions that describe each of the sets of class-level observations.

The complete description of the normal hierarchical model is:

$$\begin{aligned} y_{1m}, ..., y_{n_m m} | \boldsymbol{\phi}_m &\sim \text{normal}(\theta_m, \sigma_m^2), \quad \boldsymbol{\phi}_m = \{\theta_m, \sigma_m^2\} \\ \theta_1, ..., \theta_M | \boldsymbol{\psi} &\sim \text{normal}(\mu, \tau^2), \quad \boldsymbol{\psi} = \{\mu, \tau^2\} \end{aligned} \tag{6}$$

where m indexes the hierarchy. The first relationship describes the within-class variability and the second relationship describes the between-class variability. Often, we assume homogeneity of variance across classes. In such case, this model requires prior distributions on the σ^2 parameter and on the hyper-parameters in $\boldsymbol{\psi}$. To fit the model, semi-conjugate priors can be placed on all three parameters to allow a Gibbs sampler (Hoff, 2010).

2.2.2 The Multivariate Normal Hierarchical Model

We can easily extend the normal hierarchical model to the multivariate normal case. In such a setting, the complete description of the model is:

$$\begin{aligned} \boldsymbol{y}_{1j}, ..., \boldsymbol{y}_{n_j j} | \boldsymbol{\phi}_j &\sim \text{mvnormal}(\boldsymbol{\theta}_j, \boldsymbol{\Sigma}), \quad \boldsymbol{\phi}_j = \{\boldsymbol{\theta}_j, \boldsymbol{\Sigma}\} \\ \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_J | \boldsymbol{\psi} &\sim \text{mvnormal}(\boldsymbol{\mu}, \boldsymbol{V}), \quad \boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{V}\} \end{aligned} \tag{7}$$

In this case, prior distributions are placed on the $\boldsymbol{\Sigma}$ parameter and hyper-parameters in $\boldsymbol{\psi}$ to get estimates; typically, semi-conjugate priors are placed on all three parameters to allow for Gibbs sampling (Hoff, 2010). The multivariate normal hierarchical model allows for a straightforward extension to normal hierarchical regression.

2.2.3 Normal Hierarchical Regression

The hierarchical normal regression model is a direct result of the multivariate normal hierarchical model described above, replacing $\boldsymbol{\theta}_j$ with the familiar mean structure from normal regression, $\boldsymbol{X}\boldsymbol{\beta}_j$. That is, we have:

$$\begin{aligned} \boldsymbol{y}_{1j}, ..., \boldsymbol{y}_{n_j j} | \boldsymbol{\phi}_j &\sim \text{mvnormal}(\boldsymbol{\theta}_j, \boldsymbol{\Sigma}), \quad \boldsymbol{\phi}_j = \{\boldsymbol{\theta}_j, \boldsymbol{\Sigma}\}, \boldsymbol{\theta}_j = \boldsymbol{X}\boldsymbol{\beta}_j \\ \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_J | \boldsymbol{\psi} &\sim \text{mvnormal}(\boldsymbol{\mu}, \boldsymbol{V}), \quad \boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{V}\} \end{aligned} \tag{8}$$

In this setting, prior distributions must be placed on $\boldsymbol{\Sigma}$ and the hyper-parameters in $\boldsymbol{\psi}$. A Gibbs sampler is again available with the right choice of semi-conjugate priors (Hoff, 2010). In the next section, we consider the benefits of a hierarchical model, and justify the extra complexity of such a model.

2.2.4 Shrinkage Estimators

In addition to accounting for the dependence in observations within groups, hierarchical modeling allows for better estimates of multiple group means than does the non-hierarchical alternative. When ignoring the hierarchical structure, the true mean of each group in the hierarchy is estimated by its individual group mean. In a hierarchical framework, the true mean of each group is estimated with a *shrinkage estimator*, which is a weighted average of the individual group mean and the overall grand mean of all the data; that is, this estimator “shrinks” each individual group mean towards the overall mean. The degree to which each group mean is shrunk towards the grand mean depends upon the sample size for that particular group; the larger the sample size, the lesser the effect of the shrinkage (Hoff, 2010).

Perhaps counter-intuitively, Stein and James showed that when there exist more than two groups, this shrinkage estimator results in a lower expected mean square prediction error than does the individual mean estimator, a phenomenon known as Stein’s paradox (Efron and Morris, 1977). Therefore, in a prediction context as is typically the case in a hierarchical framework, the hierarchical model outperforms its non-hierarchical counterpart. In a sense, the shrinkage estimator allows a group within the hierarchy to “borrow” information from the other groups in the hierarchy, a feature that proves very useful when we have little data on some members of the hierarchy. Next, we consider the hierarchical multinomial regression model and its relationship with the normal hierarchical model.

3 Methods

In this section, we build upon the methods discussed above to develop the model we fit to the Kershaw data. We begin with a discussion of multinomial hierarchical regression, and conclude with a discussion of the Kershaw model.

3.1 Multinomial Regression

Before considering the hierarchical multinomial regression model, we first consider the simpler case when there is no hierarchy. In general, with multinomial regression, the goal is to estimate the probabilities associated with each of the levels of the response variable; it is assumed that these probabilities are a function of covariates. To relate the covariates to the probabilities, we use the multinomial logit link function. For J levels of the response variable, the full description of the model is:

$$\boldsymbol{y}_1, ...\boldsymbol{y}_n|\boldsymbol{\beta} \sim \text{multinomial}(1, \boldsymbol{\pi}_i), \qquad \pi_{ij} = \frac{e^{\boldsymbol{x}_i\boldsymbol{\beta}_j}}{\sum_{k=1}^{J} e^{\boldsymbol{x}_i\boldsymbol{\beta}_k}} \tag{9}$$

where $\boldsymbol{y}_i$ is the response vector for the i^{th} observation and $\boldsymbol{\beta}_j$ is the vector of true regression coefficients associated with the j^{th} level of the response. In this particular parameterization of the model, we assume that each observation is a single draw from a multinomial random variable. We can think of this as placing a single ball into one of J bins. Therefore, $\boldsymbol{y}_i$ is a vector containing a single one, and $J - 1$ zeros. Note that this model implies that each of the J levels of the response has its own set of regression coefficients.

3.1.1 Hierarchical Multinomial Regression

In hierarchical multinomial regression, each member of the hierarchy has its own vector of probabilities for a particular pattern of the covariates. This is analogous to the normal hierarchical regression model, where each group has its own mean for a particular pattern of the covariates. Therefore, each member of the hierarchy has its own matrix of regression coefficients that relate the covariates to each level of the response through the multinomial logit link function described in equation 9 above.

If we think of each set of regression coefficients as arising from a multivariate normal distribution, then this model becomes a straight forward extension of the hierarchical multivariate normal model. Assuming J levels of the response and M members in the hierarchy, the complete description of the model is:

$$\begin{aligned} \boldsymbol{y}_1, ...\boldsymbol{y}_{n_m} | \boldsymbol{\pi}_{im} &\sim \text{multinomial}(1, \boldsymbol{\pi}_{im}), \quad \pi_{ijm} = \frac{e^{\boldsymbol{x}_i \boldsymbol{\beta}_{jm}}}{\sum_{k=1}^{J} e^{\boldsymbol{x}_i \boldsymbol{\beta}_{km}}} \\ \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_M | \boldsymbol{\psi} &\sim \text{mvnormal}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \end{aligned} \tag{10}$$

This model implies that each member of the hierarchy has a unique matrix of regression coefficients of dimension $p \times J$, where p represents the number of regression coefficients in the model and J represents the number of levels of the response. Next, we define the model in the context of the Kershaw data.

3.2 The Kershaw Model

In this section, we discuss the hierarchical multinomial regression model fit to the Kershaw data. First, we outline our exploratory data analysis, and then we describe the model we fit and conclude with a discussion of the computational algorithm used to approximate the joint posterior distribution.

3.2.1 Exploratory Data Analysis

Clayton Kershaw throws four different pitches: a fastball, a curveball, a slider and a change-up. Our goal with this analysis was to predict the probability of observing each of those four possibilities on any pitch. In order to build a model capable of predicting the type of pitch to be thrown, we need a response that denotes the type of pitch thrown. Fortunately, one of the variables contained within the PITCHf/x database is a pitch type classifier. This classifier predicts the type of pitch thrown based on a neural network algorithm that takes into account a number of factors, including the speed, curvature, and vertical and horizontal shift of the pitch. It is worth noting that this is not necessarily a perfect predictor of the type of pitch actually thrown; there are some entertaining interviews with players in which they share that the PITCHf/x classifier mis-classifies the type of pitch they throw. Nevertheless, the goal of this research is to predict the output of that classifier, which at the very least, serves as a good proxy for the type of movement we expect to see for a particular pitch (Nathan, 2011).

There are many potential predictor variables available in the PITCHf/x database. For this research, we chose to focus on the count variable, which is a categorical predictor with 12 levels. This variable denotes the count of balls to strikes, and is generally regarded as one of the most influential factors when trying to predict the type of pitch thrown. As visual evidence of this statement, we provide a plot the relative frequencies of each of Kershaw's four pitches by count in figure 3.

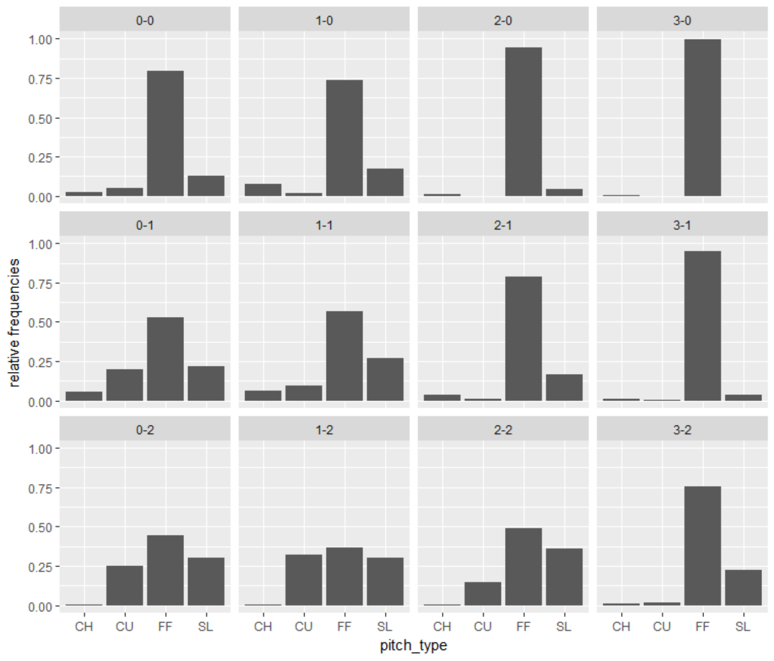

Figure 3: Pitch type relative frequencies by count, where CH, CU, FF and SL represent change-ups, curveballs, fastballs and sliders, respectively.

In the top right-hand corner of the grid, we see the case where Kershaw is down in the count, or has thrown more balls than strikes. In such a case, he is closer to walking the batter than he is to striking him out. As a result, we can see that he almost exclusively throws fastballs in an attempt to work his way back into the count. In the bottom left-hand corner of the grid, we see the case where Kershaw is up in the count, or has thrown more strikes than balls. In such a case, we see a good deal more variability in the type of pitch he will throw. Overall, this plot suggests that count would be a useful predictor in a pitch prediction model.

In addition to the current count, we wanted to incorporate information regarding the sequence of previous pitches into our model. In the MLB, there is this idea of “setting up a batter;” this phrase refers to a pitcher’s tendency to throw a particular sequence of pitches such that the terminal pitch in that sequence has a higher probability of success. For example, a talented pitcher may intentionally throw two fastballs high and inside, to “setup” a curveball low and away; after seeing two very fast pitches high and inside, the batter may not expect to see a slower pitch curving low and away from them. To incorporate some of this information into our model, we built a previous pitch type predictor, which is a categorical predictor with five levels denoting the type of previous pitch thrown. Again, to visualize the utility of such a predictor, we provide relative frequency bar charts below.

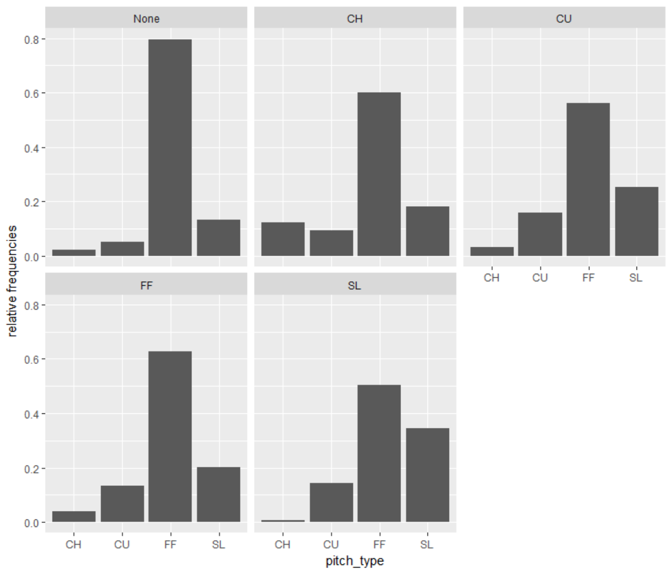

Figure 4: Pitch type relative frequencies by previous pitch type, where CH, CU, FF and SL represent change-ups, curveballs, fastballs and sliders, respectively.

We see a very interesting story in these plots. The plot in the top-left corresponds to the case where there is no previous pitch for a particular batter, or the very first pitch of the at-bat; in such a case, we see a fastball almost 80% of the time. And indeed, in most cases, we predominantly see fastballs, as general baseball intuition might suggest. However, we do think we were able to tease out some of the pitch sequencing relationships discussed above. The final panel of the grid corresponds to the case where the previous pitch was a slider. In this case, we see a slider occurs as the next pitch with far greater frequency than it does in any of the other four cases. This may suggest that Kershaw has a tendency to throw sliders in pairs; we see a similar relationship again with change-ups. Overall, these plots suggest that this predictor may also be of use in a pitch prediction model.

And finally, though not a predictor itself, we wanted to consider differences in pitch type probabilities across the different catchers that Kershaw has thrown to over his career. As mentioned above, the catcher is responsible for signaling the type of pitch to be thrown next; therefore, we may anticipate some differences in pitch type probabilities across catchers. To include this information, we choose to fit a hierarchical model, allowing each catcher to have a unique set of pitch type probabilities.

Between the count and previous pitch predictor, the anticipated model would have 16 regression coefficients to estimate per level of the response, for a total of 64 regression coefficients. Furthermore, each member of the hierarchy has a unique set of coefficients. Additionally, the variance-covariance matrix grows in dimension with the number of total regression coefficients to be estimated. Therefore, for primarily computational reasons, we restricted our model to consider only these components. Next, we formally specify the model we chose to fit.

3.2.2 Model Specification

We let i index the observation number, j index the level of the response and m index the catcher in the hierarchy, of which there were 11. Then the i^{th} observation for the m^{th} catcher is a 1×4 vector with a single element equal to one, denoting which pitch was observed.

$$\begin{aligned} \boldsymbol{y}_1, ...\boldsymbol{y}_{n_m}|\boldsymbol{\pi}_{im} &\sim \text{multnomial}(1, \boldsymbol{\pi}_{im}) \quad \pi_{ijm} = \frac{e^{\boldsymbol{x}_i \boldsymbol{\beta}_{jm}}}{\sum_{k=1}^{4} e^{\boldsymbol{x}_i \boldsymbol{\beta}_{km}}} \\ \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_{11}|\boldsymbol{\psi} &\sim \text{mvnormal}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \end{aligned} \tag{11}$$

The count predictor has 12 levels, and the previous pitch predictor has 5 levels; therefore, for each level of the response, there are 16 regression coefficients to be estimated, for a total of 64 regression coefficients per member of the hierarchy. The hyper-parameters in $\boldsymbol{\psi}$ require prior distributions. Due to the link function, Gibbs sampling is not available for the regression coefficients; instead, the Metropolis-Hastings algorithm is implemented. We can, however, specify semi-conjugate prior distributions on the hyper-parameters in $\boldsymbol{\psi}$ that allow for Gibbs sampling from the joint posterior distribution of $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. Therefore, our semi-conjugate priors are:

$$\begin{aligned} \boldsymbol{\mu} &\sim \text{mvnorm}(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0) \\ \boldsymbol{\Sigma} &\sim \text{inverse-Wishart}(\eta_0, \boldsymbol{S}_0) \end{aligned} \tag{12}$$

We chose $\boldsymbol{\mu}_0 = \boldsymbol{0}$, $\boldsymbol{\Lambda}_0 = 100\boldsymbol{I}_{64}$, $\eta_0 = 64$ and $\boldsymbol{S}_0 = \boldsymbol{I}_{64}$ such that our priors were weakly informative. Given these prior distributions, it can be shown that the full conditional distribution of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are:

$$\begin{gathered}
\boldsymbol{\mu}|\boldsymbol{\Sigma}, \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_m \sim \text{mvnorm}(\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n) \\
\text{where } \boldsymbol{\mu}_n = \boldsymbol{\Lambda}_n \left(\boldsymbol{\Lambda}_0^{-1}\boldsymbol{\mu}_0 + \boldsymbol{\Sigma}^{-1} \sum_{k=1}^{m} \boldsymbol{\beta}_k \right) \text{ and } \boldsymbol{\Lambda}_n = \left(\boldsymbol{\Lambda}_0^{-1} + m\boldsymbol{\Sigma}^{-1} \right)^{-1} \\
\boldsymbol{\Sigma}|\boldsymbol{\mu}, \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_m \sim \text{inverse-Wishart}\left(\eta_0 + m, (\boldsymbol{S}_0 + \boldsymbol{S}_{\boldsymbol{\mu}})^{-1}\right) \\
\text{where } \boldsymbol{S}_{\boldsymbol{\mu}} = \sum_{k=1}^{m} (\boldsymbol{\beta}_k - \boldsymbol{\mu})(\boldsymbol{\beta}_k - \boldsymbol{\mu})^T
\end{gathered} \tag{13}$$

Given these full conditional distributions, a Gibbs sampler can be implemented to sample from the joint posterior of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. To sample from the joint posterior distribution of the regression coefficients, the Metropolis-Hastings algorithm is implemented. The full description of this mixed sampling procedure is described in the next section.

3.2.3 The Block Metropolis-within-Gibbs Sampler

The framework for this sampling procedure was presented in section 2.1; here, we make one small addition to the method and then we formally describe the sampler. The Metropolis-Hastings step of this algorithm updates the 64 regression coefficients for a single catcher, one catcher at a time. Therefore, a vanilla Metropolis-Hastings step for this hierarchical model would propose a random walk on each of the 64 coefficients per catcher, then accept or reject all of these 64 new estimates per catcher at once.

This process can be overly restrictive and lead to additional computation time. It is not hard to imagine a case where a step in the wrong direction for a handful of the 64 coefficients could lead to the rejection of the entire proposal, despite the remaining coefficients making marginal steps in the right direction. To try and account for this problem, we chose to update the set of coefficients associated with each covariate in the model block-wise. That is, we first propose a random walk on the four intercept coefficients, accept or reject that set and then propose a walk on the 44 count coefficients, accept or reject them, and so on. Formally, the computational algorithm that we implemented looks like the following:

The Block Metropolis-within-Gibbs Sampler

Let $\boldsymbol{\mu}^{(s-1)}$ and $\boldsymbol{\Sigma}^{(s-1)}$ denote the current values of the hyper-parameters. Furthermore, let $\boldsymbol{\beta}_{.1}^{(s-1)}, ..., \boldsymbol{\beta}_{.m}^{(s-1)}$ denote the current estimates for the vector of regression coefficients for each of the members of the hierarchy and let $\boldsymbol{\beta}_{km}$ denote the set of regression coefficients associated with the k^{th} covariate in the regression model for the m^{th} member of the hierarchy. Finally, let $J(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(s-1)})$ denote a random walk distribution. Initialize $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta}_{.1}, ..., \boldsymbol{\beta}_{.m}\}$ at $\{\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}, \boldsymbol{\beta}_{.1}^{(0)}, ..., \boldsymbol{\beta}_{.m}^{(0)}\}$. The block Metropolis-within-Gibbs sampler generates $\{\boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}, \boldsymbol{\beta}_{.1}^{(s)}, ..., \boldsymbol{\beta}_{.m}^{(s)}\}$ from $\{\boldsymbol{\mu}^{(s-1)}, \boldsymbol{\Sigma}^{(s-1)}, \boldsymbol{\beta}_{.1}^{(s-1)}, ..., \boldsymbol{\beta}_{.m}^{(s-1)}\}$ in the following manner:

Gibbs step for $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$

1) sample $\boldsymbol{\mu}^{(s)} \sim p(\boldsymbol{\mu}|\boldsymbol{\Sigma}^{(s-1)}, \boldsymbol{\beta}_{.1}^{(s-1)}, ..., \boldsymbol{\beta}_{.m}^{(s-1)})$.

2) sample $\boldsymbol{\Sigma}^{(s)} \sim p(\boldsymbol{\Sigma}|\boldsymbol{\mu}^{(s)}, \boldsymbol{\beta}_{.1}^{(s-1)}, ..., \boldsymbol{\beta}_{.m}^{(s-1)})$.

```
Metropolis-Hastings step for $\{\boldsymbol{\beta}_{.1}, ..., \boldsymbol{\beta}_{.m}\}$
For $j$ in the $1:m$ members of the hierarchy,
    and for $i$ in the $1:K$ variables in the model,

      3) sample $\boldsymbol{\beta}_{.j}$, the regression coefficients for the $j^{th}$ member of the hierarchy.

          i) sample $\boldsymbol{\beta}^*_{ij} \sim J(\boldsymbol{\beta}_{ij}|\boldsymbol{\beta}^{(s)}_{1j}, ..., \boldsymbol{\beta}^{(s)}_{(i-1)j}, \boldsymbol{\beta}^{(s-1)}_{(i+1)j}, ..., \boldsymbol{\beta}^{(s-1)}_{Kj})$. Denote
             $\{\boldsymbol{\beta}^{(s)}_{1j}, ..., \boldsymbol{\beta}^{(s)}_{(i-1)j}, \boldsymbol{\beta}^*_{ij}, \boldsymbol{\beta}^{(s-1)}_{(i+1)j}, ..., \boldsymbol{\beta}^{(s-1)}_{Kj}\}$ as $\boldsymbol{\beta}^*_{.j}$ and
             $\{\boldsymbol{\beta}^{(s)}_{1j}, ..., \boldsymbol{\beta}^{(s)}_{(i-1)j}, \boldsymbol{\beta}^{(s-1)}_{ij}, \boldsymbol{\beta}^{(s-1)}_{(i+1)j}, ..., \boldsymbol{\beta}^{(s-1)}_{Kj}\}$ as $\boldsymbol{\beta}^{(s-1)}_{.j}$
          ii) compute the acceptance ratio
             $r = \frac{p\left(\boldsymbol{y}_j|\boldsymbol{\beta}^*_{.j}\right) p\left(\boldsymbol{\beta}^{(s)}_{.1}, ..., \boldsymbol{\beta}^{(s)}_{.(j-1)}, \boldsymbol{\beta}^*_{.j}, \boldsymbol{\beta}^{(s-1)}_{.(j+1)}, ..., \boldsymbol{\beta}^{(s-1)}_{.m}|\boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}\right)}{p\left(\boldsymbol{y}_j|\boldsymbol{\beta}^{(s-1)}_{.j}\right) p\left(\boldsymbol{\beta}^{(s)}_{.1}, ..., \boldsymbol{\beta}^{(s)}_{.(j-1)}, \boldsymbol{\beta}^{(s-1)}_{.j}, \boldsymbol{\beta}^{(s-1)}_{.(j+1)}, ..., \boldsymbol{\beta}^{(s-1)}_{.m}|\boldsymbol{\mu}^{(s)}, \boldsymbol{\Sigma}^{(s)}\right)}$
          iii) set $\boldsymbol{\beta}^{(s)}_{.j} = \begin{cases} \boldsymbol{\beta}^*_{.j} & \text{with probability } min(r,1) \\ \boldsymbol{\beta}^{(s-1)}_{.j} & \text{with probability } 1 - min(r,1) \end{cases}$.
```

This process results in a sequence of parameter estimates that converge to the true joint posterior distribution of $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta}_{.1}, ..., \boldsymbol{\beta}_{.m}\}$. In the next section, we apply this sampler to simulated data in a simulation study that explores the advantages of hierarchical modeling.

4 Simulation Study

Before discussing the analysis on the Kershaw data, we first consider a simulation study on multinomial hierarchical regression. To gain some confidence in our sampler, and to verify that the increased predictive capability of a hierarchical model justifies the added complexity of the hierarchical structure, we simulated hierarchical multinomial data and compared the predictive capability of a multinomial regression model and its hierarchical counterpart using a Bayes factor as a criterion. In this section, we discuss how we generated hierarchical multinomial data, describe the simulation study, define the Bayes factor, and evaluate the predictive capability of the hierarchical model.

4.1 Generating Hierarchical Multinomial Data

We consider a simple example for this simulation study. Suppose we are interested in predicting a response with three levels based on a single categorical predictor, which also has three levels. Further suppose that the relationship between the predictor and the response differs for three different groups in a hierarchy. In a regression context, we fit the following model:

$$\begin{aligned} \boldsymbol{y}_{1m}, ..., \boldsymbol{y}_{n_m m} | \boldsymbol{\pi}_{im} &\sim \text{multinomial}(1, \boldsymbol{\pi}_{im}), \quad \pi_{ijm} = \frac{e^{\boldsymbol{x}_i \boldsymbol{\beta}_{jm}}}{\sum_{k=1}^{J} e^{\boldsymbol{x}_i \boldsymbol{\beta}_{km}}} \\ \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_M | \boldsymbol{\psi} &\sim \text{mvnormal}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \end{aligned} \tag{14}$$

Here, i indexes the observation number, j indexes the level of the response and m indexes the member of the hierarchy. In an estimation context, our goal is to estimate the 3×3 matrix of coefficients for each group in the hierarchy.

To generate these data, we first specify $\boldsymbol{\mu}$, the hyper-parameter that describes the mean vector of the top level of the hierarchy, and $\boldsymbol{\Sigma}$, the variance-covariance matrix for that multivariate normal distribution. In our simulation, we randomly generated $\boldsymbol{\mu}$ by taking nine draws from a discrete uniform$(-1, 1)$ distribution and set $\boldsymbol{\Sigma}$ to be a 9×9 identity matrix. We then generated a set of regression coefficients for each of the three members of the hierarchy by taking three draws from a mvnormal$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution.

These regression coefficients were then used to generate the true matrices of probabilities for each of the members of the hierarchy, using the link function described in equation 14 above. Once we had these 3×3 matrices of probabilities for each member of the hierarchy, we used them to generate 300 observations for each member of the hierarchy by taking draws from a multinomial distribution with the appropriate vector of probabilities. These 900 observations were then used to fit both the hierarchical multinomial regression model and the model ignoring the hierarchy. Finally, we replicated this process to generate another 900 observations, based on the same true probability matrices, upon which to validate our estimates.

4.2 The Bayes Factor

In Bayesian statistics, a popular model comparison tool is known as the Bayes factor; this quantity is a measure of the evidence in the data for a particular model relative to another. It is the factor by which we multiply the priors odds ratio for the two competing models in order to calculate the posterior odds ratio for those two models. That is, given some data $\boldsymbol{y}$ and two competing models M_1 and M_2, the posterior odds in favor of model 2 is given by:

$$\frac{p(M_2|\boldsymbol{y})}{p(M_1|\boldsymbol{y})} = \frac{p(\boldsymbol{y}|M_2)}{p(\boldsymbol{y}|M_1)} \frac{p(M_2)}{p(M_1)} \tag{15}$$

The quantity in equation 15 is the Bayes factor; in the event where equal prior probability is placed on the two competing models, as is often the case, the Bayes factor is itself the posterior odds of model 2 to model 1 given the data. Using the law of total probability, the Bayes factor can be expressed as the following:

$$BF_{21} = \frac{\int p(\boldsymbol{y}|\boldsymbol{\theta}_2)p(\boldsymbol{\theta}_2)}{\int p(\boldsymbol{y}|\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_1)} \tag{16}$$

where $\boldsymbol{\theta}_i$ is the set of unknown parameters associated with model M_i. These integrals are often approximated using Monte Carlo integration techniques across the posterior samples (Hoff, 2010). It is worth noting that inherent to these calculations is a penalty for more complicated models; that is, the Bayes factor favors less complicated models if both models provide adequate descriptions of the data (Jefferys and Berger, 1992).

An extension of this standard Bayes factor to a cross-validation framework is described in *An Introduction to Bayesian Analysis: Theory and Methods* by Ghosh et al. (2010). In such a framework, the posterior samples based on the training dataset replace the prior distributions for the parameters under each model. That is, letting $\boldsymbol{y}_1$ denote the training dataset and $\boldsymbol{y}_2$ denote the validation dataset, the conditional Bayes factor comparing model 2 to model 1 is given by:

$$BF_{21}(\boldsymbol{y}_1) = \frac{\int p(\boldsymbol{y}_2|\boldsymbol{\theta}_2)p(\boldsymbol{\theta}_2|\boldsymbol{y}_1)d\boldsymbol{\theta}_2}{\int p(\boldsymbol{y}_2|\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_1|\boldsymbol{y}_1)d\boldsymbol{\theta}_1} \tag{17}$$

Again, these integrals are typically approximated using Monte Carlo methods across the posterior samples. In terms of practical application, Kass and Raferty developed a nice table (see table 1) of interpretations of Bayes factors on the log scale.

2log(BF)	strength of evidence
0 to 2	not worth more than a bare mention
2 to 6	positive
6 to 10	strong
> 10	very strong

Table 1: Table of interpretations of logged Bayes factors from Kass and Raferty (1995)

Table 1 provides a practical interpretation of the conditional Bayes factor discussed above. At a high level, the Bayes factor is just a ratio of the posterior likelihood under each model integrated across each parameter space. In this way, it gives us a complete description of the variability in our posterior estimates.

4.3 Results

We replicated this simulation 50 times, each time generating a new set of hyper-parameters from which to draw the regression coefficients for each group. We then fit both the multinomial regression model and its hierarchical counterpart to the 900 observations generated from each of these sets of regression coefficients. Finally, we calculated the conditional Bayes factor comparing the hierarchical model to its non-hierarchical counterpart, summarized in figure 5.

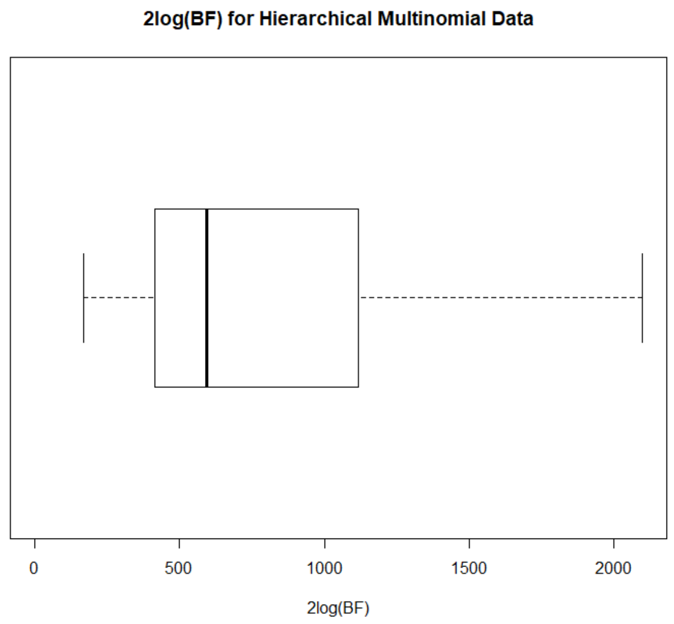

Figure 5: Boxplot of twice the logged conditional Bayes factor comparing the hierarchical model to its non-hierarchical counterpart.

Based on the plot, there appears to be very strong evidence that the hierarchical model provides a better description of the data in each of the 50 simulations than does the non-hierarchical model. From this simulation, we gain confidence in our sampler and also in the advantage of fitting a hierarchical model in the presence of hierarchical data. With this confidence in hand, we turned our attention to the Kershaw data.

5 Kershaw Analysis

In this section, we discuss the results of the Kershaw model. First, we discuss the set of models to which we compared the hierarchical model discussed above. We then assess the convergence of the samplers used to fit those models and conclude with a discussion of the results.

5.1 Model Comparison

The primary model of interest is the hierarchical multinomial regression model discussed in section 3.2., which we call model 1 for convenience. This model was motivated by the fact that we anticipated differences in the probability of observing each pitch across the 11 different catchers Kershaw has thrown to over his career. To evaluate this claim, we also fit the non-hierarchical counterpart to this model to serve as a baseline for comparison, which we call model two.

5.2 Convergence Diagnostics

We consider both graphical and numerical convergence diagnostics for these competing models. For each model, we consider a random sample of trace plots and the multivariate Gelman-Rubin statistic to assess the convergence of each sampler.

5.2.1 Assessing Model One Convergence

For model one, we ran three independent chains from random starting locations, and allowed each chain to run for 1,000,000 iterations. To conserve memory, we chose a thinning interval of 10, meaning we kept every 10^{th} iteration and discarded the rest. Of primary interest in this analysis are the predicted probabilities; therefore, we chose to look at traceplots on the probability scale, as opposed to the regression coefficients themselves. These traceplots are likely a better representation of the convergence of the algorithm, as there often exist identifiability issues on the regression coefficient scale in the multinomial logisitic regression framework. These issues can lead to regression coefficients that appear to "wander" in traceplots, despite the predicted probabilities remaining constant (Grün and Leisch, 2008).

For these data, there are 45 unique combinations of the covariates for which we calculated predicted probabilities; each of these combinations led to four traceplots, corresponding to the four types of pitches Kershaw throws. Therefore, for each member of the hierarchy, there are 180 traceplots that we could consider, for a grand total of 1,980 plots. Consequently, to evaluate the convergence of our sampler, we provide a few examples of randomly selected traceplots and rely more heavily upon numerical convergence diagnostics to assess the convergence of all model parameters.

Provided below are randomly selected traceplots corresponding to predicted probabilities for nine combinations of the covariates for A.J. Ellis, the catcher to whom Kershaw threw the most.

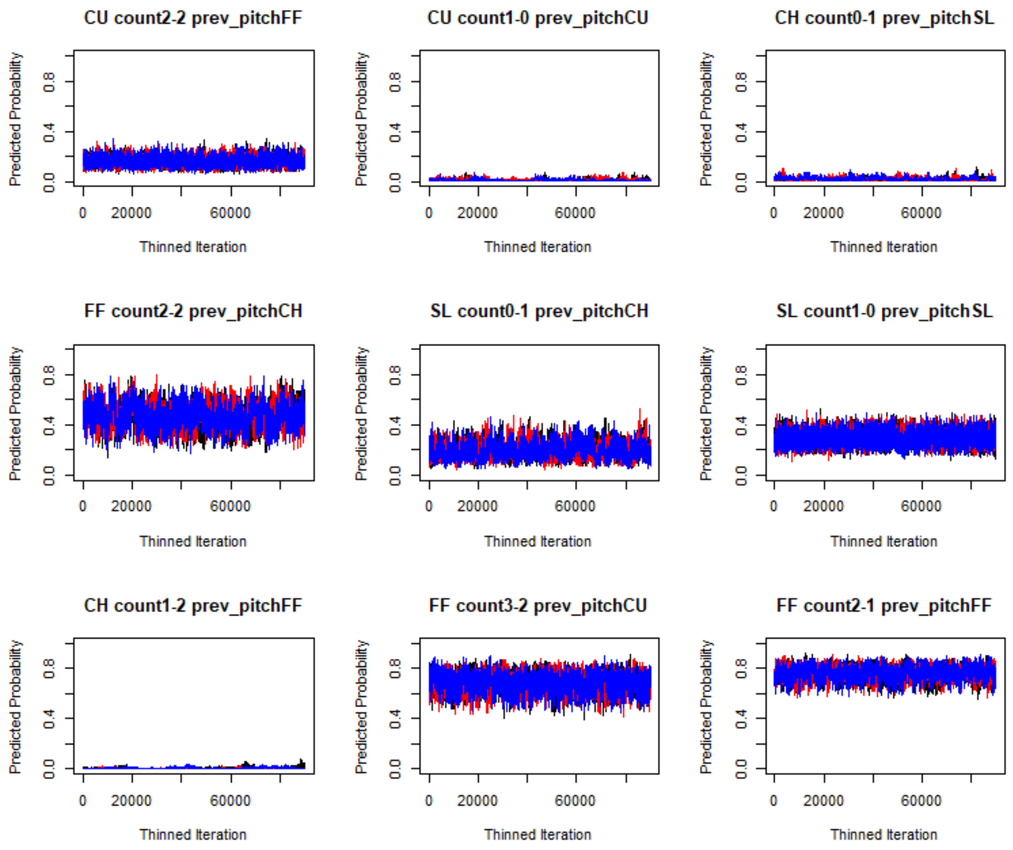

Figure 6: Randomly selected traceplots corresponding to predicted probabilities for nine combinations of the covariates for A.J. Ellis.

These traceplots display the "fuzzy caterpillar" like appearance discussed in section 2.1.4, suggesting that those parameter estimates have converged to their proper location in the target distribution. We do see some interesting artifacts in the plot corresponding to predicting a change-up on a 1-2 count given the previous pitch was a fastball (bottom left). However, we must keep in mind that probabilities are bounded below by 0; therefore, if an event is unlikely to occur, such as a change-up following a fastball on a 1-2 count, we might expect to see traceplots like we do above. Overall, these nine plots suggest that those predicted probabilities converged to their true location in the target distribution.

While these nine plots look good, there are 1,971 more to consider. To evaluate the convergence of all 1,980 parameters, we considered boxplots of the PSRFs for the 180 probability estimates for each of the 11 catchers in the hierarchy (see figure 7). For all 11 catchers, the bulk of each of the PSRFs are very close to 1, with none exceeding 1.5 suggesting that the sampler converged to the true target distribution. This, combined with the traceplots, gives us confidence in our estimated posteriors.

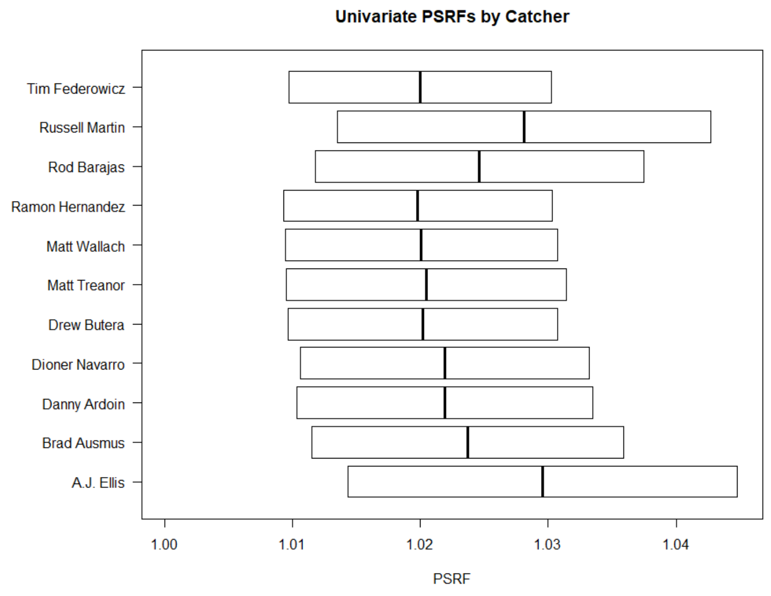

Figure 7: Boxplots of the potential scale reduction factors for the 180 predicted probabilities for each of the 11 catchers.

5.2.2 Assessing Model Two Convergence

We used a similar approach to assessing the convergence of model 2; however, we no longer had the hierarchical component to take into account. Therefore, there were only 180 parameter estimates to consider. Again, we randomly selected nine traceplots to consider, and looked at a boxplot of the PSRFs.

These nine traceplots all display the "fuzzy caterpillar" like appearance that we associate with convergence, and the PSRFs are all very close to 1. Overall, these diagnostics suggest that our sampler converged to the target distribution. With confidence in the convergence of our sampler for both models, we next consider the results of this analysis.

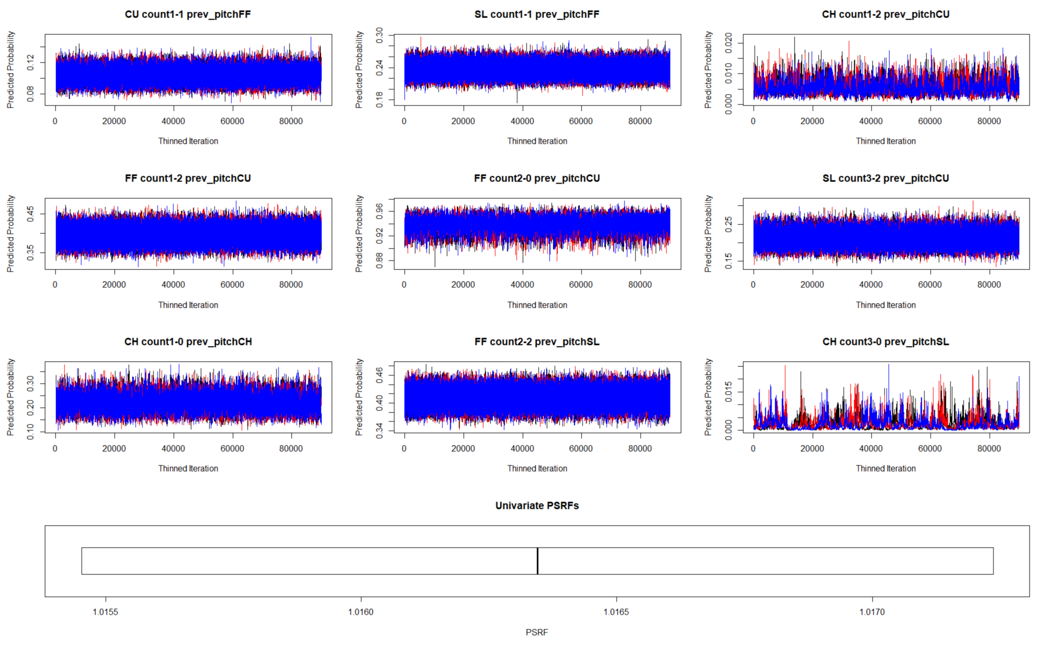

Figure 8: Randomly selected traceplots corresponding to predicted probabilities for nine combinations of the covariates for the non-hierarchical model, with a boxplot of the 180 potential scale reduction factors overlaid.

5.3 Results

In this section, we compare the hierarchical model to its non-hierarchical counterpart using the conditional Bayes factor discussed in section 4.2. To calculate this quantity, we evaluated the log likelihood for each observation in the validation dataset under both the hierarchical and non-hierarchical models. To account for the full variability in the posterior distribution of each model, we averaged the value of the log likelihood across all posterior samples for each observation; we then summed the log likelihood for each model across the 4,980 observations in the cross validation dataset for each model. Finally, we took twice the difference between these two approximated integrals to calculate $2\log(BF_{21}(\boldsymbol{y}_1)) = 140.33$ for these data.

Based on the table 1, this conditional Bayes factor provides extremely strong evidence that the hierarchical model is a better description of these data than the non-hierarchical model, which confirms our intuition that the probability of observing each of Kershaw's four pitches differs by catcher. In the next section, we consider the implications of this model, and consider improvements upon it.

6 Conclusion

In this section, we begin by revisiting the 2016 NLDS game discussed in the introduction. We then discuss the things we liked about this project and potential improvements that could be made to this model. We conclude with our finals thoughts on this research.

6.1 Example Revisited

We now return to the 2016 NLDS game we discussed in the introduction. In that game, Wilmer Difo found himself in a 1-2 count, having just seen a slider, with the Nationals' World Series aspirations on the line. In such a case, based on our model, the most probable pitch is a slider with probability 0.3600, closely followed by a curveball with probability 0.3364. The very next pitch Difo saw was a slider, which he fouled off, bringing him again to a 1-2 count with the previous pitch being a slider. Therefore, our model returned the same predicted probabilities. The next pitch Difo saw was a curveball, which he struck out on. Would this model have made a difference? We would like to think yes, but leave that judgment to the reader. In terms of practical application, the Nationals third base coach could have communicated the high slider and curveball probabilities to Difo, perhaps changing the outcome of that game.

6.2 Further Investigations

The evidence we saw in favor of the hierarchical model confirmed our intuition about differences across catchers, which was an exciting result. Furthermore, the overall predictive capability of the model surprised us, as evidenced by the example above.

This model will serve as a good foundation for more complicated models moving forward. In general, there are a number of improvements that could be made to this model. For example, we could start by considering additional predictors, such as the batting average of the opposing batter. Furthermore, we would really like to consider a better way of modeling the pitch sequencing aspect of baseball. To do so, rather than forcing it into the model matrix through the creation of a previous pitch predictor, we could think about allowing the probabilities of observing each of Kershaw's pitches to change over the course of a game by including a dynamic component to this model.

We could further extend that dynamic component to allow Kershaw's pitch type probabilities to change over the course of his career. In fact, because Kershaw threw to different catchers at different times in his career, part of the strong hierarchical signal we found in these data could be due to changing pitch type probabilities over time. Finally, we would like to consider a spatial prediction component in this model, which would allow us to not only predict the type of pitch to be thrown, but also the general location. These are ideas we look to further explore in the coming years.

6.3 Closing Thoughts and Acknowledgments

I would like to thank my advisors, Dr. Andrew Hoegh and Dr. Jennifer Green, for all the time they committed to the development of both this model and my understanding of statistics in general. Without their help, this project would not have been possible. I would also like to thank Dr. Mark Greenwood and the Montana State University Statistics faculty for their continued support over the course of this project and their dedication to excellence in teaching that allowed me to grow as a statistician over the course of my Master's degree.

7 References

Brooks, S. P. and Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455.

Chen, Z. and Kuo, L. (2001). A note on the estimation of the multinomial logit model with random effects. *The American Statistician*, 55(2):89–95.

Crasnick, J. (2018). Clayton Kershaw at 30: A decade of dominance as seen by the ace and those who know him best. `http://www.espn.com/mlb/story/_/id/22885658/clayton-kershaw-30-decade-dominance-seen-pitcher-know-best`. [Online; accessed 27-March-2018].

Efron, B. and Morris, C. (1977). Stein's paradox in statistics. *Scientific American*, 236(5):119–127.

Gelfand, A. E. (2000). Gibbs sampling. *Journal of the American Statistical Association*, 95(452):1300–1304.

Gelman, A. (2014). *Bayesian Data Analysis*. CRC Press, 3rd edition.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Ghosh, J. K., Delampady, M., and Samanta, T. (2010). *An Introduction to Bayesian Analysis Theory and Methods*. Springer.

Grün, B. and Leisch, F. (2008). Identifiability of finite mixtures of multinomial logit models with varying and fixed effects. *Journal of Classification*, 25(2):225–247.

Hoff, P. D. (2010). *A First Course in Bayesian Statistical Methods*. Springer.

Jefferys, W. H. and Berger, J. O. (1992). Ockham's razor and Bayesian analysis. *American Scientist*, 80(1):64–72.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.

Madigan, D., Genkin, A., Lewis, D. D., and Fradkin, D. (2005). Bayesian multinomial logistic regression for author identification. *AIP Conference Proceedings*, 803(1):509–516.

Nathan, A. M. (2011). The physics of baseball. `http://baseball.physics.illinois.edu/pitchtracker.html`. [Online; accessed 04-January-2018].

Novy-Williams, E. (2016). MLB teams allowed to use iPads in dugout all season. *The Chicago Tribune*. Published 30 March 2016.

Schmidt, M. S. (2017). Boston Red Sox used Apple watch to steal signs against Yankees. *The New York Times*. Published 05 September 2017.

Sievert, C. (2014). Taming PITCHf/x data with xml2r and pitchrx. *The R Journal*, 6(1):5–19.

8 Appendix - R Code

A script containing all code used for this analysis is available at

https://github.com/StrattonCh/WritingProjectCode