BAYESIAN LOGISTIC REGRESSION WITH JARO-WINKLER STRING COMPARATOR SCORES PROVIDES SIZABLE IMPROVEMENT IN PROBABILISTIC RECORD MATCHING

A Dissertation

by

DOMINIC ANDREW JANN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved By:

Co-Chairs of Committee,	Simon J. Sheather
	Michael Speed
Committee Members,	Jeffrey D. Hart
	Jin-Whan Jung
	Edward Murguia
	Suojin Wang
Head of Department,	Simon J. Sheather

December 2012

Major Subject: Statistics

ABSTRACT

Record matching is a fundamental and ubiquitous part of today's society. Anything from typing in a password in order to access your email to connecting existing health records in California with new health records in New York requires matching records together. In general, there are two types of record matching algorithms: deterministic, a more rules-based approach, and probabilistic, a model-based approach. Both types have their advantages and disadvantages. If the amount of data is relatively small, deterministic algorithms yield very high success rates. However, the number of common mistakes, and subsequent rules, becomes astronomically large as the sizes of the datasets increase. This leads to a highly labor-intensive process updating and maintaining the matching algorithm. On the other hand, probabilistic record matching implements a mathematical model that can take into account keying mistakes, does not require as much maintenance and overhead, and provides a probability that two particular entities should be linked. At the same time, as a model, assumptions need to be met, fitness has to be assessed, and predictions can be incorrect. Regardless of the type of algorithm, nearly all utilize a 0/1 field-matching strucure, including the Fellegi-Sunter algorithm from 1969. That is to say that either the fields match entirely, or they do not match at all. As a result, typographical errors can get lost and false negatives can result. My research has yielded that using Jaro-Winkler string comparator scores as predictors to a Bayesian logistic regression model in lieu of a restrictive binary strucutre yields marginal improvement over current methodologies.

ACKNOWLEDGMENTS

There are many people worthy of my gratitude for the completion of this dissertation. First and foremost, I must thank Erika Delgado Hernandez, my wife, for having put up with me while I've been busily working away on this. Her patience with me, as well as her support of me, made the rest of the process a bit more bearable. In the middle of this dissertation, I became a father, and Erika has been an exceptional mother to her in my absence. Every time I see Sofia, I am further reminded of what I am doing. My family comes first and foremost in my life.

I also must acknowledge the impact my parents, Cheryl and Andrew Jann, have had in my life. They fostered an environment which allowed me to grow and learn. They provided me with many opportunities both in education and in life. I gained an immense understanding of mathematics from my dad, who has a great ability to analyze any situation. My personality stems from my mother, a woman who knows what she wants and how to get it done. Between them, I left the nest with all the tools I needed to succeed.

Also, I must thank Dr. Simon J. Sheather and Dr. Jeffrey D. Hart, whose guidance and encouragement have propelled me to the finish line with minimal distraction. My weekly meetings have been met with much praise and direction from these individuals, without whom I would be desperately lost in my research. In addition, I would like to thank Dr. Jin-Whan Jung and Dan Kelly, at SAS Institute Inc., for providing me not only the topic itself, but the environment in which to continue and complete my work.

In addition, I must thank Dr. Michael Speed, who was not only the Co-Chair of my Dissertation Committee, but was also Co-Chair of my Master's Committee, my professor for two classes, and the Director of Online Learning, where I worked for my entire tenure at Texas A&M University. Beyond all that, he provided the bridge that has landed me at SAS, and has played a very parental role in my development, both professionally and personally.

Lastly, I have made a lot of friends in and around the Texas A&M University campus. Each of them has provided me with something different in my educational journey. From

late nights programming, to late nights socializing, my friends provided me with a break from the monotony of graduate school. I wish them all well in their endeavours.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Record matching has always played an important role in our society. It can be utilized directly when you input a username and a password to access an account. The database has to take the information you provide and compare that to all the available records to determine if there is a match. If your entry matches with what is in the database, then you are granted access. If there is no match, then you usually get the response that the password is incorrect or that no such username exists.

We have all encountered situations such as these, where we would like, even if we made a mistake, the software to recognize the information as ours. However, it would obviously not be ideal to match similar entries for account access. Hackers have a field day as it is with accessing our accounts, so why give them more ammunition to work with? But, on the other hand, there are many places where slight mistakes in data entry should not necessarily result in a flat-out non-match.

For example, suppose you have lived in California for 20 years and then you move to New York. It would be highly advantageous for "new" information in New York to accurately match to your "current" information in California. However, what if there is a spelling mistake, and the system in place does not have a rule for that? For example, the name "Dominic" can be spelled "Dominik" or even "Dominique". Now, handling this particular spelling variation would not be difficult. The problem occurs when you deal with such a large number of potential errors. Almost every first name could be spelled in multiple ways, depending on the country or even the region within the country. Predicting how a particular name will be spelled can only occur if you are lucky (or experienced) enough to know in advance how each name is spelled throughout the area. Last names can be a nightmare to spell (in any country), addresses often have severe standardization issues, and business names could be abbreviated. What's worse still is that the number of common mistakes, and subsequent rules, becomes astronomically large as the sizes of the datasets increase. This leads to a highly labor-intensive process updating and maintaining the algorithm.

This is where probabilistic record matching (PRM) comes into play. No need to worry about a series of rules, having to modify algorithms when a new exception occurs, or lack of scalability. A little time investment in formulating an adequate mathematical model plus estimating a few key probabilities leads us to a much more easily scalable algorithm.

PRM models keying mistakes, spelling variations, business abbreviations and the like in providing a probability that two entities should be linked. In looking at any arbitrary dataset, suppose that there are 5 fields: first name, last name, day of birth, month of birth, and year of birth. If we further suppose that the probability of a typographical error is .10 for first name and .05 for all the other fields (which is reasonable according to Winkler (2006)) then by Boole's Inequality, the probability of a mistake in any of these fields is less than or equal to .30. In other words, unless we can model the typographical errors correctly, we can miss up to 30% of matches. If a particualr dataset has a million pairings, that means about 300,000 false non-matches.

Aside from the obvious issues with mathematical models in general, there are other issues specific to record matching itself. For example, name and address standardization is a paramout issue in this field. Addresses can be reported in many different ways, with or without abbreviations, with or without specific street, road, drive declarations, and so on. If the data is time-dependent, addresses may be vastly different if the individual has moved, or has changed his or her last name as a result of marriage. Business names can be abbreviated, or even differentiated by area. These issues, while detrimental to any predictive model, are somewhat controllable by comparison. Typograhical mistakes are a relative nightmare because of their erratic nature. Some typographical mistakes are easily predictable (such as the "Dominic" vs "Dominik" example from before). However, keying mistakes are more difficult to predict. For example, there is no deterministic rule for saying that "Brown" and "Brosn" are the same person, though there are models in place to take into account key location. Lastly, some mistakes are just too severe for any model to catch, regardless of how complicated the model or how many clerical reviewers you have

at your disposal. In addition, one would not want to catch certain situations in which the information is badly damaged due to the propensity of subsequent false matches.

It bears noting that there are many aspects of PRM which are worthy of research. For example, estimating false matching rates are of paramount importance. It is necessary to be able to adequately assess any given procedure and false-match rates tend to be the gold standard for accomplishing this. The most foolproof way of achieving this would be to know how many of the possible pairs are matches and non-matches. Then, given the output of the record matching algorithm, you can compare the number of matches from the algorithm with the actual number of matches and report on how many matches were actually caught. This could be thought of as the sensitivity, or recall rate, of the algorithm, or how many of the actual matches were captured by the algorithm. However, as you can easily see, we do not actually know the matching status of all the pairs in a particular comparison. If we did, then none of this would be necessary. There are numerous procedures (Fellegi and Sunter (1969), Belin and Rubin (1995), among many others) that deal with the estimation of false-match rates.

Another component of PRM which is of high research interest is the topic of blocking. Given a million records in file A and another million records in file B, you can easily see how computationally difficult the whole matter of matching becomes. The idea of matching 10^{13} records can be a daunting task for any computer...and these are small in comparison to the types of datasets the US Census deals with. So, blocking schemes are designed that group together observations which have something in common. For example, we could group observations together that were born in the same year, whose last names start with the same letter, who live in a particular geographic region, and so on. There are many variables that can be used in blocking, just as there are many schemes for blocking, such as bi-gram indexing and canopy clustering with TFIDF (Term Frequency / Inverse Document Frequency). What is noteworthy regarding blocking is that whatever is used as the blocking key, i.e., first letter of last name, accuracy of that key is paramount to the success of the subsequent record matching scheme.

However, while both of these subtopics to record matching are highly useful and definitely in need of additional research, this discussion will be focusing solely on developing a new methodology for the matching scheme itself. It is assumed that the methods developed here can be applied to any already-blocked observations that have been generated. In terms of measuring the effectiveness of the proposed algorithm, this dissertation will be focusing on recall, precision and F-Score, which is a weighted average of the prior two assessments.

The rest of this dissertation will be organized as follows. Section 2 will consist of a background into PRM, along with a discussion of notation and assumptions. Section 3 is a literature review with respect to current record matching methodology. In addition, you will see a glimpse into why I have chosen the logistic regression approach as the backbone of my methodology. Section 4 will outline the theory behind the mathematical model being developed. Section 5 will outline the details behind the simulation itself, including the software, the variables, and incorporating missing and erroneous data in order to mimic reality. Section 6 will cover implementation of both the baseline (Fellegi-Sunter), and the proposed Bayesian Logistic Regression approach. Section 7 will discuss the results of the comparison, both in terms of Bayesian Logistic Regression versus the Fellegi-Sunter and using string comparators versus strict binary matching structures. Section 8 will describe some potential future research to be done in this field. Section 9 will include a summary and some concluding remarks.

2 BACKGROUND

Record matching in and of itself is a relatively simple concept. Given two datasets and one entry in the first dataset, can you find a match in the second dataset? We determine matching records by looking at coincident fields between the records, such as first name, last name, address, etc. Deterministically, this can be done by incorporating rules to handle the many possible variations in the data. Probabilistically, we incorporate a mathematical model in an attempt to extract a probability of two records matching conditional on the comparison attributes. Within both these realms, most literature focuses strictly on match/non-match structures with respect to the fields. For example, "Dominic" = "Dominic" and the comparison attribute for First Name would be set to 1. However, "Dominic" and "Dominik" do not match entirely, so the comparison attribute for First Name in this case would be set to 0. Instead of two fields matching absolutely or not matching at all, we can use string comparators to output a similarity index for two fields. From these outputs, we can build a model which will output the probability that two particular entities match given the values of the comparison attributes.

What follows is a summary of the theory for PRM presented by Fellegi and Sunter (1969).

2.1 Notation

Basically, you have two datasets, call them A and B, and you wish to partition all possible comparisons ($A \times B$) into two mutually exclusive subsets:

$$M = \{(a, b) : a = b, a \in A, b \in B\} \tag{1}$$

and

$$U = \{(a, b) : a \neq b, a \in A, b \in B\} \tag{2}$$

where M constitutes the set of all *matches* and U constitutes the set of all *unmatches* or *non-matches*.

How do we go about deciding whether or not two particular records reference the same entity (i.e., match)? Well, we look at the fields that are common to both datasets (like First Name, Last Name, Gender, etc.) Given these common comparison components, we achieve our matching decision via a component by component comparison vector, commonly denoted by γ:

$$\gamma^j = [\gamma_1^j, \ldots, \gamma_K^j] \tag{3}$$

where

$$\gamma_i^j = \begin{cases} 1 & \text{if field } i \text{ agrees for the } j\text{th pair} \\ 0 & \text{if field } i \text{ does not agree for the } j\text{th} \end{cases} \tag{4}$$

This is the most simplistic version of the comparison vector: either the fields match completely ($\gamma_i^j = 1$) or they do not match at all ($\gamma_i^j = 0$).

Now, each of these components, γ_i, being an event, can have a conditional probability attached to it, see Fellegi and Sunter (1969). In fact, there are two:

$$m(\boldsymbol{\gamma}) = P(\boldsymbol{\gamma}|(a,b) \in M) = P(\boldsymbol{\gamma}|M) \tag{5}$$

$$u(\boldsymbol{\gamma}) = P(\boldsymbol{\gamma}|(a,b) \in U) = P(\boldsymbol{\gamma}|U) \tag{6}$$

The first probability, $m(\boldsymbol{\gamma})$, represents the probability of observing a particular vector conditional on the fact that the two records associated with that vector are a match. This probability is a proxy for measuring the reliability of the data being compared. It is desirable that this probability be large for agreement patterns, i.e. patterns where most components are 1, though this is not always the case. The second probability, $u(\boldsymbol{\gamma})$, represents the probability of observing a particular vector conditional on the fact that the two

Table 1: Subset of data used in calculating m- and u-probabilities.

A_Fname	A_Lname	A_Gender	B_Fname	B_Lname	B_Gender	Vector
Chris	Johnson	F	Chris	Johnson	F	111
Chris	Johnson	F	Robert	Ordway	M	000
Chris	Johnson	F	Kathryn	Lowry	F	001
Robert	Mills	M	Robert	Ordway	M	101
Joan	Ordway	F	Robert	Ordway	M	010
Charles	Hernandez	M	Billy	Hernandez	M	011
Chris	Johnson	F	Chris	Johnson	M	110
Robert	Mills	M	Robert	Simons	F	100
⋮	⋮	⋮	⋮	⋮	⋮	⋮

records associated with that vector are not a match. In terms of agreement configurations, this probability is a proxy for measuring random matching chances. Obviously, we would like these probabilities to be pretty low for agreement configurations, though in fields where only a select number of values are possible (such as 'male' and 'female' for gender), a higher u-probabilty is expected. In fact, in many algorithms, these u-probabilities are assigned as simply 1 over the number of possible field values. So, in the case of gender, with only two values (Male/Female), $u(\gamma_{Gender})$ would be $1/2 = 0.5$.

These conditional probabilities are paramount to the Fellegi-Sunter algorithm of PRM, which many current-day algorithms either stem from or are compared with. For a better insight into these vector probabilities, let us consider the following example. You have three attributes/components (First Name, Last Name and Gender) and you wish to estimate the m- and u-probabilities for the data listed above, assuming you know matching status (which we usually do not know). In the strictly 0/1 matching scheme, there are $2^3 = 8$ possible vectors. Table 1 displays a subset of the type of data we would be looking at.

As you can see in Table 1, we have an example of each of the 8 possible scenarios that can occur. We can break these down further into sub-categories. If these were the only three categories involved, we would declare the first category to consist of the obvious matches (row 1) and non-matches (row 2). It would be great if everything was so cut and dried. The second category consists of mostly random matching, with gender much more

likely to randomly match given that there are only 2 possible values (M and F). Row 3 shows us two females who share the same gender field, but nothing more. Row 4 shows two males with the same first name, Robert. There is less variability in first name than in last name, so we would expect random matching of last name to be rare. The third category (Rows 5 and 6) shows us this situation as well as last name and gender matching. Likely reasons for this would be that they are either related (brothers, cousins, husband and wife, etc.) or that the last name is common, (such as Jones or Smith). These situations are particularly challenging because it usually requires some sort of manual review to sort this out without additional information. It can be quite difficult to predict exactly how some mistakes occur. The last category is the most problematic: where first and last name matches but gender does not. In situations like this, we could be dealing with situations where a unisex first name is possible (like Chris in Row 7). Another possible reason is that Robert could be Roberta (Row 8), but the person inputting the information accidentally left off the last letter.

Going back to the first row, where it seemed we had an obvious match, the following should be noted. Being a relatively common name, there could be many people named "Chris Hernandez" in the world, each living at a different address. Having address would help substantially in differentiating between these potential false-matches. An additional problem would occur if the first dataset came from 2005 and the second one came from 2012. In that instance, the address may be different because "Chris Hernandez" moved. There are a lot of elements to consider when attempting to match two datasets together, and what works for one pair of datasets may generalize very well.

Now, to demonstrate how we go about calculating m- and u-probabilities based on the comparison vector components AND known matching status, we have Table 2.

In Table 2, the comparison vectors are given on the left, the number of pairs corresponding to that vector are in the 2nd column, and the number of true matches and true non-matches are given in the middle two columns. Granted, in reality, we do not know this information, which makes estimation of the last two columns, the m- and u-probabilities, a

Table 2: Calculation of m- and u-probabilities.

Comparison vector (γ)	n_i	Matches	Non-Matches	$m(\gamma)$	$u(\gamma)$
111	10	10	0	$10/10 = 1.00$	$0/150 = 0.00$
110	1	0	1	$0/10 = 0.00$	$1/150 = 0.01$
101	1	0	1	$0/10 = 0.00$	$1/150 = 0.01$
100	1	0	1	$0/10 = 0.00$	$1/150 = 0.01$
011	1	0	1	$0/10 = 0.00$	$1/150 = 0.01$
010	1	0	1	$0/10 = 0.00$	$1/150 = 0.01$
001	68	0	68	$0/10 = 0.00$	$68/150 = 0.44$
000	77	0	77	$0/10 = 0.00$	$77/150 = 0.51$
Totals	160	10	150	1.00	1.00

bit of a challenge. Based on the dataset used to build Table 1, and the matching statuses, we see that calculating m- and u-probabilities is quite simple, just like calculating a conditional probability back in introductory statistics courses. The trick is, we do not know the matching status, which means we can only estimate, not calculate, $m(\gamma)$ and $u(\gamma)$.

Once we have these conditional probabilities, methods for obtaining them forthcoming, then weights are calculated based on a likelihood ratio statistic. The weights are calculated as follows:

$$w_i = \frac{m(\gamma_i)}{u(\gamma_i)} \text{ for } i = 1, ..., K. \tag{7}$$

These weights are further aggregated to create a composite score representing the probability that two particular records refer to the same entity:

$$\mathbf{w} = \sum_{i=1}^{K} w_i. \tag{8}$$

If the composite score is above a certain point, then we claim the two records match (labeled as A_1 in Fellegi and Sunter (1969)). If the composite score is below a different point, then we claim that the two records do not match (labeled as A_3). If the score is between these two thresholds, then we call them a possible match (A_2) and we can either match them, not match them, or perform additional review. Thus, we have a linkage rule,

or decision function, which assigns probabilities to each of the three decisions:

$$d(\gamma) = \{P(A_1|\gamma), P(A_3|\gamma), P(A_2|\gamma)\} \tag{9}$$

where the three probabilities are constrained to sum to 1.

As in any classification scheme, there are two potential types of mistakes that can be made. The first of these, called a Type I error, occurs when we match two entities together that should not be matched together (i.e., *false match*) and is denoted by:

$$\mu = P(A_1|U) = \sum_{\gamma \in \Gamma} u(\gamma) P(A_1|\gamma). \tag{10}$$

The second error, Type II, occurs when we do not match two entities that should have been matched (i.e., *false non-match*) and is denoted by

$$\lambda = P(A_3|M) = \sum_{\gamma \in \Gamma} m(\gamma) P(A_3|\gamma). \tag{11}$$

The Fellegi-Sunter algorithm is deemed optimal in the sense that, given the probabilities of false match and false non-match, we minimize the amount of clerical review performed post-algorithm. As you will see, each algorithm is "optimal" according to some metric.

2.2 Assumptions

Within the framework of the Fellegi-Sunter algorithm, in which much of this notation is based, there are two assumptions involved in the calculation of the weights:

- conditional probability distributions, $m(\gamma)$ and $u(\gamma)$, must be known (or at least estimated)
- probability distributions, $m(\gamma)$ and $u(\gamma)$, must also be statistically independent conditional on matching status Fellegi and Sunter (1969)

2.2.1 Known Conditional Probability Distributions

The first assumption is that the conditional probability distributions must be known (or at least estimated). This can be relatively easily handled in the following ways:

- Assign the probabilities (Verykios et al. (2003))
- Estimate the probabilities (Fellegi and Sunter (1969))
- Apply a prior distribution (Judson (2006))

The first thing we could do would be to assign the probabilities to each of the fields. There are a total of $2K$ conditional probabilities that would have to be assigned (2 for each field we are comparing: the m- and the u-probability). If a domain-expert is available, perhaps this could work. However, it is believed that this methodology is too subjective and requires too many inputs and, as such, this method is usually avoided.

The second option would be to estimate these conditional probabilities. Within this framework, there are two routes we can go.

- Estimate probabilities via pre-processing of external datasets
- Estimate probabilities via internal processing of configuration frequencies

The first of these approaches involves having a large set of already classified comparisons. One of the great challenges, however, has been ascertaining a large enough dataset from which we can adequately estimate all $2K$ probabilities. For organizations like the Bureau of the Census, where record linkage is a regular part of their operations, these external datasets do exist and help substantially in building a model for subsequent data. In most situations, however, the models are being built on the very same data that we are trying to match. The second method, the one I am comparing my algorithm to, has been around for over 40 years, presented by Fellegi and Sunter (1969). This involves vector frequency in actually estimating these probabilities from within the dataset itself.

The last method is the one which this dissertation is based upon. Using informative priors, one could theoretically tailor the prior distributions for each parameter if domain-specific information and/or previous record matching results were made available. Once we have the posterior distributions, we can do a variety of things with them. We could obtain their expected values and feed these numbers into a likelihood-ratio-statistic-based weighting algorithm and make matching decisions.

2.2.2 Statistically Independent Conditional Distributions

The second assumption is that the conditional distributions must be statistically independent given matching status. This is what allows us to treat the likelihood function as a mere product of pdfs, which is what subsequently allows us to take the log of the product and sum up the weights in the calculation of our composite score. Basically, it states the following:

$$m(\gamma) = m_1(\gamma^1) \times m_2(\gamma^2) \times \cdots \times m_K(\gamma^K) \tag{12}$$

$$u(\gamma) = u_1(\gamma^1) \times u_2(\gamma^2) \times \cdots \times u_K(\gamma^K) \tag{13}$$

where $m(\gamma)$ and $u(\gamma)$ are defined as in (5) and (6) respectively. Thus, we are stating, in simpler terms, that the conditional probability of a particular vector given matching status is equal to the product of the conditional probabilities of the individual components given matching status. For example, the probability that first name, last name and gender agree given that the records match is equal to the probability that first name agrees given the records match times the probability that last name agrees given the records match times the probability that gender matches given the records match.

This simplifying assumption is a bit more problematic because it is often violated in practice. For example, consider address and last name data. If two records are supposed to match, then the fact that addresses match would have an impact on whether last name matches. However, Herzog et al. (2007) states that parameters estimated when the condi-

tional independence assumption is violated may still yield accurate decision rules in many situations.

However we go about obtaining these probabilities, they are paramount for the implementation of the PRM scheme presented by Fellegi and Sunter, along with many others based on it. Once we have these probabilities, we can calculate agreement and disagreement weights, which will be subsequently summed together to give us a composite score, a proxy for estimating the probability that two particular entities should be linked.

3 LITERATURE REVIEW

This Section consists of two parts: (1) a general glimpse into the world of record matching and the different methodologies available and (2) an in-depth look at the Bayesian approach I am modifying.

3.1 The World of Record Matching

In terms of record matching, there are many different algorithms that have been employed. The four principal directions, highlighted in Winkler (2006), are as follows:

- Conditional Independence (naive Bayes)
- Logistic Regression
- Boosting
- Support Vector Machines (SVM)

3.1.1 Conditional Independence

Those methods which incorporate the conditional independence assumption can essentially be summarized in terms of the Fellegi-Sunter framework discussed above. In the Appendix, you will find explicit equations which allow for implementation of the Fellegi and Sunter (1969) algorithm. While we have already discussed how the conditional independence assumption is frequently violated in practice, it is generally accepted that Naive Bayes is computationally much faster and more straightforward than SVM, boosting, or logistic regression, according to Winkler (2006).

In record linkage under conditional independence, the weights for individual field agreements is summed to obtain the total agreement weight associated with that record pair. The weighting of this type of record linkage is a straightforward linear weighting. According to Winkler (2006), in theory, SVM and boosting should outperform the conditional independence model because the weights w are optimal for the type of linear weighting

used in the decision rule. One reason that SVM or boosting may not improve much is that record linkage weights that are computed via an EM algorithm also tend to provide better separation than weights computed under a pure conditional independence assumption. Additionally, the conditional independence assumption may not be valid. If conditional independence does not hold, then the linear weighting of the scores is not optimal.

Another noteworthy component with respect to the independence assumption is that we are not only dealing with the independence of the vector conditional on the matching status, but the independence of the typographical errors present in the records. In many situations, if a typographical error exists in any one field of a particular record, it stands to reason that the probability of additional errors would be higher.

3.1.2 Logistic Regression

Logistic Regression is a regression-based statistical methodology implemented to predict the probability of being in a particular class (e.g., matching records) based on a series of quantitative and/or categorical predictor variables. For a binary response variable Y (e.g., matching status), $\pi(\mathbf{x})$ denotes the "success" probability at value $\mathbf{x}$ (e.g., the comparison component vector).

Regardless of the sampling mechanism, the logistic regression model may or may not describe a relationship well. In one special case, it does necessarily hold. According to Agresti (2007), suppose the distribution of X for subjects having $Y = 1$ is normal $N(\mu_1, \sigma)$, and suppose the distribution of X for subjects having $Y = 0$ is normal $N(\mu_0, \sigma)$; that is with different means but the same standard deviation. Then, a Bayes theorem calculation converting from the distribution of X given $Y = y$ to the distribution of Y given $X = x$ shows that $P(Y = 1|x)$ satisfies the logistic regression curve. For that curve, the effect of x is $\beta = \frac{\mu_1 - \mu_0}{\sigma^2}$. If the distributions of X are bell-shaped but with highly different spreads, then a logistic model containing also a quadratic term (i.e., both x and x^2) often fits well. In that case, the relationship is not monotone. Instead, $P(Y = 1)$ increases and then decreases, or the reverse. This part is relevant here because in Belin and Rubin

(1995), the weights are transformed to be treated as normal distributions. As such, the logistic regression methodology can be applied to these transformed values.

The following is an example of how logistic regression has been applied to the record linkage problem. In fact, Harville and Moore (1999) say that the Fellegi-Sunter approach and an iterative logistic regression are essentially equivalent. What follows is some underlying theory into Harville's method. Understanding this is important to understanding the method itself.

Let $B_i(B_i > 0)$ be the bonus assigned for agreement of values of the i-th field. Let $P_i(P_i < 0)$ be the corresponding penalty for disagreement. Let A = set of i's where the values of the ith fields agree. We can write the final score as

$$SCORE = \sum_{i \in A} B_i + \sum_{j \notin A} P_j. \tag{14}$$

Theorem 1: Let $X_i = +1$, when the values of the ith field agree, and $X_i = -1$ otherwise. Then there exist unique coefficients a_i, and translations t_i, such that

$$SCORE = \sum_{i=1}^{n} a_i * (X_i + t_i) \tag{15}$$

Corollary 1: Under the conditions in **Theorem 1**, (16) can be written as

$$SCORE = a_0 + \sum_{i=1}^{n} a_i X_i \tag{16}$$

with $a_0 = \sum_{i=1}^{n} a_i t_i$ and the a_i unique.

Step 1: Pair all potential linkages and develop agreement patterns in terms of the $X_i(X_i = \pm 1)$; then define $Y = +1$, if the unique identifiers agree, and $Y = -1$, otherwise. For each pair, create the vector $(X_1, X_2, ..., X_n, Y)$.

Step 2: Use logistic regression to model the likelihood of Y as a function of X_i. This will result in an equation similar to (14).

From **Corollary 1**, we know that

$$SCORE = a_0 + \sum_{i=1}^{n} a_i X_i \tag{17}$$

with the a_i unique for the set of Fellegi-Sunter bonuses and penalties. The same corollary guarantees that there exist unique t_i's such that

$$a_0 = \sum_{i=1}^{n} a_i t_i. \tag{18}$$

If we can determine the t_i, we can combine (18) and (19) and then use Theorem 3.1 to obtain the bonuses and penalties

$$\begin{aligned} B_i &= a_i \times (+1 + t_i) \quad \text{and} \\ P_i &= a_i \times (-1 + t_i) \quad \text{for } i = 1, 2, \ldots n \end{aligned} \tag{19}$$

Step 3: Suppose that we select a proper subset of the X_i and model the logistic regression. We would get

$$SCORE^* = a_0^* + \sum_{i=1}^{n} a_i^* X_i \tag{20}$$

where $a_i^* = 0$ when the ith matching variable has been dropped from the model. Analogous to (19), we can assume that there exist unique t_i^* such that

$$a_0^* = \sum_{i=1}^{n} a_i^* t_i^*. \tag{21}$$

Note that when $a_i^* \neq 0$ in (22), it will probably differ from that of a_i in (19). Note also that the t_i^* in (23) will probably different from the t_i in (20). Assume that we carefully choose a subset such that $a_i^* = a_i$ for all values i where the ith matching variable appears in the subset. Under this condition, we will assume $t_i^* = t_i$. This allows us to eliminate the "*" on the t_i's in (23), so

$$a_0^* = \sum_{i=1}^{n} a_i^* t_i. \tag{22}$$

Eqs. (20) and (24) now define a system of 2 linear equations in n unknowns (the t_i's). If we select a different proper subset, which generate coefficients $a_i^* = a_i$, we will be able to create a third linear equation for this system. We can continue to iteratively select proper subsets, model and add equations to the system until we have generated n linearly independent equations. For a set of 5 predictors, there are 31 possible subsets, so getting the 5 necessary equations to solve for the t_i's should not be a challenge.

Step 4 Solve the system of equations to find the unique values for each t_i. Then substitute into (21) to obtain the desired bonuses and penalties.

The advantages of this method are that logistic regression is a much easier method to interpret than the complicated algebraic manipulations utilized in the Fellegi-Sunter method. As logistic regression is a widely-used statistical procedure, teaching this method would be less daunting due to its higher level of familiarity. According to Harville and Moore (1999), the method is easy to use, gives good results, and the resulting parameters are consistent with the principles outlined by Fellegi and Sunter.

While Winkler (2006) states that logistic regression is outperformed by support vector machines and boosting, Ng and Jordan (2002) have demonstrated that logistic regression is essentially an approximation of SVM. Based on this statement and Harville's assessment, it seems that SVM, logistic regression, and the Fellegi-Sunter (or conditional independence) method are all different faces of the same methodology. While one way of interpreting this is that SVM should perform better than logistic regression, I feel that it opens the door for using logistic regression. As stated above, two advantages for using logistic regression are familarity and interpretability.

3.1.3 Boosting

Boosting is a maching learning method which attempts to "boost" the accuracy of any given learning algorithm, according to Schapire (1999). Basically, boosting takes a series of weak learners and turns them into a single strong learner. A weak learner is defined to be a classifier which is only slightly correlated with the true classification (i.e., it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

Boosting has its roots in a theoretical framework for studying machine learning called the probably approximately correct (PAC) learning model. In this framework, the learner receives samples and must select a generalization function (called the hypothesis) from a certain class of possible functions. The goal is that, with high probability (the "probably" part), the selected function will have low generalization error (the "approximately correct" part).

According to Winkler (2006), with N steps of boosting, we select a set of initial weights w_0 and successively train new weights w_i where the record pairs r that are misclassified on the prevous step are given a different weighting. The starting weight, w_0 is usually set to $\frac{1}{n}$ for each record where n is the number of record pairs in the training data. As usual with training data, the number in one class (matches) needs to be approximately equal to the number of pairs in the other class (nonmatches). Various authors have demonstrated that boosting is competitive with SVM. One research issue is determining situations where boosting substantially outperforms the Fellegi-Sunter classification rule. Another research issue, brought to attention by Winkler (2006) is whether it is possible to develop boosting methods that work with only unlabelled data or work in a semi-supervised manner as is done in record linkage.

3.1.4 Support Vector Machines

A support vector machine is a concept in computer science for a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes the input is a member of, which makes the SVM a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

The original SVM algorithm was invented by Vladimir Vapnik and the current standard incarnation (soft margin) was proposed by Cortes and Vapnik (1995).

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes and the goal is to decide which class a *new* data point is in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So, we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier, or equivalently, the perceptron of optimal stability. Below, in Figure 1, we see three different hyperplanes. The green hyperplane, represented by H3, does not separate the two different classes (represented by black and white dots). The blue hyperplane, represented by H1,

does separate the classes, but only by a small margin. The red hyperplane, represented by H2, would be the maximum-margin hyperplane.

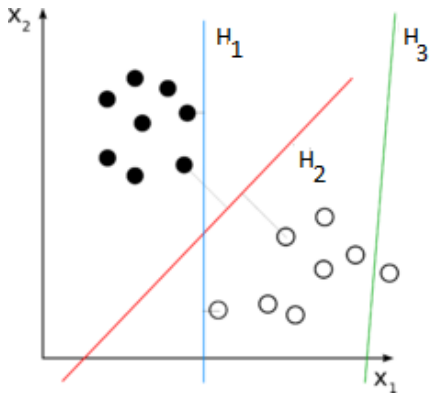

Figure 1: Looking at three different potential hyperplanes.

Ng and Jordan (2002) have demonstrated empirically and theoretically that SVM-like procedures will often outperform naive Bayes. In theory, SVM should outperform basic record linkage (possibly not by much) because the weights w are optimal for the type of linear weighting used in the decision rule. One reason that SVM may not improve much is that record linkage weights that are computed via an EM algorithm also tend to provide better separation than weights computed under a pure conditional independence assumption.

3.2 Bayesian Logistic Regression Meets Record Matching

This section presents the foundation to the Bayesian logistic regression methodology used in my research. The idea presented by Judson (2006) was to establish a decision rule from which a record pair could be declared a match or a non-match. This decision rule made itself available in logistic regression, and the Bayesian twist came about because the situation was treated as a latent model where the true matching status of the record pair is unknown.

The most relevant element of Judson's work, at least in terms of my research, is where he shows the proportional relationship between the coefficients from a logistic regression

model and the weights in the Fellegi-Sunter setup. To me, this provided the perfect link in being able to compare my modification of his method (incoporating string comparator scores in place of strictly binary predictors) to the original Fellegi-Sunter algorithm. The link is proved below, with p representing the probability of two records being declared a match given the comparator vector $\vec{x}$

Fix any $k \in \{2, ..., K\}$ where K represents the number of fields being compared.
We let the event $M = \{\text{the } i^{th} \text{ record is a match}\}$
with $U = \{\text{the } i^{th} \text{ record is a non-match}\}$
By assumption, $\log \frac{p}{1-p} = \vec{x}_i\vec{\beta}$, holds in the population, thus

$$\begin{aligned} \frac{P[M|\vec{x}\vec{\beta}]}{P[U|\vec{x}\vec{\beta}]} &= \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_K x_K) \\ &= \exp(\beta_0)\exp(\beta_1 x_1)\ldots\exp(\beta_K x_K) \end{aligned} \tag{23}$$

We may set $x_k = 1$ and $x_j = 0$ for all $j \neq k$. Then:

$$\frac{P[M|x_k]}{P[U|x_k]} = \exp(\beta_0)\exp(\beta_k x_k). \tag{24}$$

We solve for β_k and obtain:

$$\log\left[\frac{P[M|x_k]}{P[U|x_k]}\left(\frac{1}{\exp(\beta_0)}\right)\right] = \beta_k. \tag{25}$$

Utilizing Bayes' Theorem, we can rewrite the conditional probabilities in the first part of the above equation as follows:

$$\begin{aligned} P[M|x_k] &= \frac{P[x_k|M]P[M]}{P[x_k|M]P[M] + P[x_k|U]P[U]}, \\ P[U|x_k] &= \frac{P[x_k|U]P[U]}{P[x_k|M]P[M] + P[x_k|U]P[U]}. \end{aligned} \tag{26}$$

Substituting these equations back into the equation for β_k yields

$$\log \frac{\frac{P[x_k|M]P[M]}{P[x_k|M]P[M]+P[x_k|U]P[U]}}{\frac{P[x_k|U]P[U]}{P[x_k|M]P[M]+P[x_k|U]P[U]}} \left(\frac{1}{\exp(\beta_0)}\right) = \beta_k \tag{27}$$

Simplifying the fraction within the natural log gives us:

$$\log \left[\frac{P[x_k|M]}{P[x_k|U]} \left(\frac{P[M]}{P[U]}\right) \left(\frac{1}{\exp(\beta_0)}\right)\right] = \beta_k \tag{28}$$

Taking the exponential of both sides yields $e^{\beta_k} \propto \frac{P[x_k|M]}{P[x_k|U]} = \frac{m(\gamma_k)}{u(\gamma_k)}$, the Fellegi-Sunter weight.

This information allows us to apply a statistical model to the field of record matching, measure the impacts of various predictors, assess any potential interactions between said predictors, and perform model diagnostics to assess model fit. In other words, it provides structure and flexibility, two things the Fellegi-Sunter algorithm lack.

It is worthy of note here that I am not employing the exact same procedure. Markov chain Monte Carlo (MCMC) is being implemented in order to evaluate the posterior distribution and to be able to incorporate more subjective/informative prior distributions. As of the writing of this dissertation, the prior distributions are limited to simply Normal distributions with 0 mean and very large variances (essentially, non-informative). Future directions of this research would include the incorporation of subjective priors, as well as the latent model approach.

3.3 Jaro-Winkler String Comparator

In this digital age of information saturation, it is not uncommon to find excessive typographical mistakes in data entry. There have been many different attempts to understand the nature of these errors in an attempt to reconcile them. There are probabilistic models that will take into account which hand and finger are typically used to key a letter and then attempt to correct potential mistakes by looking at the keys surrounding the one that was pressed. For example, a potential mistype would be to write "Brosn" instead of "Brown". Almost any human would be able to discern what the correct word should

be, but a probabilistic model can do this as well, recognizing that "Brosn" is more likely to mean "Brown" than "Broan", "Broxn", "Brodn", or any other potential replacement letter surrounding the letter "s". The point here is that there are many ways to reconcile typographical errors.

The Jaro-Winkler (JW) similarity index is simply another attempt at incorporating some automated function to gauge string similarity. It is a variant of the Jaro metric that takes into consideration matching characters at the beginning of the strings. The higher the JW index for two strings is, the more similar the strings are. The JW metric is designed and best suited for short strings such as person names. The score is normalized such that 0 equates to no similarity and 1 is an exact match. The Jaro distance (Jaro (1989)), d_j, of two given strings s_1 and s_2 is

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \tag{29}$$

where

- m is the number of matching characters, and
- t is half the number of transpositions.

Two characters from s_1 and s_2, respectively, are considered *matching* only if they are not farther than

$$\lfloor \frac{\max(|s_1|, |s_2|)}{2} \rfloor - 1.$$

Each character of s_1 is compared with all its matching characters in s_2. The number of matching (but different sequence order) characters divided by the numeric value '2' defines the number of *transpositions*. For example, in comparing CRATE and TRACE, only 'R' 'A' 'E' are the matching characters, i.e., $m = 3$. Although 'C' and 'T' appear in both strings, they are farther than 1.5 characters apart (i.e., (5/2) - 1 = 1.5). Therefore, t=0. The $\lfloor x \rfloor$ function represents the largest integer value equal to or smaller than x.

The Jaro-Winkler distance Winkler (1990) uses a prefix scale p which gives more favorable ratings to strings that match from the beginning for a set prefix length l. Given two strings s_1 and s_2, their Jaro-Winkler distance d_w is

$$d_w = d_j + (lp(1 - d_j)) \tag{30}$$

where

- d_j is the Jaro distance for strings s_1 and s_2
- l is the length of common prefix at the start of the string up to a maximum of 4 characters
- p is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. This should not exceed 0.25. Otherwise, the distance can become larger than 1. The standard value for this constant in Winkler's work is $p = 0.1$.

Back to our example, "Dominic" vs. "Dominik"; if we look at the Jaro score for these two strings, we get a score of

$$d_j = \frac{1}{3}\left(\frac{6}{7} + \frac{6}{7} + \frac{6-0}{6}\right) = 0.9048.$$

If we boost the score using the Winkler modification since the first four letters match, we get a score of

$$d_w = 0.9048 + (4(0.1)(1 - 0.9048)) = 0.9429.$$

If we can retain this score, then we would preserve 94% of the information given in this pair of fields, instead of throwing away all that information because they do not exactly match.

Due to the way in which the likelihood is defined, incorporating these string similarity scores does not adversely impact the design of the likelihood. Obviously, we are no longer

dealing with a series of Bernoulli trials. In terms of the Fellegi-Sunter algorithm, this approach has the disadvantage that we will not be able to estimate all the m- and u-probabilities for each possible vector. A method of circumventing this issue would be to assign some threshold (say 0.85) beyond which we call the strings a match (1) and below which we say they do not match (0). Thus, we would return to the binary structure which is inherent to the Fellegi-Sunter algorithm. However, now, we have created a second point in the process by which we are using a threshold, which is essentially arbitrary, if not data-dependent. The first point being at the string comparator index step, with the second coming at the matching decision step.

4 LIKELIHOOD APPROXIMATION

Let us assume that lists A and B are independent random samples drawn from populations $\mathcal{A}$ and $\mathcal{B}$, respectively. In addition, let lists A and B consist of n and m individuals, respectively, with $n \leq m$. We will condition on the longer list, i.e., the one from population $\mathcal{B}$. The information recorded for the ith individual on list A is $\{(Y_{ij}, \gamma_{ij}) : 1 \leq j \leq m\}$, where $Y_{ij} = 1$ if individual $i \in A$ matches with $j \in B$ and 0 otherwise, and γ_{ij} representes the vector of comparators for this particular pair (i, j). It should be noted that this binary structure is the most common comparator structure with regards to record matching. One of the modifications presented in my research is to extend the parameter space from binary $\{0, 1\}$ to continuous on the [0,1] scale by way of the Jaro-Winkler string comparators discussed in the previous Section. For convenience, let us use the following notation: $(\mathbf{Y}_i, \mathbf{\Gamma}_i)$, where $\mathbf{Y}_i = (Y_{i1}, \ldots, Y_{im})^T$ and $\mathbf{\Gamma}_i$ is the $m \times p$ matrix with jth row $\gamma_{ij}, j = 1, ..., m$.

If the sample size n is small in comparison to the size N of population $\mathcal{A}$, then the sample from $\mathcal{A}$ constitutes an approximate multinomial experiment. The number of categories is $m + 1$ times the number, C, of different values that can be taken on by the comparator matrix $\mathbf{\Gamma}$. The category probabilities can be written as follows:

$$\pi(r, s) = P(Y_{i1} = 0, \ldots, Y_{ir} = 1, Y_{i(r+1)} = 0, \ldots, Y_{im} = 0, \mathbf{\Gamma}_i = \mathbf{C}_s), \tag{31}$$

$$r = 1, \ldots, m, s = 1, ..., C,$$

and

$$\pi(m + 1, s) = P(Y_{i1} = 0, \ldots, Y_{im} = 0, \mathbf{\Gamma_i} = \mathbf{C}_s),\ s = 1, ..., C \tag{32}$$

where $\mathbf{C}_1, \ldots, \mathbf{C}_C$ denote the possible values taken by the comparator matrix.

In (33), we are saying that record $i \in A$ matches with $r \in B$. In (34), we are saying that record $i \in A$ does not match with any records in B. As you can see, we are making the assumption that each individual in sample A matches with AT MOST one observation in

sample B. This write-up does not take into account when a particular observation matches with more than one individual. There is a place for either approach in the field of record matching.

Since the event $Y_{i1} = 0, \ldots, Y_{ir} = 1, Y_{i(r+1)} = 0, \ldots, Y_{im} = 0$ is equivalent to the event $Y_{ir} = 1$, we have

$$\pi(r, s) = P(Y_{ir} = 1 | \mathbf{\Gamma}_i = \mathbf{C}_s) f_{\mathbf{\Gamma}}(\mathbf{C}_s),\ r = 1, ...m \tag{33}$$

where $f_{\mathbf{\Gamma}}$ is the pmf of $\mathbf{\Gamma}$ in the strictly binary $\{0, 1\}$ case or the pdf of $\mathbf{\Gamma}$ in the Jaro-Winkler string comparator score case. Let us assume that $P(Y_{ir} = 1 | \mathbf{\Gamma}_i = \mathbf{C}_s)$ is the same for $r = 1, \ldots, m$. This conditional probability is the quantity of interest in this research.

As a practical matter, it is undoubtedly necessary to parameterize $P(Y_{ir} = 1 | \mathbf{\Gamma}_i = \mathbf{C}_s)$. A simplifying assumption is the following:

$$P(Y_{ir} = 1 | \mathbf{\Gamma}_i) = P(Y_{ir} = 1 | \boldsymbol{\gamma}_{ir}) \tag{34}$$

This states that if we have the comparator vector for the pair of names (i, r), then knowing the comparator vectors of i and each of the other names on list B is not necessary for purposes of predicting Y_{ir}. Also, we assume that

$$P(Y_{ir} = 1 | \boldsymbol{\gamma}_{ir}) = f(\boldsymbol{\gamma}_{ir} | \boldsymbol{\theta}) \text{ for all } r \tag{35}$$

for some vector-valued parameter $\boldsymbol{\theta}$. Inferring $\boldsymbol{\theta}$ is now the goal of the investigation.

The likelihood has the form

$$L(\boldsymbol{\theta}) = \prod_{i \in S_{match}} f(\boldsymbol{\gamma}_{ir_i} | \boldsymbol{\theta}) \prod_{i \in S^c_{match}} \left[1 - \sum_{r=1}^{m} f(\boldsymbol{\gamma}_{ir} | \boldsymbol{\theta}) \right] \prod_{i=1}^{n} f_{\mathbf{\Gamma}}(\mathbf{\Gamma}_i) \tag{36}$$

where S_{match} is the set of indices i such that $Y_{ir_i} = 1$ for $1 \leq r_i \leq m$.

For the purposes of MCMC, the quantity $\prod_{i=1}^{n} f_{\mathbf{\Gamma}}(\mathbf{\Gamma}_i)$ is just a constant of proportionality. Thus, the posterior for $\boldsymbol{\theta}$ is completely free of this quantity.

The form of the parametric probability function, $f(\boldsymbol{\gamma}|\boldsymbol{\theta})$, chosen for this research is of logistic form:

$$\log\left[\frac{f(\boldsymbol{\gamma}|\boldsymbol{\theta})}{1-f(\boldsymbol{\gamma}|\boldsymbol{\theta})}\right] = \theta_0 + \sum_{j=1}^{p}\theta_j\gamma_j \tag{37}$$

where γ_j is the jth component of $\boldsymbol{\gamma}, j = 1, \ldots, p$.

Ultimately, we wish to devise a method for predicting matches on a new set of data $(\widetilde{\mathbf{Y}}_i, \widetilde{\boldsymbol{\Gamma}}_i), i = 1, \ldots, l$. In this situation, only $\widetilde{\boldsymbol{\Gamma}}_1, \ldots, \widetilde{\boldsymbol{\Gamma}}_l$ are known. Suppose we assume that $(\widetilde{\mathbf{Y}}_i, \widetilde{\boldsymbol{\Gamma}}_i), i = 1, \ldots, l$, follow the same parametric model as do $(\mathbf{Y}_i, \boldsymbol{\Gamma}_i), i = 1, \ldots, n$. In addition, we assume that, given $\boldsymbol{\theta}$, the two datasets are independent of each other. Letting $\mathbf{D}$ represent the original set of data, the posterior predictive distribution would then be calculated in the following manner:

$$m(\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_l|\widetilde{\boldsymbol{\Gamma}}_1, \ldots \widetilde{\boldsymbol{\Gamma}}_l, \mathbf{D}) = \frac{\int_\Theta L(\boldsymbol{\theta})L_{new}(\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}{\int_\Theta L(\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{38}$$

where $L_{new}(\boldsymbol{\theta})$ is the likelihood of the new data set $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_l$ and π is the prior for (Θ).

5 DETAILS OF A VIRTUAL SIMULATION

While obtaining data for which records can be matched is a simple task unto itself, obtaining a pair of datasets where all possible combinations have known matching status is considerably more difficult. There is a considerable amount of effort that goes into discerning a large number of matched or non-matched pairs. As a result of this limitation, this dissertation only presents simulated data, which were simulated using a software package called LINKSOLV from a company called Strategic Matching, led by Michael H. McGlincy. The purpose of this type of simulation is to output records with information that would be expected on accident and ambulance forms. In addition, unique identifier numbers attached to each observation allow 100% knowledge of matching status, which is crucial when training a model. The specific data that are being matched are crash reports (CRASH) and ambulance run reports from Emergency Medical Services (EMS). As McGlincy (2006) writes, many times, it can be difficult to match individuals from crashes to ambulance run reports because each particular organization records different information and they are not necessarily consistent on which information is obtained, or even if that information is accurate. Simulating record linkage data can give individuals and corporations the opportunity to build training models on data in which one already knows the matching status of a particular pair (unique identifiers are outputted with the simulated data for matching purposes).

There are 22 different variables simulated in the record linkage software that are common to both the CRASH and EMS datasets. Those variables, their descriptions, and the number of possible values are outlined in Table 3. The description is important because it provides us with a bit of context in terms of the data and why particular predictors are significant to the prediction of matching status. The number of possible values is important because it outlines an interesting paradox in record matching. In terms of how many possible values there are, for some of the variables, the number is fixed, like with Sex or KABCO). On the other hand, other variables will be data-dependent (indicated as such by

"look at data"), like the County in which the incident occurred, or the number of different Soundex codes.

In general, a predictor with too few possible values (like Gender) will do very poorly in terms of record matching. Intuitively, this is not surprising, for often two people are the same gender, but are not the same individual. Mathematically, we can articulate this argument in the following fashion: In calculating weights for the Fellegi-Sunter method, see Equation (7), we are taking the ratio of $m(\gamma)$ to $u(\gamma)$ and summing the logarithms of those ratios. Since $m(\gamma)$ should be greater than $u(\gamma)$ for agreement configurations (otherwise, the data is too messy to be useful to any algorithm), the ratio should be greater than 1 and the logarithm of this ratio should be greater than 0. In the case of $u(\gamma_{Gender})$, we are dealing with a value that will be close to 0.5. This means that if $m(\gamma_{Gender})$ is close to 0.5, then the ratio will be close to 1 and the logarithm will be close to 0. As such, Gender would contribute very little to the Fellegi-Sunter algorithm, in much the same way that it contributes little to the logistic regression setting.

The flip side of this paradox is considerably more troubling. If a variable has too many possible values (i.e., bordering on the unique) then that does not bode well either for record linkage purposes. The reason here is twofold: (1) the sparseness of the data for each unique value makes coming up with maximum likelihood estimates difficult and (2) the uniqueness of the data may drown out other variables in terms of predictive power. Regarding the first point, if we had a seemingly unique variable, then the comparator score for that unique variable (e.g. SSN) would almost line up perfectly with the matching status. For example, if the two SSN's match, then the probability of the records matching would likely be 1. For two SSN's that did not match, the probability of those records matching would likely be 0. As such, we would have perfect or near perfect discrimination; Agresti (2007)

For the CauseOfInjury variable, the ICD-9 E code refers to the International Classification of Diseases and is designed to promote international comparability in the collection, processing, classification, and presentation of mortality statistics. With respect to First-Soundex and LastSoundex, Soundex codes are output via a phonetic algorithm designed

to encode homophones in the same representation so that they can be linked regardless of spelling. For example, Dominic and Dominik, while spelled differently, would have the same Soundex code of D552. Soundex codes, and their counterpart, the New York State Identification and Intelligence System (NYSIIS) are used very frequently in record matching as this helps to eliminate inconsistencies in data entry. FIPS county codes are five digit Federal Information Processing Standard codes which uniquely identify counties. The first two digits actually correspond to the FIPS state code while the last three encompass the county code.

There are two parameters under the user's control in simulating this data: Error Probability and Missing Probability. As anyone can guess, data collection can be an erroneous affair, assuming that data is even collected. This phenomenon is particularly prevalent in record matching. Many times there will be variables that have missing values. Reasons for this include there not being enough time to collect the data, an individual choosing not to answer a question, surveys getting lost in the mail, and so on. If data is obtained, it may not be accurate. If data is hand-written, the person keying in the data may either make a mistake in reading the responses, or may make a typographical (a.k.a "fat-finger") mistake. Regardless of how data is collected and/or keyed in, typographical mistakes are ubiquitous. Even worse, data can be falsified or fraudulent for any number of possible reasons. As such, attempting to build training models on perfect data will not generalize well to any real datasets. The missing and error probabilities for each of the 22 variables is listed in Table 4. There were two different datasets involved in the construction of this final model: DS1 represents the better looking data in terms of missing and erroneous data. The model here, however, did quite well, and the quality of the data was a considerable contributor to that success. As such, we upped the ante by increasing the probability of missing/erroneous for DS2, the messier data. Ultimately, the process flow treats DS1 as the training dataset, building the model, and then scoring the model on DS2.

The record matching simulation software does have some issues of its own that should be pointed out and taken into account when considering modeling results. For example,

Table 3: Description of common variables simulated in record matching software.

Variable	Description	# of possible values
Age	Age in years	look at data
BirthDate	Date of birth of occupant (m/d/yyyy)	look at data
CauseOfInjury	Crash Information coded as ICD-9 E code (E810.0)	look at data
Collide	Type of crash coded like Std E Code (NO, OBJ, OTH, PED, TRN, VEH)	6
County	Location of crash as FIPS county code	look at data
Date	Date of crash (m/d/yyyy)	look at data
Time	Time of crash (hh:mm)	look at data
CrashZip	Location of crash as USPS zipcode	look at data
FirstInitial	Initial of occupant's first name	26
FirstSoundex	Soundex code of occupant's first name	876096
HomeZip	Location of residence as USPS zipcode	look at data
KABCO	Injury Severity (K=Killed, A=Incapacitating Injury B=Non-incapacitating Injury, C=Possible Injury, O=No Injury	5
LastInitial	Initial of occupant's last name	26
LastSoundex	Soundex code of occupant's last name	876096
PartInj	Injured body part (NO=None, HD=Head, CH=Chest, BK=Back, AR=Arms, LG=Legs)	6
PlateNbr	License Plate Number (A123456)	look at data
Race	Race (LT=Light, DK=Dark)	2
SSN	Social Security Number (123456789)	look at data
Safety	Safety equipment used (N=No, Y=Yes, X=N/A)	3
Seat	Seating position coded like Std E Code (DRV=Driver, PAS=Passenger)	2
Sex	Sex of occupant (F=Female, M=Male)	2
Vehicle	Type of vehicle coded like Std E Code (MC=Motorcycle, MV=Motor Vehicle)	2

the error generation algorithm (controlled by the error probability given above) randomly substitutes a different value for the correct value in each character location. The rationale behind this is that in health care records, the birthdate or social security number of a spouse or other family member may be substituted in place of the patient's. In situations such as this, it would not be surprising to see situations where the name of the driver on the police crash report would be completely different from a cousin's information at the hospital. This makes the prospect of record matching difficult in these, ideally rare, scenarios. Also, if

Table 4: Probabilities of Error/Missing of simulated variables.

Variable	P(Miss) - DS1	P(Error) - DS1	P(Miss) - DS2	P(Error) - DS2
Age	0.05	0.02	0.10	0.04
BirthDate	0.05	0.02	0.10	0.04
CauseOfInjury	0.05	0.002	0.10	0.004
Collide	0.05	0.002	0.10	0.004
CrashCounty	0.05	0.002	0.10	0.004
CrashDate	0.05	0.002	0.10	0.004
CrashTime	0.05	0.02	0.10	0.04
CrashZip	0.05	0.002	0.10	0.004
FirstInitial	0.05	0.02	0.10	0.04
FirstSoundex	0.05	0.02	0.10	0.04
HomeZip	0.05	0.02	0.10	0.04
KABCO	0.05	0.05	0.10	0.10
LastInitial	0.05	0.02	0.10	0.04
LastSoundex	0.05	0.02	0.10	0.04
PartInj	0.05	0.05	0.10	0.10
PlateNbr (A123456)	0.05	0.02	0.10	0.04
Race	0.05	0.02	0.10	0.04
SSN	0.05	0.02	0.10	0.04
Safety	0.05	0.02	0.10	0.04
Seat	0.05	0.002	0.10	0.004
Sex	0.05	0.002	0.10	0.004
Vehicle	0.05	0.002	0.10	0.004

data are time-dependent, addresses and last names can easily be modified, making adequate record matching even more complicated.

6 IMPLEMENTATION

The Implementation Section itself will consist of two parts: (1) a collaboration between the Fellegi-Sunter (FS) algorithm with a cost-based algorithm designed by Verykios et al. (2003) for declaring matches in the case where binary values are the inputs and (2) the modification of Judson's Bayesian logistic regression (BLR) taking into account Jaro-Winkler string comparator scores.

6.1 The Champion: FS

The baseline algorithm is a mix of two different algorithms. The first one, which comes straight from Fellegi and Sunter (1969), is what I used to estimate the conditional probabilities. From there, these probabilities are fed into the second algorithm, from Verykios et al. (2003), along with some cost matrix elements, to calculate and aggregate the weights. Finally, matching decisions are made based on how these composite scores compare to particular thresholds.

For the FS approach, four things are required:

1. Cost matrix elements (user-provided)
2. Conditional probabilities of comparison vector given matching status (estimated by FS)
3. Overall probabilities of matching (estimated by FS)
4. Data vectors (data-provided)

The only foreign element here is the cost matrix. The cost matrix elements used in Verykios' algorithm are meant to generalize the idea of minimizing error probability. In certain business applications, it may be more pertinent to minimize false matches over false non-matches, or vice versa. In the context of minimizing errors, false matches and false non-matches are treated equally.

The setup for, and equation used in calcuating, the conditional probabilities $m(\gamma)$ and $u(\gamma)$ are presented in the second section of the Appendix. Given those conditional probabilities, we input them into the likelihood ratio statistic along with the data vectors. Here, the likelihood scores will be calculated and aggregated to yield the composite scores utilized in the decision-making process.

Using (7), (12) and (13),

$$\log\frac{u(\gamma)}{m(\gamma)} = \log\frac{u_1(\gamma^1)\cdot u_2(\gamma^2)\cdots u_K(\gamma^K)}{m_1(\gamma^1)\cdot m_2(\gamma^2)\cdot m_K(\gamma^K)}, \tag{39}$$

which can also be written as follows:

$$\log\frac{u(\gamma)}{m(\gamma)} = \log\frac{u_1(\gamma^1)}{m_1(\gamma^1)} + \log\frac{u_2(\gamma^2)}{m_2(\gamma^2)} + \cdots + \log\frac{u_K(\gamma^K)}{m_K(\gamma^K)} = \sum_{i=1}^{K}\log\frac{u_i(\gamma^i)}{m_i(\gamma^i)}. \tag{40}$$

Now, because the comparison vector components are binary in nature (and subsequently, the conditional distributions are both Bernoulli trials, the log-likelihood-ratio statistic is expressed in the following manner:

$\gamma_i|M \sim$Bernoulli(p_i)

$m_i(\gamma_i = 1) = P(\gamma_i = 1|M) = p_i$

$m_i(\gamma_i = 0) = P(\gamma_i = 0|M) = 1 - p_i$

$\gamma_i|U \sim$Bernoulli(p_i)

$u_i(\gamma_i = 1) = P(\gamma_i = 1|U) = q_i$

$u_i(\gamma_i = 0) = P(\gamma_i = 0|U) = 1 - q_i$

$$\log\left(\frac{u(\gamma_i)}{m(\gamma_i)}\right) = \log\left(\frac{q_i^{\gamma_i}(1-q_i)^{1-\gamma_i}}{p_i^{\gamma_i}(1-p_i)^{1-\gamma_i}}\right) \tag{41}$$

$$= \log\left(q_i^{\gamma_i}(1-q_i)^{1-\gamma_i}\right) - \log\left(p_i^{\gamma_i}(1-p_i)^{1-\gamma_i}\right) \tag{42}$$

$$= [\gamma_i \log q_i + (1-\gamma_i)\log(1-q_i)] - [\gamma_i \log p_i + (1-\gamma_i)\log(1-p_i)] \tag{43}$$

$$= \gamma_i \log\frac{q_i}{p_i} - \gamma_i \log\frac{1-p_i}{1-q_i} + \log\frac{1-q_i}{1-p_i} \tag{44}$$

$$= \gamma_i \left[\log\frac{q_i}{p_i} + \log\frac{1-p_i}{1-q_i}\right] + \log\frac{1-q_i}{1-p_i} \tag{45}$$

$$= \boxed{\gamma_i \left[\log\frac{q_i(1-p_i)}{p_i(1-q_i)}\right] + \log\frac{1-q_i}{1-p_i}} \tag{46}$$

Based on (48), (42) can be rewritten as follows:

$$\log\frac{u(\gamma)}{m(\gamma)} = \sum_{i=1}^{K} \gamma_i \left[\log\frac{q_i(1-p_i)}{p_i(1-q_i)}\right] + \sum_{i=1}^{K} \log\frac{1-q_i}{1-p_i}. \tag{47}$$

Now, given this summation, we can calculate the likelihood scores from the data vectors and the estimated conditional probabilities (p_i and q_i) from the Fellegi-Sunter algorithm. Given these likelihood scores, we compare those to a couple of thresholds. These thresholds are dependent on the cost matrix elements as well as the overall probability of matching. The cost matrix looks like the following:

$$\begin{pmatrix} c_{A_1M} & c_{A_1U} \\ c_{A_2M} & c_{A_2U} \\ c_{A_3M} & c_{A_3U} \end{pmatrix}.$$

These thresholds are meant to minimize the overall cost of the record matching associated with three different results

1. Type I Errors (false matches) (c_{A_1U} - first row, second column of cost matrix)
2. Type II Errors (false non-matches) (c_{A_3M} - third row, first column of cost matrix)
3. Additional Review (c_{A_2M} and c_{A_2U} - second row of cost matrix)

The thresholds from Verykios' model are calculated via the cost matrix elements and the a priori probabilities of matching:

$$\lambda = \frac{\pi_0}{1-\pi_0} \cdot \frac{c_{A_2M} - c_{A_1M}}{c_{A_1U} - c_{A_2U}} \tag{48}$$

$$\mu = \frac{\pi_0}{1-\pi_0} \cdot \frac{c_{A_3M} - c_{A_2M}}{c_{A_2U} - c_{A_3U}} \tag{49}$$

Using the log-likelihood score (the composite score) and these thresholds, the following decisions are made:

$$\text{Decision} = \begin{cases} \text{Match} & \log\frac{u(\gamma)}{m(\gamma)} < \lambda \\ \text{Possible Match} & \lambda < \log\frac{u(\gamma)}{m(\gamma)} < \mu \\ \text{No Match} & \log\frac{u(\gamma)}{m(\gamma)} > \mu \end{cases} \tag{50}$$

Typically, the cost matrix elements would be provided by the client or customer.

6.2 The Challenger: BLR

The Bayesian Logistic Regression setup is implemented via the MCMC procedure in the SAS software. Markov chain Monte Carlo (MCMC) is a simulation method which allows one to sample from non-standard probability distributions utilizing a Markov chain. One of the desired properties of MCMC is that as the number of simulations increases, the subsequent distribution converges to the stationary distribution. In Bayesian statistics, we are often interested in obtaining posterior and posterior predictive distributions. The posterior is proportional to the product of a prior distribution and a likelihood distribution. Specific to the posterior predictive distribution, integration is key in calculating probabilities, which is precisely where MCMC's true power can be actualized. At the same time, however, I utilize the posterior distribution in my research when I perform my cross-validation scheme. The MCMC procedure in SAS uses a random-walk Metropolis algorithm to obtain posterior samples.

It bears noting that, after viewing initial outputs, modifications were made to the MCMC methodology itself. For example, the number of burn-in values, initially defaulted at 1000, has been boosted to 7500. The reason for this is that the Geweke diagnostics were significant for three of the parameters, ultimately indicating that convergence had not been reached and that more initial observations needed to be discarded. Also, the

level of thinning increased quite substantially from 5 to 250. This occurred because the autocorrelation diagnostics indicated a significant correlation between iterations. Thinning is a method commonly used in MCMC as a means of systematically selecting a sample from the MCMC iterations, as a means of reducing autocorrelation since the samples generated by MCMC have high positive autocorrelation. Because of the significant increase in terms of thinning and the additional burn-in values, we generated a total of 250,000 observations, yielding a total of 970 observations used to approximate the posterior distribution.

Wrapped around this MCMC procedure is a leave-one-out cross-validation approach used for model selection. There are 22 variables in the initial model, but not all of these are significant in predicting matching status. The design of the cross-validation is as follows:

1. Create all possible combinations of the pairs
2. Remove first pair
3. Run PROC MCMC on remaining pairs to obtain posterior distribution
4. Given posterior distribution and covariates, calculate matching probability (p)
5. If $p > 0.50$, assign pair as a match
6. Repeat steps 2 through 5 for all possible combinations
7. Given true matching status, calculate Recall, Precision, F-Score

Of course, the threshold for deciding a match is arbitrary and, as will be discussed in the coming Section, can be tuned to suit specific needs.

Beyond the simulation of the datasets, there were additional steps taken with regard to the data in an attempt to improve the performance of the Bayesian logistic regression model. Those modifications are listed below:

- Trailing 0's removed from SOUNDEX variables

- BirthDate partitioned into Month, Day and Year

  - Removed '19' from year

- Threshold changed from 0.5 to 0.15

The first change likely requires a bit of explanation as to how the SOUNDEX variable works. SOUNDEX is a phonetic algorithm which basically breaks words down into how they sound. The idea behind this algorithm, and the similar NYSIIS (New York State Identification and Intelligence System), is to group together words that sound the same, in an attempt to alleviate spelling errors. The rules, applied to my name, "Dominic," are as follows:

1. Retain the first letter of the name and drop all other occurrences of a, e, i, o, u, y, h, w

   Dominic $\rightarrow$ Dmnc

2. Replace consonants with digits as follows (after the first letter)

   b, f, p, v $\rightarrow$ 1

   c, g, j, k, q, s, x, z $\rightarrow$ 2

   d, t $\rightarrow$ 3

   l $\rightarrow$ 4

   m, n $\rightarrow$ 5

   r $\rightarrow$ 6

   Dmnc $\rightarrow$ D552

3. Two adjacent letters (in the original name) with the same number are coded as a single number, as are two letters with the same number separated by either 'h' or 'w'. However, such letters separated by a vowel are coded twice.

4. Continue until you have one letter and three numbers. If you run out of letters, simply pad the ending with 0's.

In terms of the Jaro-Winkler string comparator score, that last rule is paramount. A lot of the records had trailing zeroes, which artifically inflates the comparator score, leading to false matches.

The second change occurred because the simulation outputs birthdates in the following format: mm/dd/19yy. As all birthdates are listed in this fashion, the string comparator will see the "/" and the "19" and raise the score. Since the slashes and the first two numerals in the year provide no distinctive power, these were removed them from the dataset as well. As we progress further into the 21st century, the first two digits of the year will become more important. However, with only 2 possible values (19 and 20), we encounter a similar problem as with Gender and this particular component will likely not be significant.

The final change occurred after the procedure had already run and I had the opportunity to see what the distribution of probabilities looked like. In addition, I can run the algorithm at different thresholds of matching in an effort to optimize the algorithm in terms of Recall and Precision (vis-a-vis the F-Score). While it may make intuitive sense to place this probability at 0.5, that is not usually the most optimal in terms of maximizing F-Score. The Results Section will expound more on the ultimate choice of the threshold along with some ideas for why 0.5 was not optimal.

As you will notice, there are some predictors that have very few values, indicating that they are unlikely to provide a lot of predictive power to the overall matching scheme. Some of these variables actually do have something to provide, while others with more possible values (and seemingly higher differentiability) are not as significant. The rationale behind this is that while not having very many fields is bad (Dominic Jann and Bob Jones are both males, but obviously not the same person), the same is actually true for having too many fields, because then you do not have enough data per predictor value to provide a good model.

As you can already tell, in terms of constructing a logistic regression model, there are

going to be some redundancies given the aforementioned list of variables. For this reason, those variables providing less specific information were removed. For example, Age and BirthDate essentially give the same information, but whereas age is likely restricted to between 16-100 (only 85 possible values), BirthDate can assume a much larger number of values (365*85=31025). The total number of possible values is important in terms of predictive value. Obviously, a variable with a larger amount of variation would be better suited for determining matching pairs than a variable with only 2 levels, such as gender.

The prior distributions used in the MCMC were strictly non-informative. Each coefficient had a $\mathcal{N}(0, 100^2)$ prior distribution. As the data structure was not well-understood, it was thought that beginning with non-informative priors would allow the data to speak for itself. It would be most advantageous to see how tweaking these prior distributions would impact the final model and its ability to discern matching status.

7 RESULTS

7.1 Variable Selection

In Section 5, the variables used in the modeling were described in more detail. After running leave-one-out cross-validation on the first set of simulated data, the following variables were found to be the most significant in adequately predicting matching probability:

- BirthDate
- Collide
- CrashZip
- FirstSoundex
- LastInitial
- PartInj
- Safety

An obvious question after seeing this list would be "Why these particular variables?" If you notice in the list, refer to Section 5 for a refresher on what the variables are, you will notice that there is quite a mix of variable types. For example, variables such as Birthdate have a much broader range of values. It is unlikely that multiple people are going to share the same birthday. Thus, if two records match on BirthDate, it would stand to reason that the probability of matching for those particular records is going to increase. If they do not match on Birthdate, a variable that will not change with time or relocation, then, odds are, they do not match. What is curious, however, is that Social Security Number (SSN) was not included in the final list. Part of the reason for this is that Social Security Number does not necessarily uniquely identify individuals. There is evidence that, after the Social Security Administration was formed in 1935, many people believed new employment

signified a new social security number. In addition, employers used to submit applications on behalf of former employers so that earnings could be deposited in a particular account.

It is worthy of note that the only real comparisons made between the FS algorithm and the Bayesian Logistic Regression were based on only three predictors. The reason for this is as follows: in Fellegi and Sunter (1969), the method has an unmistakable caveat: "if, in fact, there are more than three components (fields, attributes, etc.), they can be grouped until there are only three left." While there is no denying that these groupings can occur, the obvious question becomes, "How do we group them?" I have found that the success of the algorithm depended in large part on which variables where used and how they were grouped. This makes a direct, one-to-one comparison between my modified BLR algorithm and the FS algorithm quite difficult for anything beyond 3 predictors.

What are we most concerned about in terms of results with respect to record matching? Well, like any classification scheme, there are several measures we are interested in:

- $\text{Recall} = \dfrac{\#\text{ correctly assigned matches}}{\#\text{ total true matches}}$
- $\text{Precision} = \dfrac{\#\text{ correctly assigned matches}}{\#\text{ total assigned matches}}$
- $\text{F1 Score} = 2 \cdot \dfrac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Recall is a measure of how many true matches the algorithm obtains, whereas Precision is a measure of how many correct matches the algorithm assigns. In general, as one of these last two measures increases, the other decreases, making it difficult to completely assess a classification scheme in terms of one or the other. For example, if we increase the threshold by which we define a match, then our Precision is likely to increase, owing to the fact that it takes more to be assigned as a match. By the same token, because it takes more to be a match, fewer pairs will be assigned as matches, thereby decreasing the Recall. This is where the F1 Score helps us. It considers both the Precision and the Recall in terms of a weighted average.

Another element that is of interest is that of computation time. In this era of supercomputers and megabyte download speeds, many times companies are just as interested, if not more so, in the efficiency of a particular algorithm or process than the accuracy of it. Some may be willing to sacrifice a percentage point or two in terms of an accuracy statistic to gain hours or days on an algorithm's run time. Unfortunately, my algorithm does not improve on the baseline Fellegi-Sunter algorithm in terms of computation time. Due to the MCMC nature of my algorithm, the many simulations that are performed in creating posterior and posterior predictive distributions adds considerably to the run time. However, it should be noted that my algorithm is not meant to be run on entire datasets or entire combinations of all possible pairs. While the Fellegi-Sunter algorithm, and many others, do focus considerably on computation speed as well as accuracy, still many other algorithms focus more of their attention on the accuracy. The reason for this is because of blocking. Blocking is discussed more in the introduction as its relevance is more pertinent there and I am not incorporating a blocking scheme here. So, while my algorithm may not perform as well as the baseline Fellegi-Sunter in terms of computation speed, the elevated accuracy combined with a good blocking scheme will yield better results in terms of adequate matching.

7.2 Comparing FS to BLR

There are two ways in which these can be run. One can use the more traditional binary (0/1) structure in terms of how they define γ or one can use something of a more continuous nature (e.g., string comparator scores). For the Fellegi-Sunter algorithm, in order to calculate the conditional probabilities $m(\gamma)$ and $u(\gamma)$, the comparator components must be of a binary nature. This does not necessarily mean one can not use the string comparators in this case, but it does mean that a threshold will have to be defined in terms of what constitutes a "match" and what constitutes a "non-match" in terms of γ. This is not to be confused with the threshold defined at the end of any record-matching algorithm in which one must decide where to draw the line between the matching population (M) and the non-

Table 5: Results on first set of simulated data comparing FS procedure with BLR.

Method	Comp Time (s)	FPR	Recall	Precision	F1 Score
FS	6	0.1003	0.9705	0.8997	0.9338
BLR	10920	0.0068	0.9607	0.9932	0.9767

matching population (U). This is one disturbing aspect of the Fellegi-Sunter algorithm that is not seen in the generalized linear model method I am performing. The only threshold component in what I am doing is the same threshold prevalent in any record-matching: beyond what point does a final score/probability constitute a matching pair?

There are two different elements to the results worth highlighting. The first is the more relevant, in which I compare my PROC MCMC procedure with the Fellegi-Sunter algorithm on a select subset of data. Table 5 presents this information for the 93025 comparisons that were assigned in my initial analysis.

What does this table tell us? Well, first and foremost, the FS procedure is not a bad procedure. If it were, it would not have gained the acclaim and application it has over the course of the past 4+ decades. However, we do see that BLR performs better in terms of Precision, meaning that when the algorithm selects a match, it is more likely to be a correct match than the FS procedure. Because the Recall numbers are so close, the subsequent F1 score also indicates that the MCMC procedure is a slightly better approach. A somewhat disturbing trend is that the FS procedure accuracy tends to deterioriate as the number of comparisons went up. For example, Table 6 outlines what happens to FS as the number of comparisons increases.

Table 6: How FS deterioriates as # of comparsisons increases.

# of comps	Comp Time (s)	FPR	Recall	Precision	F1 Score
93025	6	0.1003	0.9705	0.8997	0.9338
2917020	1260	0.2429	0.9606	0.7571	0.8468
11022732	7200	0.3180	0.9561	0.6820	0.7961

As we can see, while the Recall stays virtually the same across the 3 sizes, the Precision

(and, subsequently, the F1 Score) steadily declines. A consistent Recall and a decreasing Precision indicate that the same matches are being classified as matches, but that there is an increasing number of incorrectly matched pairs. As you may notice in Table 5, it took about 10,920 seconds (or slightly over 3 hours) to run the MCMC procedure on a seemingly small set of data. Again, I must reiterate that this procedure is meant to be run on data that is already blocked. Having said that, while Table 6 provides evidence that FS deteriorates as sample sizes increase, I do not have such a table for MCMC. As we can see in Table 5, it took MCMC 1820 times longer to analyze the same data. If we simply extrapolated, which would actually be conservative in this case, the MCMC procedure would take 1820*1260=2,293,200 seconds (or approximately 27 days).

7.3 Final Model

The final Bayesian Logistic Regression model is the following:

$$ln\left(\frac{p}{1-p}\right) = -32.6608 + 22.6693\gamma_{BirthDate} + 2.0365\gamma_{Collide} + 12.8769\gamma_{CrashZip}$$

$$+14.5329\gamma_{FirstSoundex} + 5.4063\gamma_{LastInitial} + 2.0610\gamma_{PartInj} + 1.4925\gamma_{Safety}$$

$$-7.9291\gamma_{BirthDate}\gamma_{CrashZip} - 12.2675\gamma_{BirthDate}\gamma_{FirstSoundex} - 2.3669\gamma_{CrashZip}\gamma_{LastInitial}$$

SAS outputs certain Bayesian model diagnostics by default and those are included below:

Posterior Summaries						
Parameter	N	Mean	Standard Deviation	Percentiles		
				25%	50%	75%
beta0	1000	-32.6608	1.2505	-33.4229	-32.6437	-31.7883
beta1	1000	22.6693	1.2988	21.7816	22.6331	23.5007
beta2	1000	2.0365	0.2001	1.8990	2.0358	2.1727
beta4	1000	12.8769	1.2185	12.0576	12.8325	13.6396
beta5	1000	14.5329	0.9057	13.8999	14.5597	15.1323
beta6	1000	5.4063	0.7095	4.9228	5.3845	5.8484
beta7	1000	2.0610	0.1723	1.9422	2.0611	2.1795
beta8	1000	1.4925	0.1857	1.3693	1.4957	1.6189
beta11	1000	-7.9291	1.2278	-8.6983	-7.9251	-7.1012
beta12	1000	-12.2675	1.0313	-12.9475	-12.2806	-11.6073
beta28	1000	-2.3669	0.8661	-2.9288	-2.3581	-1.7670

Figure 2: Coefficients for Bayesian Logistic Regression model.

Monte Carlo Standard Errors			
Parameter	MCSE	Standard Deviation	MCSE/SD
beta0	0.0395	1.2505	0.0316
beta1	0.0387	1.2988	0.0298
beta2	0.00633	0.2001	0.0316
beta4	0.0383	1.2185	0.0314
beta5	0.0270	0.9057	0.0298
beta6	0.0230	0.7095	0.0324
beta7	0.00580	0.1723	0.0337
beta8	0.00587	0.1857	0.0316
beta11	0.0351	1.2278	0.0286
beta12	0.0324	1.0313	0.0314
beta28	0.0278	0.8661	0.0321

Figure 3: Monte Carlo Standard Error and Standard Deviation.

Posterior Autocorrelations				
Parameter	Lag 1	Lag 5	Lag 10	Lag 50
beta0	0.0825	0.0257	0.0174	0.0269
beta1	0.1022	0.0158	-0.0167	-0.0064
beta2	0.1485	0.0263	0.0334	0.0205
beta4	0.2144	0.0097	-0.0424	0.0609
beta5	0.1163	0.0608	0.0089	-0.0170
beta6	0.1432	-0.0015	-0.0115	-0.0134
beta7	0.0632	0.0462	-0.0341	0.0300
beta8	0.1389	0.0621	0.0148	0.0291
beta11	0.1077	-0.0355	-0.0346	0.0287
beta12	0.1612	0.0712	0.0131	-0.0291
beta28	0.1301	-0.0103	-0.0239	-0.0048

Figure 4: Autocorrelations at Lags 1, 5, 10, 50 for data thinned by every 250th iteration.

Geweke Diagnostics		
Parameter	z	Pr > |z|
beta0	-1.3386	0.1807
beta1	1.9769	0.0481
beta2	-0.8323	0.4052
beta4	2.4534	0.0142
beta5	-0.1755	0.8607
beta6	0.2225	0.8239
beta7	-1.7599	0.0784
beta8	-0.7407	0.4589
beta11	-3.2041	0.0014
beta12	0.0314	0.9749
beta28	-0.4214	0.6734

Figure 5: Geweke Diagnostics testing convergence of Markov chain.

Effective Sample Sizes			
Parameter	ESS	Autocorrelation Time	Efficiency
beta0	1000.0	1.0000	1.0000
beta1	1127.6	0.8868	1.1276
beta2	1000.0	1.0000	1.0000
beta4	1012.4	0.9877	1.0124
beta5	1129.1	0.8857	1.1291
beta6	954.0	1.0482	0.9540
beta7	882.0	1.1338	0.8820
beta8	1000.0	1.0000	1.0000
beta11	1226.3	0.8155	1.2263
beta12	1014.1	0.9861	1.0141
beta28	971.7	1.0291	0.9717

Figure 6: Effective Samples Sizes for each parameter.

There are several things that we can tell from these tables. First and foremost, Figure 2 indicates that all the coefficients are statistically different from 0, reinforcing their inclusion in the model. Judging by the fact that the Monte Carlo Standard Error (MCSE) in Figure 3 is no more than 3.21% of the standard deviation for any of the parameters, we can conclude that the parameters are well-captured here. In terms of posterior autocorrelations, Figure 4, none of the Lag 1 autocorrelations are larger than 0.21 in absolute value, indicating that we have essentially removed any dependencies by way of thinning every 250th observation. Figure 5 outlines the Geweke diagnostics which indicate that almost all of the coefficients have converged. Lastly, Figure 6 tells us the effective sample size, or how many independent observations we have extracted from the Monte Carlo simulations. Since the effective sample sizes for each of the coefficients is close to the actual sample size, we can feel

confident that we have attained good mixing. The autocorrelation time in the second column of Figure 6 is inversely proportional to the effective sample size.

The trace plots, autocorrelation (ACF) plots and posterior densities for each of the parameters in the final model can be found in the Appendix.

One thing that should be pointed out about the Geweke diagnostics is that they are highly sensitive to large sample sizes. This is similar to issues found with the Shapiro-Wilk statistics in assessing normality in that slight deviations from normality will lead to rejecting the null hypothesis of normality. The same phenomenon is known to occur with the Geweke diagnostics. In order to show this, I took the entire Markov chain, including those 7500 observations that would ultimately be burned in, thinned by every 250th observation, and I broke down the chain into blocks of 30. Doing this resulted in 33 blocks with 30 observations in each. Figure 7 lays out 4 graphs breaking down exactly how the Markov chain looks for the β_1 coefficient. The distributions look very similar across blocks, indicating that, while the Geweke diagnostics would have us believe that convergence has not occurred, convergence has, in fact, been achieved. Table 7 outlines some summary statistics for each of the blocks. The same graphics and tables can be found in the Appendix for the other 3 coefficients ($\beta_4, \beta_7, \beta_{11}$) that were at least marginally significant according to the Geweke diagnostics.

Table 7: Summary statistics for $\beta_1(\gamma_{BirthDate})$.

Block #	Mean	St Dev	Minimum	Maximum
1	22.569	0.973	20.681	24.592
11	23.008	1.408	20.054	26.598
22	22.480	1.300	18.703	24.710
33	22.502	1.358	19.547	25.027

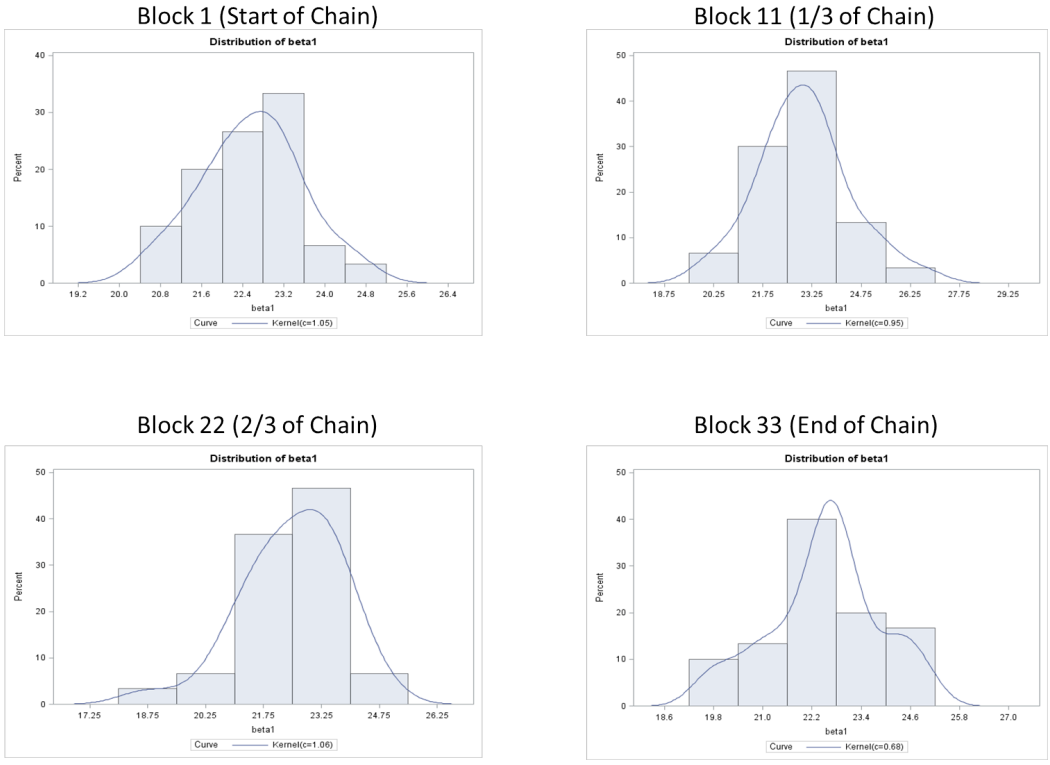

Figure 7: Breakdown of Markov chain for $\beta_1(\gamma_{BirthDate})$.

7.4 Finding Optimal Threshold

Once we homed in on what has become the final model, the last step became ascertaining the true value of the "optimal" threshold. Now, the term optimal deserves some attention, because it has been used in many different senses. For example, in Fellegi and Sunter (1969), they state that, "for fixed levels of error, the rule minimizes the probability of failing to make positive dispositions." In other words, they were homing in on what is

commonly referred to as "manual/clerical review". The algorithm Verykios et al. (2003) is optimal with respect to a particular cost matrix, in which false matches, false non-matches, and even clerical review, can be assigned various costs in order to minimize expenditure. This is a sort of generalization to the FS algorithm in that they assume the cost of a false match is not equal to that of a false non-match, which may be applicable depending on the business setting. In Judson (2006), it is stated that "$-\hat{\beta}_0$ is the optimally predictive threshold for the nonlinear threshold record linkage rule", in which this threshold is meant to ensure that if the probability of a match is larger than 0.50, we declare the pair a match. In this sense, "optimal" is not referring to manual review, for here, we are looking at only two partitions of the decision space (match or non-match). Rather, "optimal" is in reference to minimizing misclassification (false negative and false positive). It is in this sense that I am using optimality. Via the aforementioned "F-Score", I can measure the algorithm's ability not only to capture all the possible matches (Recall), but to measure the accuracy of those deemed matches (Precision).

While it may seem counterintuitive to consider any thresholds other than 0.5, Table 8 will provide you with the evidence needed to fully understand why my threshold is slightly different.

An obvious question here is, "Why would a counterintuitive threshold such as 0.15 be deemed optimal?" I offer up two possible theories for this. First, and foremost, the 0.50 threshold presented as optimal in Judson (2006) is merely a theoretical point at which one could define a subsequent threshold within the Bayesian logistic regression setting. In practice, none of their posterior probabilities of matching given the data ever made it above 0.50. Secondly, the Jaro-Winkler comparator scores yield many zeroes in terms of predictor variables for the model used here. As such, many matching probabilities will be essentially 0 with only a select few non-zero values. Those non-zero probabilities, for true matches, will range from close to 1 (where all fields match between records) all the way down to 0 (in cases where spousal information was utilized instead of patient information). Due to how right-skewed the posterior probability distribution is, we can pull that threshold a

Table 8: Determining probability threshold for final model.

Threshold	FPR	Recall	Precision	F1 Score
0.10	14.242	82.829	85.758	0.843
0.15	9.503	81.756	90.497	0.859
0.20	8.202	79.707	91.798	0.853
0.25	7.549	77.659	92.451	0.844
0.30	5.776	76.390	94.224	0.844
0.35	5.732	72.195	94.268	0.818
0.40	3.963	70.927	96.037	0.816
0.45	3.523	69.463	96.477	0.808
0.50	3.477	67.707	96.523	0.796
0.55	3.170	65.561	96.830	0.782
0.60	3.040	62.244	96.961	0.758
0.65	2.031	61.171	97.969	0.753
0.70	2.020	56.780	97.979	0.719
0.75	1.546	55.902	98.454	0.713
0.80	1.183	48.878	98.817	0.654
0.85	1.073	44.976	98.927	0.618
0.90	0.519	37.366	99.481	0.543
0.95	0.339	28.683	99.661	0.445

little closer to 0 in order to increase recall without losing too much in precision.

7.5 Interaction Effects

It was determined that three different interactions effects were statistically significant. As you can see in the Final Model section, those interaction effects are the following:

1. BirthDate*CrashZip
2. BirthDate*FirstSoundex
3. CrashZip*LastInitial

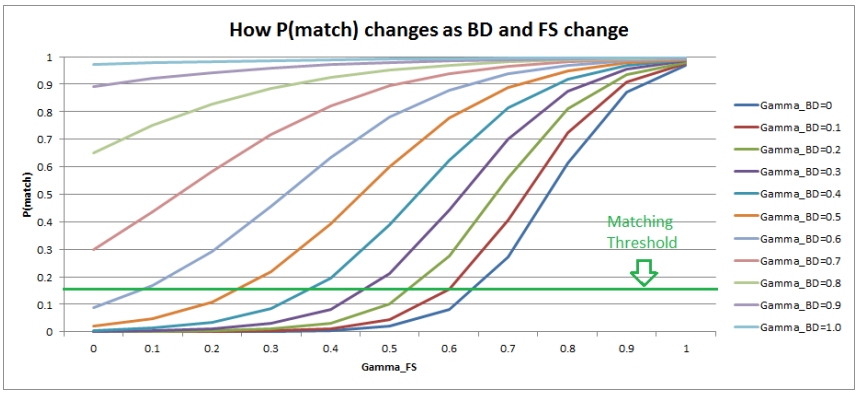

Figure 8: Interaction plot showing how BirthDate and FirstSoundex impact matching probability.

These three graphs highlight how the probability of matching is impacted across varying levels of the predictors. For each graph, if the variable is not inherently involved in the interaction, it is assumed to be equal to 1. The rationale behind this was that setting these values too low made the predictors of interest irrelevant, as too many non-matching fields yield very low probabilities of matching.

Figure 8 highlights how the probability of being declared a match changes for different values of γ_{FS} and γ_{BD}. What we can see here is that if the comparator score for BirthDate is 0, represented by the lowest (dark blue) line, then it takes a value of at least 0.65 from the FirstSoundex comparator score before that particular pair of records would be deemed a match. As you can see, the optimal threshold of 0.15 is indicated to highlight at what point records are classified as matches. However, if the comparator score for BirthDate is 1 (indicating completely matching birthdates), then the value of FirstSoundex becomes irrelevant, as the matching probability hovers around 1 regardless of FirstSoundex. Here, we can graphically see how the interaction between BirthDate and FirstSoundex plays out.

The following graphs highlight similar patterns for the other two interactions listed.

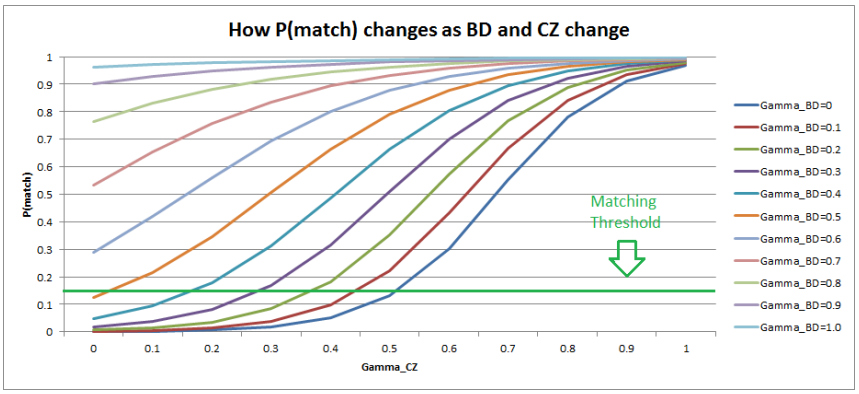

Figure 9: Interaction plot showing how BirthDate and CrashZip impact matching probability.

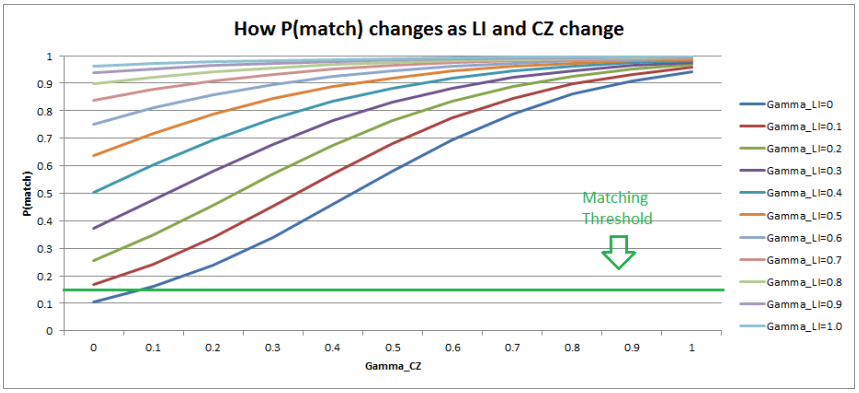

Figure 10: Interaction plot showing how LastInitial and CrashZip impact matching probability.

For the interaction plot between BirthDate and CrashZip in Figure 9, we see much of the same pattern as we saw between BirthDate and FirstSoundex, but not quite as strong. Case in point, where the value of Gamma_FirstSoundex needed to be above 0.65 in order for a pair to be deemed a match when Gamma_BirthDate was 0, now CrashZip only needs to be above 0.5. In the BirthDate*FirstSoundex setting, if BirthDate ≥ 0.7, the value of FirstSoundex was irrelevant as a match would have been declared. In the BirthDate*CrashZip setting, that boundary drops to BirthDate ≥ 0.6. As such, the differences are subtle, but they highlight why the interaction effect between BirthDate and FirstSoundex was deemed more significant than that between BirthDate and CrashZip.

With the interaction effect between LastInitial and CrashZip, indicated by Figure 10, we see that, for all intents and purposes, because the optimal threshold was found to be 0.15, the interaction is not of real consequence here. In essence, the only value of LastInitial that would make the value of CrashZip relevant in terms of declaring matches would be at 0. Now, while that may not seem significant, it should be noted that 95% of all the record pairs in this dataset were 0's, and there is little reason to believe this would not be the case in future datasets.

8 FUTURE WORK

While the methodologies discussed here have advanced the field of record matching beyond the restrictive binary structure of field-matching predictors, it does require training data. Judson (2006) ups the ante in his paper by considering a latent model approach. Not surprisingly, the success of that algorithm was considerably worse due to the lack of knowledge on matching status. It would be interesting to see how this Bayesian logistic regression model setup with the Jaro-Winkler comparator scores would perform in a latent model setting.

In addition to removing the knowledge we have on matching status, perhaps pursuing other prior distributions would yield even better results here, or at the very least, give the latent model somewhat of a headstart. All the prior distributions used in the current setup are noninformative by nature. One could apply the "today's posterior is tomorrow's prior" mentality, run the algorithm with non-informative priors, gain more informative posterior distributions, and re-run the algorithm on a new set of data whose structure mimics that of the training data. If subject domain expertise exists for helping build an adequate model, then we need not run record matching in a vacuum.

9 CONCLUSION

Much has been contributed to the field of record matching in the past 4+ decades since Ivan Fellegi and Alan Sunter blew the doors off the field in their ground-breaking article. Virtually every record matching algorithm either extends their algorithm or compares it to their own. The computational advantage and high accuracy of their algorithm are what have allowed them to remain mainstays far into the information age. However, everything must be done on a binary scale and that lends itself to unnecessary restriction. In addition, the computational advantages are afforded due to a sometimes statistically invalid assumption made by the authors in which they assume the conditional independence of the comparator vector distributions given matching status.

My algorithm makes no such assumptions regarding the conditional independence of the predictors given matching status. It does, however, inherit assumptions required by logistic regression. In particular, we have assumed that the logit of the odds of matching is a linear function of comparators and products of comparators.

The Bayesian logistic regression model allows us to incorporate some interpretability and flexibility into the scheme. From a business perspective, we can take what has been presented here and inform managers and technicians alike which variables to invest additional time in with regards to record matching. We have shown here that incorporating the string comparator scores leads to an improvement in terms of the ability to correctly identify true matching records. We have also shown that the intuitively chosen posterior probability of 0.50 does not yield optimality in terms of precision and recall. While implementing the Markov chain Monte Carlo schemes can take a considerable amount of time, especially if we are dealing with large datasets and many components, a common method for circumventing this issue is blocking, by which you are increasing the likelihood of false positives, but you are also increasing the recall rate of the algorithm.

Ultimately, what we would like is to take the model built by the training dataset and apply this to a real-life dataset, similar in structure to that of the training dataset, but

where matching status is unknown. Through collaborations with multiple entities, we could bring together disparate data sources and evaluate the success of the model on the new dataset.

The default proposal distribution utilized by the MCMC procedure in SAS is that of the multivariate normal. This is done due to the continuous nature of the prior distributions. This proposal distribution can be changed to the t-distribution in order to sample more from the tails. This can help with the mixing of the Markov chain if some of the parameters have a skewed tail.

There are many approaches to record matching, some statistical in nature, others based predominantly on computer science. This paper is meant primarily to provide an alternative approach to those restricted by their binary structures as well as to those looking for a leg up in terms of accurately extracting positive matches.

REFERENCES

Agresti, A. (2007), *An Introduction to Categorical Data Analysis (2nd ed.)*, John Wiley & Sons, Inc., Hoboken, NJ.

Belin, T. and Rubin, D. B. (1995), "A Method for Calibrating False-Match Rates in Record Linkage," *Journal of the American Statistical Association*, 90, 694–707.

Cortes, C. and Vapnik, V. (1995), "Support-vector networks," *Machine Learning*, 20, 273–297.

Fellegi, I. P. and Sunter, A. B. (1969), "A Theory for Record Linkage," *Journal of the American Statistical Association*, 64, 1183–1210.

Harville, D. S. and Moore, R. A. (1999), "Determining Record Linkage Parameters Using an Iterative Logistic Regression Approach," *Paper presented at the 22nd Joint Statistical Meetings, Baltimore, MD, August 8-12, 1999*, 689–694.

Herzog, T. N., Scheuren, F. J., and Winkler, W. E. (2007), *Data Quality and Record Linkage*, Spring Science + Business Media, LLC, New York.

Jaro, M. A. (1989), "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," *Journal of the American Statistical Association*, 84, 414–420.

Judson, D. H. (2006), *Data Mining and Knowledge Discovery Approaches Based on Rule Induction*, Sprinter Science + Business Media, LLC, New York.

McGlincy, M. H. (2006), "Using Test Databases to Evaluate Record Linkage Models and Train Linkage Practitioners," *Proceedings of the 29th American Statistical Association, Survey Research Method Section, Seattle, WA, August 6-10, 2006*, 3404–3410.

Ng, A. and Jordan, M. (2002), "On Discriminative vs Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes," *Neural Information Processing Systems*, 14, 841–848.

Schapire, R. E. (1999), "A Brief Introduction to Boosting," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Stockholm, Sweden, July 31 - August 6, 1999*, 1401–1406.

Verykios, Moustakides, G. V., and Elfeky, M. G. (2003), "A Bayesian Decision Model for Cost Optimal Record Matching," *The VLDB Journal*, 12, 28–40.

Winkler, W. E. (1990), "String Comparator Metrics and Enhanced Decision rules in the Fellegi-Sunter Model of Record Linkage," *Proceedings of the 13th Section on Survey Research Methods, Anaheim, CA, August 6-9, 1990*, 354–359.

— (2006), "Overview of Record Linkage and Current Research Directions," *Research Report Series, Statistical Research Division, U.S. Census Bureau*, 2, 1–44.

APPENDIX

PROC MCMC Sample Code

Below is sample code for the MCMC procedure in the SAS software:

```
proc mcmc data=diss.y_and_gamma outpost=diss.posterior nmc=500
thin=5 seed=246810;
parms (beta0 beta1 beta2 beta3) 0;
prior beta0 beta1 beta2 beta3 ~ normal(mean = 0, var = 1e6);
p = logistic(beta0 + beta1*Gamma_BirthDate + beta2*Gamma_SSN +
 beta3*Gamma_FirstSoundex;
model Y ~ binary(p);
preddist outpred=diss.PPD nsim=50;
run;
```

Geweke Diagnostics Graphical Evidence

This section presents the same information found in Figure 7 and Table 7 in the Results Section, specific to the other three coefficients found to be statistically significant ($\beta_4(\gamma_{CrashZip}), \beta_7(\gamma_{PartInj})$ and $\beta_{11}(\gamma_{BirthDate*CrashZip})$)

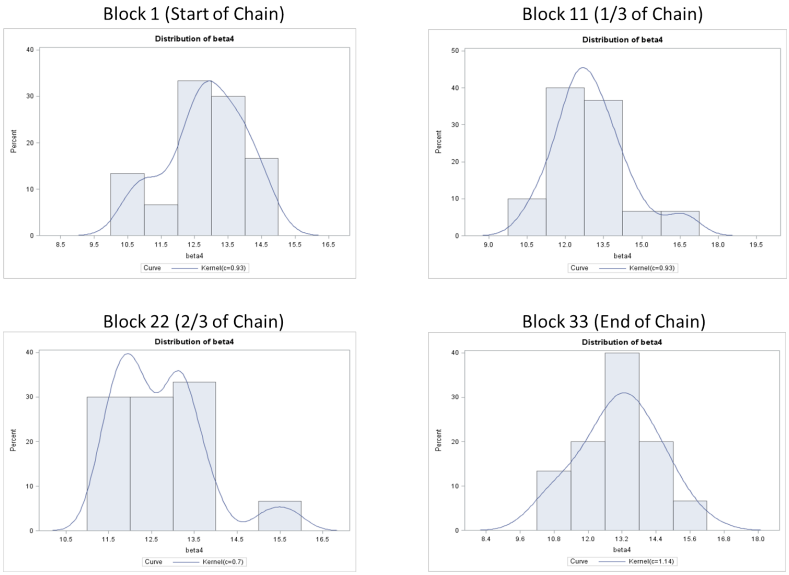

Figure 11: Breakdown of Markov chain for $\beta_4(\gamma_{CrashZip})$.

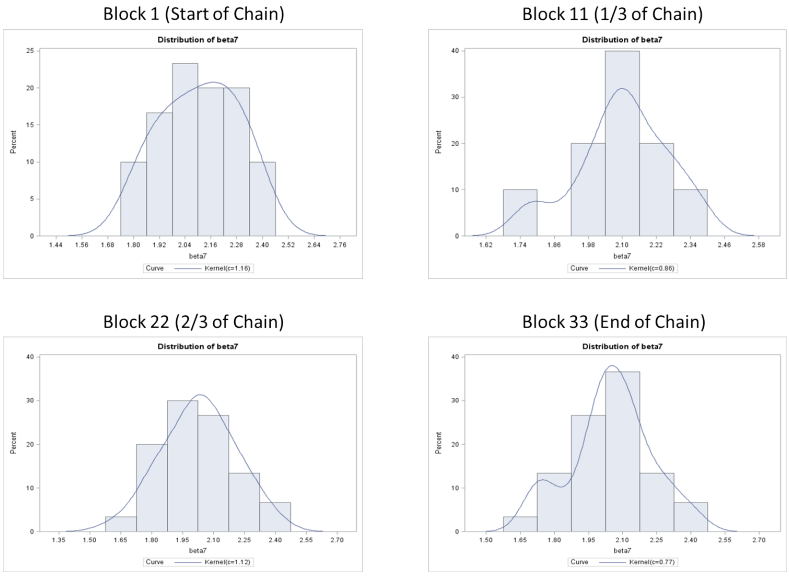

Figure 12: Breakdown of Markov chain for $\beta_7(\gamma_{PartInj})$.

Table 9: Summary statistics for $\beta_4(\gamma_{CrashZip})$.

Block #	Mean	St Dev	Minimum	Maximum
1	12.839	1.135	10.518	14.661
11	13.052	1.412	10.529	16.736
22	12.703	1.059	11.320	15.766
33	13.109	1.379	10.377	16.032

Table 10: Summary statistics for $\beta_7(\gamma_{PartInj})$.

Block #	Mean	St Dev	Minimum	Maximum
1	2.106	0.176	1.796	2.388
11	2.101	0.158	1.774	2.357
22	2.030	0.177	1.637	2.367
33	2.050	0.177	1.699	2.415

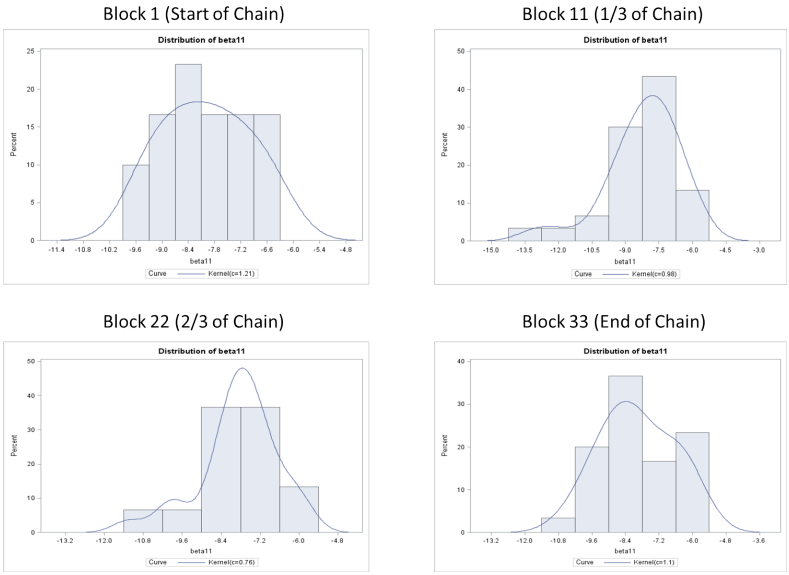

Figure 13: Breakdown of Markov chain for $\beta_{11}(\gamma_{BirthDate*CrashZip})$.

Final Model Parameter Diagnostics

Here, you will find the paramter diagnostics referenced in the Results Section.

Table 11: Summary statistics for $\beta_{11}(\gamma_{BirthDate*CrashZip})$.

Block #	Mean	St Dev	Minimum	Maximum
1	-7.994	0.983	-9.511	-6.414
11	-8.218	1.626	-13.106	-5.822
22	-7.880	1.217	-11.280	-5.755
33	-7.982	1.305	-10.620	-5.597

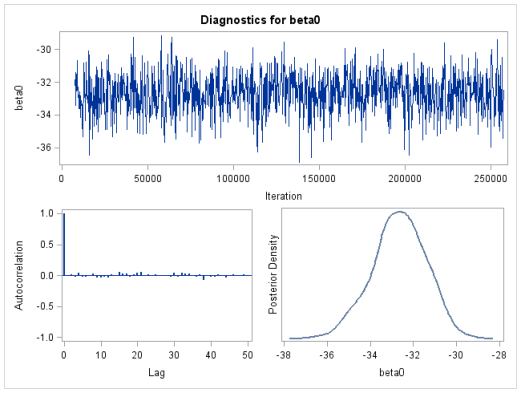

Figure 14: Trace plot, ACF and posterior density estimate for β_0 (Intercept).

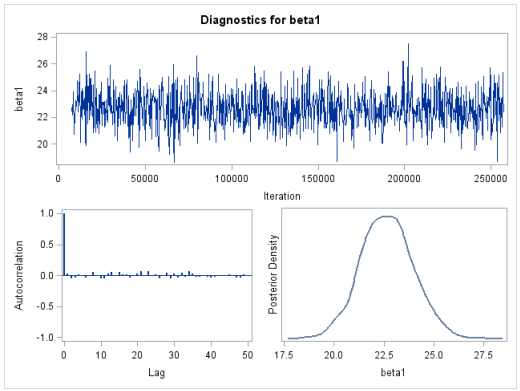

Figure 15: Trace plot, ACF and posterior density estimate for β_1 (BirthDate).

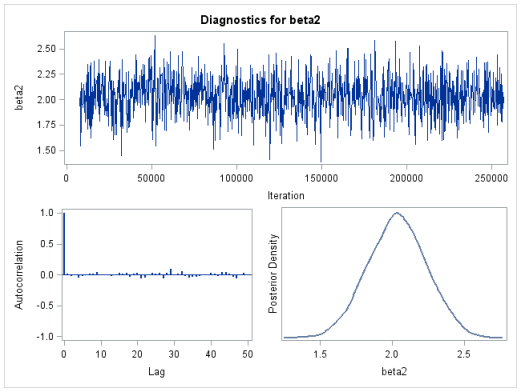

Figure 16: Trace plot, ACF and posterior density estimate for β_2 (Collide).

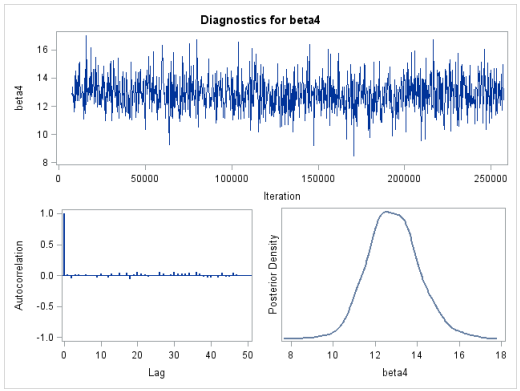

Figure 17: Trace plot, ACF and posterior density estimate for β_4 (CrashZip).

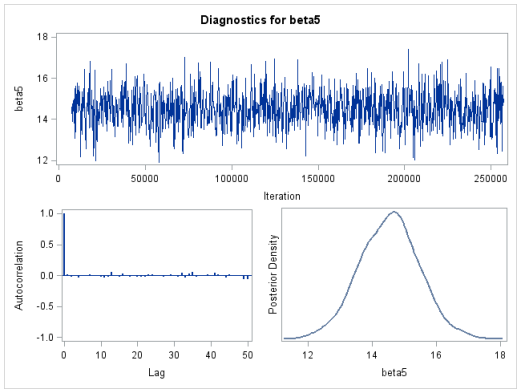

Figure 18: Trace plot, ACF and posterior density estimate for β_5 (FirstSoundex).

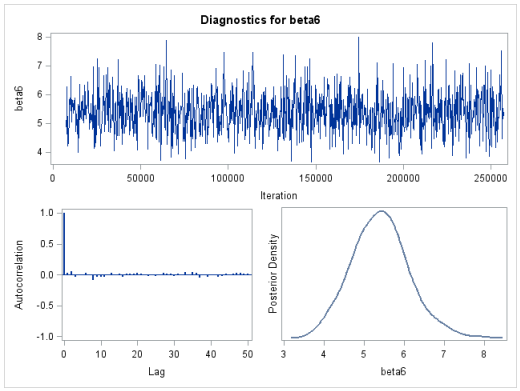

Figure 19: Trace plot, ACF and posterior density estimate for β_6 (LastInitial).

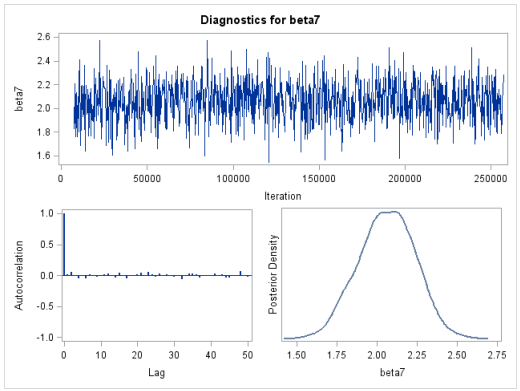

Figure 20: Trace plot, ACF and posterior density estimate for β_7 (PartInj).

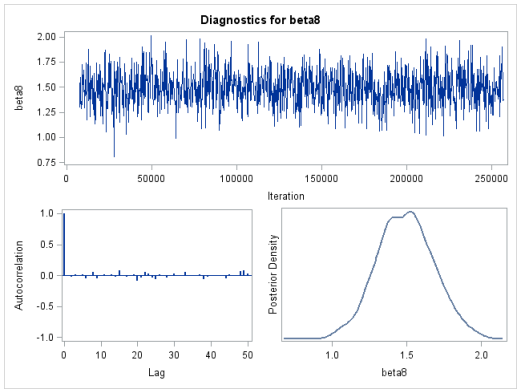

Figure 21: Trace plot, ACF and posterior density estimate for β_8 (Safety).

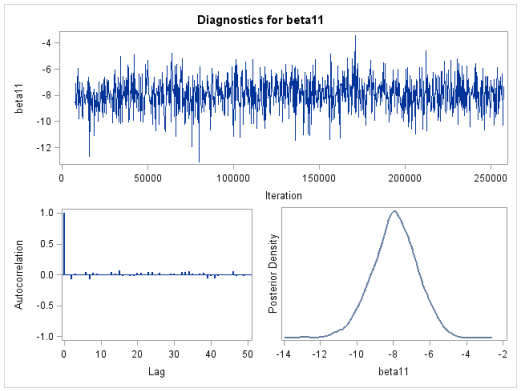

Figure 22: Trace plot, ACF and posterior density estimate for β_{11} (BirthDate * CrashZip).

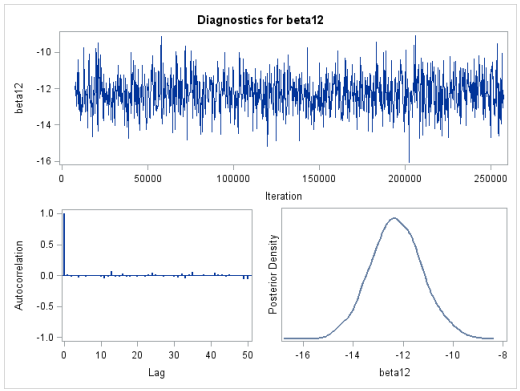

Figure 23: Trace plot, ACF and posterior density estimate for β_{12} (BirthDate * FirstSoundex).

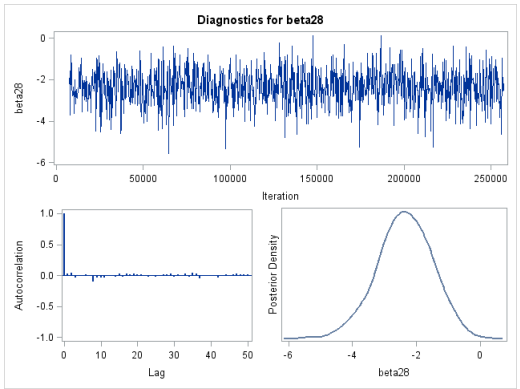

Figure 24: Trace plot, ACF and posterior density estimate for β_{28} (CrashZip * LastInitial).

Equations for Fellegi-Sunter Algorithm

The following notation refers to the frequencies of various configurations of γ, the vector of comparison components. Since they are not conditional frequencies, they can be obtained as direct counts by comparing the files A and B:

M_k: the proportion of "agreement" in all components except the kth; any configuration in the kth component

U_k: the proportion of "agreement" in the kth component; any configuration in the others

M: the proportion of "agreement" in all components

The following are the equations which form the basis for Fellegi and Sunter's weighting scheme:

$$M_k = \frac{N}{N_A N_B} \prod_{j=1, j\neq k}^{3} m_j + \frac{N_A N_B - N}{N_A N_B} \prod_{j=1, j\neq k}^{3} u_j; \quad k = 1, 2, 3 \tag{51}$$

$$U_k = \frac{N}{N_A N_B} m_k + \frac{N_A N_B - N}{N_A N_B} u_k \tag{52}$$

$$M = \frac{N}{N_A N_B} \prod_{j=1}^{3} m_j + \frac{N_A N_B - N}{N_A N_B} \prod_{j=1}^{3} u_j. \tag{53}$$

We introduce the transformation

$$m_k^* = m_k - U_k \tag{54}$$

and

$$u_k^* = u_k - U_k. \tag{55}$$

Substituting m_k and u_k from (38) and (39) into (36) we obtain

$$\frac{N}{N_A N_B} m_k^* + \frac{N_A N_B - N}{N_A N_B} u_k^* = 0. \tag{56}$$

Substituting (38) and (39) into (35) and then substituting in the resulting equations u_k^* from (40) we obtain

$$\prod_{j=1,j\neq k}^{3} m_j^* = \frac{N_A N_B - N}{N}\left[M_k - \prod_{j=1,j\neq k}^{3} U_j\right],\ \ k = 1,2,3. \tag{57}$$

Denoting

$$R_k = M_k - \prod_{j=1,j\neq k}^{3} U_j,\ \ k = 1,2,3, \tag{58}$$

we obtain by multiplying the three equations under (41) and by taking square roots

$$\prod_{j=1}^{3} m_j^* = \left(\frac{N_A N_B - N}{N}\right)^{\frac{3}{2}}\left(\prod_{j=1}^{3} R_j\right)^{\frac{1}{2}}. \tag{59}$$

Dividing (43) by (41) and putting

$$X = \sqrt{(N_A N_B - N)/N} \tag{60}$$

$$B_k = \sqrt{\prod_{j=1,j\neq k}^{3} R_j/R_k},\ \ k = 1,2,3, \tag{61}$$

we get

$$m_k^* = B_k X,\ \ k = 1,2,3, \tag{62}$$

and, from (38) to (40),

$$m_k = U_k + B_k X,\ \ k = 1,2,3, \tag{63}$$

$$u_k = U_k - B_k/X,\ \ k = 1,2,3. \tag{64}$$

We can now substitute into (37) m_k and u_k from (47) and (48) respectively and N as expressed from (44). We obtain

$$\frac{1}{X^2+1}\prod_{j=1}^{3}(U_j+B_jX)+\frac{X^2}{X^2+1}(U_j-B_j/X)=M. \tag{65}$$

After expanding (49), some cancellations and substitution of B_k from (45), we get the following quadratic equation in X:

$$\sqrt{\prod_{j=1}^{3}R_j}(X^2-1)+\left[\prod_{j=1}^{3}U_j+\sum_{i=1}^{3}R_jU_j-M\right]X=0. \tag{66}$$

The positive root of this equation is

$$X=\left\{M-\sum_{j=1}^{3}R_jU_j-\prod_{j=1}^{3}U_j+\sqrt{\left[M-\sum_{j=1}^{3}R_jU_j-\prod_{j=1}^{3}U_j\right]^2+4\prod_{j=1}^{3}R_j}\right\}/2\sqrt{\prod_{j=1}^{3}R_j}. \tag{67}$$

The estimates of m_k, u_k and N are now easily obtained from (44), (47) and (48).

Having solved these equations, we can proceed to estimate the specific values of $m(\gamma)$ and $u(\gamma)$ which are required. We introduce some additional notation which, as before, refers to observable frequencies:

$M_k(\gamma_i^k) =$ the proportion of "agreement" in all components except the kth, the specific configuration γ_i^k in the kth component

$U_1(\gamma_i^2) =$ the proportion of "agreement" in the first, γ_i^2 in the second and any configuration in the third component

$U_1(\gamma_i^3) =$ the proportion of "agreement" in the first, any configuration in the second component and γ_i^3 in the third

$U_2(\gamma_i^1) =$ the proportion of γ_i^1 in the first, "agreement" in the second and any configuration in the third component

The required values of $m(\gamma_i^k)$ and $u(\gamma_i^k)$ are estimated as

$$m(\gamma_i^1)=\frac{M_1(\gamma_i^1)-u_3U_2(\gamma_i^1)}{m_2(m_3-u_3)}(X^2+1) \tag{68}$$

$$m(\gamma_i^2) = \frac{M_2(\gamma_i^2) - u_3 U_1(\gamma_i^2)}{m_1(m_3 - u_3)}(X^2 + 1) \tag{69}$$

$$m(\gamma_i^3) = \frac{M_3(\gamma_i^3) - u_2 U_1(\gamma_i^3)}{m_1(m_2 - u_2)}(X^2 + 1) \tag{70}$$

$$u(\gamma_i^1) = \frac{m_3 U_2(\gamma_i^1) - M_1(\gamma_i^1)}{u_2(m_3 - u_3)}\frac{X^2 + 1}{X^2} \tag{71}$$

$$u(\gamma_i^2) = \frac{m_3 U_1(\gamma_i^2) - M_2(\gamma_i^2)}{u_1(m_3 - u_3)}\frac{X^2 + 1}{X^2} \tag{72}$$

$$u(\gamma_i^3) = \frac{m_2 U_1(\gamma_i^3) - M_3(\gamma_i^3)}{u_1(m_2 - u_2)}\frac{X^2 + 1}{X^2}. \tag{73}$$

The formulae (52) to (57) are easily verified by expressing the expected values of the quantities $M_k(\gamma_i^k)$, $U_1(\gamma_i^2)$, etc. in terms of m_k, u_k, $m(\gamma_i^k)$ and $u(\gamma_i^k)$, dropping the expected values and solving the resulting equations (there will be two equations for each pair $m(\gamma_i^k)$ and $u(\gamma_i^k)$). The necessary and sufficient conditions for the mechanical validity of the formulae in this section are that

$$m_k \neq u_k, \;\; k = 1, 2, 3$$

and

$$R_k > 0, \;\; k = 1, 2, 3.$$

Since

$$m_k = m(S_k) = Pr(S_k|M)$$

and

$$u_k = u(S_k) = Pr(S_k|U)$$

clearly for sensible definitions of "agreement" $m_k > u_k$ should hold for $k = 1, 2, 3$. In this case, $R_k > 0$ will hold as well. The latter statement can easily be verified by substituting (1) and (2) into (8).

Complete Datasets from Background Section

Here you will find the complete data alluded to in Table 1 and utilized in Table 2 for the construction of the m- and u-probabilities.

Table 12: Complete data used in calculating m- and u-probabilities.

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
Chris	Johnson	F	Chris	Johnson	F	111
Chris	Johnson	F	Robert	Ordway	M	000
Chris	Johnson	F	Kathryn	Lowry	F	001
Chris	Johnson	F	Mable	Martinez	F	001
Chris	Johnson	F	Joanne	Fowler	F	001
Chris	Johnson	F	David	Miller	M	000
Chris	Johnson	F	Ryan	Costello	M	000
Chris	Johnson	F	Luis	Smith	M	000
Chris	Johnson	F	Rosa	Sawyer	F	001
Chris	Johnson	F	Charles	Hernandez	M	000
Chris	Johnson	F	Chris	Johnson	M	110
Chris	Johnson	F	Robert	Mills	M	000
Chris	Johnson	F	Robert	Simons	F	001
Chris	Johnson	F	Billy	Hernandez	M	000
Chris	Johnson	F	Joan	Ordway	F	001
Chris	Johnson	F	Dominic	Jann	M	000
Robert	Ordway	M	Chris	Johnson	F	000
Robert	Ordway	M	Robert	Ordway	M	111
Robert	Ordway	M	Kathryn	Lowry	F	000

Table 12: continued

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
Robert	Ordway	M	Mable	Martinez	F	000
Robert	Ordway	M	Joanne	Fowler	F	000
Robert	Ordway	M	David	Miller	M	001
Robert	Ordway	M	Ryan	Costello	M	001
Robert	Ordway	M	Luis	Smith	M	001
Robert	Ordway	M	Rosa	Sawyer	F	000
Robert	Ordway	M	Charles	Hernandez	M	001
Robert	Ordway	M	Chris	Johnson	M	001
Robert	Ordway	M	Robert	Mills	M	101
Robert	Ordway	M	Robert	Simons	F	100
Robert	Ordway	M	Billy	Hernandez	M	001
Robert	Ordway	M	Joan	Ordway	F	010
Robert	Ordway	M	Dominic	Jann	M	001
Kathryn	Lowry	F	Chris	Johnson	F	001
Kathryn	Lowry	F	Robert	Ordway	M	000
Kathryn	Lowry	F	Kathryn	Lowry	F	111
Kathryn	Lowry	F	Mable	Martinez	F	001
Kathryn	Lowry	F	Joanne	Fowler	F	001
Kathryn	Lowry	F	David	Miller	M	000
Kathryn	Lowry	F	Ryan	Costello	M	000
Kathryn	Lowry	F	Luis	Smith	M	000
Kathryn	Lowry	F	Rosa	Sawyer	F	001
Kathryn	Lowry	F	Charles	Hernandez	M	000
Kathryn	Lowry	F	Chris	Johnson	M	000
Kathryn	Lowry	F	Robert	Mills	M	000

Table 12: continued

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
Kathryn	Lowry	F	Robert	Simons	F	001
Kathryn	Lowry	F	Billy	Hernandez	M	000
Kathryn	Lowry	F	Joan	Ordway	F	001
Kathryn	Lowry	F	Dominic	Jann	M	000
Mable	Martinez	F	Chris	Johnson	F	001
Mable	Martinez	F	Robert	Ordway	M	000
Mable	Martinez	F	Kathryn	Lowry	F	001
Mable	Martinez	F	Mable	Martinez	F	111
Mable	Martinez	F	Joanne	Fowler	F	001
Mable	Martinez	F	David	Miller	M	000
Mable	Martinez	F	Ryan	Costello	M	000
Mable	Martinez	F	Luis	Smith	M	000
Mable	Martinez	F	Rosa	Sawyer	F	001
Mable	Martinez	F	Charles	Hernandez	M	000
Mable	Martinez	F	Chris	Johnson	M	000
Mable	Martinez	F	Robert	Mills	M	000
Mable	Martinez	F	Robert	Simons	F	001
Mable	Martinez	F	Billy	Hernandez	M	000
Mable	Martinez	F	Joan	Ordway	F	001
Mable	Martinez	F	Dominic	Jann	M	000
Joanne	Fowler	F	Chris	Johnson	F	001
Joanne	Fowler	F	Robert	Ordway	M	000
Joanne	Fowler	F	Kathryn	Lowry	F	001
Joanne	Fowler	F	Mable	Martinez	F	001
Joanne	Fowler	F	Joanne	Fowler	F	111

Table 12: continued

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
Joanne	Fowler	F	David	Miller	M	000
Joanne	Fowler	F	Ryan	Costello	M	000
Joanne	Fowler	F	Luis	Smith	M	000
Joanne	Fowler	F	Rosa	Sawyer	F	001
Joanne	Fowler	F	Charles	Hernandez	M	000
Joanne	Fowler	F	Chris	Johnson	M	000
Joanne	Fowler	F	Robert	Mills	M	000
Joanne	Fowler	F	Robert	Simons	F	001
Joanne	Fowler	F	Billy	Hernandez	M	000
Joanne	Fowler	F	Joan	Ordway	F	001
Joanne	Fowler	F	Dominic	Jann	M	000
David	Miller	M	Chris	Johnson	F	000
David	Miller	M	Robert	Ordway	M	001
David	Miller	M	Kathryn	Lowry	F	000
David	Miller	M	Mable	Martinez	F	000
David	Miller	M	Joanne	Fowler	F	000
David	Miller	M	David	Miller	M	111
David	Miller	M	Ryan	Costello	M	001
David	Miller	M	Luis	Smith	M	001
David	Miller	M	Rosa	Sawyer	F	000
David	Miller	M	Charles	Hernandez	M	001
David	Miller	M	Chris	Johnson	M	001
David	Miller	M	Robert	Mills	M	001
David	Miller	M	Robert	Simons	F	000
David	Miller	M	Billy	Hernandez	M	001

Table 12: continued

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
David	Miller	M	Joan	Ordway	F	001
David	Miller	M	Dominic	Jann	M	001
Ryan	Costello	M	Chris	Johnson	F	000
Ryan	Costello	M	Robert	Ordway	M	001
Ryan	Costello	M	Kathryn	Lowry	F	000
Ryan	Costello	M	Mable	Martinez	F	000
Ryan	Costello	M	Joanne	Fowler	F	000
Ryan	Costello	M	David	Miller	M	001
Ryan	Costello	M	Ryan	Costello	M	111
Ryan	Costello	M	Luis	Smith	M	001
Ryan	Costello	M	Rosa	Sawyer	F	000
Ryan	Costello	M	Charles	Hernandez	M	001
Ryan	Costello	M	Chris	Johnson	M	001
Ryan	Costello	M	Robert	Mills	M	001
Ryan	Costello	M	Robert	Simons	F	000
Ryan	Costello	M	Billy	Hernandez	M	001
Ryan	Costello	M	Joan	Ordway	F	000
Ryan	Costello	M	Dominic	Jann	M	001
Luis	Smith	M	Chris	Johnson	F	000
Luis	Smith	M	Robert	Ordway	M	001
Luis	Smith	M	Kathryn	Lowry	F	000
Luis	Smith	M	Mable	Martinez	F	000
Luis	Smith	M	Joanne	Fowler	F	000
Luis	Smith	M	David	Miller	M	001
Luis	Smith	M	Ryan	Costello	M	001

Table 12: continued

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
Luis	Smith	M	Luis	Smith	M	001
Luis	Smith	M	Rosa	Sawyer	F	000
Luis	Smith	M	Charles	Hernandez	M	001
Luis	Smith	M	Chris	Johnson	M	001
Luis	Smith	M	Robert	Mills	M	001
Luis	Smith	M	Robert	Simons	F	000
Luis	Smith	M	Billy	Hernandez	M	001
Luis	Smith	M	Joan	Ordway	F	000
Luis	Smith	M	Dominic	Jann	M	001
Rosa	Sawyer	F	Chris	Johnson	F	001
Rosa	Sawyer	F	Robert	Ordway	M	000
Rosa	Sawyer	F	Kathryn	Lowry	F	001
Rosa	Sawyer	F	Mable	Martinez	F	001
Rosa	Sawyer	F	Joanne	Fowler	F	001
Rosa	Sawyer	F	David	Miller	M	000
Rosa	Sawyer	F	Ryan	Costello	M	000
Rosa	Sawyer	F	Luis	Smith	M	000
Rosa	Sawyer	F	Rosa	Sawyer	F	111
Rosa	Sawyer	F	Charles	Hernandez	M	000
Rosa	Sawyer	F	Chris	Johnson	M	000
Rosa	Sawyer	F	Robert	Mills	M	000
Rosa	Sawyer	F	Robert	Simons	F	001
Rosa	Sawyer	F	Billy	Hernandez	M	000
Rosa	Sawyer	F	Joan	Ordway	F	001
Rosa	Sawyer	F	Dominic	Jann	M	000

Table 12: continued

A_{Fname}	A_{Lname}	A_{Gender}	B_{Fname}	B_{Lname}	B_{Gender}	$Vector$
Charles	Hernandez	M	Chris	Johnson	F	000
Charles	Hernandez	M	Robert	Ordway	M	001
Charles	Hernandez	M	Kathryn	Lowry	F	000
Charles	Hernandez	M	Mable	Martinez	F	000
Charles	Hernandez	M	Joanne	Fowler	F	000
Charles	Hernandez	M	David	Miller	M	001
Charles	Hernandez	M	Ryan	Costello	M	001
Charles	Hernandez	M	Luis	Smith	M	001
Charles	Hernandez	M	Rosa	Sawyer	F	000
Charles	Hernandez	M	Charles	Hernandez	M	111
Charles	Hernandez	M	Chris	Johnson	M	001
Charles	Hernandez	M	Robert	Mills	M	001
Charles	Hernandez	M	Robert	Simons	F	000
Charles	Hernandez	M	Billy	Hernandez	M	011
Charles	Hernandez	M	Joan	Ordway	F	000
Charles	Hernandez	M	Dominic	Jann	M	001