



2006-06-29

Computation of Weights for Probabilistic Record Linkage Using the EM Algorithm

G. John Bauman

Brigham Young University - Provo

Follow this and additional works at: <http://scholarsarchive.byu.edu/etd>



Part of the [Statistics and Probability Commons](#)

BYU ScholarsArchive Citation

Bauman, G. John, "Computation of Weights for Probabilistic Record Linkage Using the EM Algorithm" (2006). *All Theses and Dissertations*. Paper 746.

This Selected Project is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu.

COMPUTATION OF WEIGHTS FOR PROBABILISTIC RECORD
LINKAGE USING THE EM ALGORITHM

by

G. John Bauman Jr.

A project submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Statistics

Brigham Young University

August 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a project submitted by

G. John Bauman Jr.

This project has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date

John Lawson, Chair

Date

David Whiting

Date

Del Scott

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the project of G. John Bauman Jr. in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

John Lawson
Chair, Graduate Committee

Accepted for the Department

Scott Grimshaw
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and
Mathematical Sciences

ABSTRACT

COMPUTATION OF WEIGHTS FOR PROBABILISTIC RECORD LINKAGE USING THE EM ALGORITHM

G. John Bauman Jr.

Department of Statistics

Master of Science

Record linkage is the process of combining information about a single individual from two or more records. Probabilistic record linkage gives weights to each field that is compared. The decision of whether the records should be linked is then determined by the sum of the weights, or “Score”, over all fields compared. Using methods similar to the simple versus simple most powerful test, an optimal record linkage decision rule can be established to minimize the number of unlinked records when the probability of false positive and false negative errors are specified.

The weights needed for probabilistic record linkage necessitate linking a “training” subset of records for the computations. This is not practical in many settings, as hand matching requires a considerable time investment. In 1989, Matthew A. Jaro demonstrated how the Expectation-Maximization, or EM, algorithm could be used to compute the needed weights when fields have Binomial matching possibilities. This project applies this method of using the EM algorithm to calculate weights for head-of-household records from the 1910 and 1920 Censuses for Ascension Parish of

Louisiana and Church and County Records from Perquimans County, North Carolina. This project also expands the Jaro's EM algorithm to a Multinomial framework. The performance of the EM algorithm for calculating weights will be assessed by comparing the computed weights to weights computed by clerical matching. Simulations will also be conducted to investigate the sensitivity of the algorithm to the total number of record pairs, the number of fields with missing entries, the starting values of estimated probabilities, and the convergence epsilon value.

ACKNOWLEDGMENTS

I would like to express my appreciation to Dr. John Lawson for introducing this project to me, for chairing my project committee and for his encouragement, guidance, and patience during this project. I am appreciative and indebted to Nathan Orien for his coding work that allowed this project to be finished quickly. I am grateful to my family for their continued support, patience and love as I have pursued my degree. I would like to acknowledge my father-in-law and the Kuykendalls for introducing me to the field of statistics. Finally, I would like to thank my Heavenly Father for giving me the talents that have enabled me to finish this degree.

Contents

Acknowledgments	vi
List of Tables	ix
List of Figures	xi
1 Introduction	1
2 Literature Review	3
2.1 History	3
2.2 EM Algorithm	8
2.3 Modern Applications	8
2.4 Threshold Values	10
3 Methodology	13
3.1 Introduction	13
3.2 Blocking	13
3.3 Pair Comparisons	14
3.4 EM Algorithm	16
3.5 Alternate Method for Computing $\hat{\mathbf{u}}$	18
3.6 Missing Data	18
3.7 Weights and Threshold Values	20
4 Perquimans County Results	23
4.1 Pairing and Coding	23
4.2 Results from the EM Algorithm	25

4.3	Simulation	26
5	Louisiana Results	29
5.1	Pairing and Coding	29
5.2	Results from the EM Algorithm	31
5.3	Simulation	32
5.3.1	Total Number of Record Pairs	33
5.3.2	Missing Values	35
5.3.3	Starting Values	36
5.3.4	Epsilon Values	39
6	Conclusions and Future Work	41
6.1	Future Work	43
	Bibliography	45
A	SAS Code for Perquimans County Data Pairing and Coding	47
B	SAS Code for Perquimans County EM Algorithm	53
C	SAS Code for Data Generation with the Perquimans County Simulation	57
D	SAS Code for Perquimans County Simulation EM Algorithm	59
E	SAS Code for Louisiana Data Pairing and Coding	65
F	SAS Code for Louisiana EM Algorithm	77
G	SAS Code for Louisiana Simulations	81
H	SAS Code for Louisiana Grid Simulations	89
I	SAS Code for Boxplots	93

List of Tables

3.1	Simulation Parameters	19
4.1	EM and Clerical Estimates of \mathbf{m} and \mathbf{u} for the Perquimans County Records	26
4.2	Parameters and Estimates From the Perquimans Simulation	27
5.1	EM and Clerical Estimates of \mathbf{m} and \mathbf{u} for the Louisiana Data.	31
5.2	Starting Values and Simulation Parameters	38
5.3	Estimates for Name Same and Name Different for Changing Starting Values.	39
5.4	Estimates for Name Same and Name Different for Changing Epsilon Values.	40

List of Figures

3.1	Prototype Family History Search Form	20
4.1	Example of SQL pairing	24
5.1	Boxplots of Ten Simulation Estimates of m . Total Pairs: 10,050 . . .	33
5.2	Boxplots of Ten Simulation Estimates of m . Total Pairs: 100,500 . .	34
5.3	Boxplots of Ten Simulation Estimates of m . Total Pairs: 502,500 . .	34
5.4	Boxplots of Simulation Estimates for Pairs with Missing Entries Coded as Different.	36
5.5	Boxplots of Simulation Estimates for Pairs with Missing Entries Coded as Missing.	37
5.6	Boxplots of Simulation Estimates for Pairs with Missing Entries Sub- stituted with Estimates from u	37

Chapter 1

Introduction

Record linkage is the process of combining information about a single individual from two or more records. There are many needs for linking records such as medical records or census information. Ancestral research is one application that depends on linking records so that the history of specific individuals can be pieced together. Traditionally, especially in ancestral records, record linkage is done by hand. This method usually involves a person viewing the records and trying to decipher whether, based on the information given, the two records represent the same individual.

One method for improving the traditional method of hand matching records is to use probabilistic record linkage. Probabilistic record linkage utilizes weights given to each field compared. Positive weights are assigned to fields that match and negative weights are assigned to fields that do not match. The sum of the weights for all fields, or “Score”, is compared to threshold values in order to determine if the records are a match. This is advantageous because not all fields give you the same amount of information. For example, matching genders would not carry the same weight as matching last names. Using this method of probabilistic record linkage also has the desirable property of minimizing the amount of unclassified pairs when two threshold values are specified. The threshold values are cutoff points that are aimed at making the probabilities of linking unmatched records and not linking matched records as small as possible. They can be chosen by considering the tradeoff between the two error probabilities and the non-classification rate. This optimization property is similar to the way a classical Neyman-Pearson simple versus simple hypothesis test

maximizes the power of the test when the Type I error rate is specified.

The one fault of this probabilistic method is that it still requires some clerical work to establish weights. A ‘training’ set of records from two files are matched by hand in order to compute the weights. After that is done, the weights are used for the rest of the records in the two files. However, if new groups of records are going to be compared, a new training set must be hand-matched in order to calculate new weights. The necessity of having to match training sets by hand makes probabilistic record linkage only a small improvement over traditional hand matching if new groups of records are frequently being added to the set of records available for comparison.

This project applies the Expectation-Maximization, or EM, algorithm to data sets where weights have been previously calculated in order to eliminate the need for a ‘training’ data set. Previous uses of the EM algorithm have been demonstrated; however, in those cases each field had the binomial outcomes of “Same” or “Different”. This project will extend the EM algorithm to allow multiple levels of outcomes when comparing fields of different records. The EM algorithm will be used with two datasets to calculate weights. First, it will be applied to Church and County Records from Perquimans County, North Carolina using the binomial method. Second, for head-of-household records from the 1910 and 1920 Censuses for Ascension Parish of Louisiana, the multinomial method will be applied.

Using the EM algorithm to calculate weights would be invaluable in an environment where new groups of records are frequently being compared. One application that would see immediate benefit are websites that allow people to search for ancestral information in census records. Records from ancestral searches could be returned in rankings based off the probabilistic record linkage “Score” computed by comparing the records to the search criterion. Also, eliminating the setup necessary with traditional probabilistic linkage would allow new databases to be added and weights updated with great ease.

Chapter 2

Literature Review

2.1 History

Record linkage got its start in 1946 when Dr. Halbert Dunn, chief of the U.S. National Office of Vital Statistics, used the term to refer to linking vital records, such as birth and death certificates, pertaining to a single individual [Dunn 1946]. Computerized record linkage followed in 1959 when Newcombe et al. used computers to link vital records in an effort to track hereditary diseases [Newcombe et al. 1959]. From that point, the study of record linkage and efforts to make it efficient really began.

Newcombe’s efforts seem to be focused on promoting computers as a viable resource to do record linkage. He saw that there was valuable information that could be obtained by bringing records together but that the “high cost of searching manually” [Newcombe et al. 1959] had been a deterrent to using that information. Using a computer to do record linkage required that the program do the linkage rapidly, cheaply, and accurately. Much of Newcombe’s writing deals with optimizing these criteria.

Record linkage requires two steps. First, there is a searching operation that brings linkable records together for comparison. In this step it is important to minimize failure to bring together potentially linkable records while at the same time minimizing the number of comparisons between unlinkable records. The second step in record linkage is a detailed comparison used to determine if the pair of records can be linked. To optimize this step there are two types of errors, accepting false

linkages and rejecting true linkages, that should be minimized according to the use of the program [Newcombe and Kennedy 1962].

To help the optimization of the searching operation Newcombe discusses the idea of creating subdivisions to arrange the files in some orderly sequence so that comparisons are only conducted on a small portion of the two files [Newcombe 1967]. In Newcombe's example, he is comparing single incoming records against a master file. This means that each single record is only compared to a small portion of the master file. In this project, two files with many records will be compared in order to determine a linkage rule. Subdividing, or blocking, the two files will reduce the number of pairs. Newcombe discusses a few options on how to divide the files for comparison. He shows that while the most common method is using the alphabetic surnames and first given names, it is not necessarily the most efficient method due to the high level of errors inherent in them. He suggests that using the Russel Soundex codem, which is a phonetic coding, has strong advantages [Newcombe et al. 1959; Newcombe and Kennedy 1962; Newcombe 1967]. In his example of 114,000 marriage records, he shows that using a division based off the pair of surname codes for grooms and brides keeps the number of wasted comparisons of incoming birth records as low as 0.6 per incoming record [Newcombe 1967]. Newcombe also mentions that using more than one level of subdivision further reduces the number of wasted comparisons. The price of blocking records is an increase in the number of failures to bring potentially linkable records together and a reduction in the number of fields available for the comparison of potentially linkable records.

To optimize the matching step of record linkage, Newcombe notes that you must do more than simply count the number of items, or fields, in a pair of records that agree or disagree. This simple approach would essentially give each field equal weight; thereby ignoring the fact that some fields have more discriminating power than others [Newcombe and Kennedy 1962]. To incorporate the inherent discriminating power of fields, Newcombe suggests that the weights can be computed as a ratio of two frequencies. These frequencies are the number of agreements of a particular field in record pairs that represent the same individual and the number of agreements in

record pairs that do not represent the same individual [Newcombe 1967]. To obtain weights that can be added across the fields in a pair Newcombe takes the logarithm to the base two of the agreement ratio. The weights are then given as binit weights [Newcombe et al. 1959]. Using this structure will produce positive weights for fields that match and negative weights for fields that do not match. Summing across all the compared fields will give one number that can be compared to threshold values to determine if the pair of records is a match or not. Newcombe’s example uses threshold values of -10 and 10. This implies that the sums of weights over 10 are linked and those that sum below -10 are not linked, while those in between are left for further clerical investigation [Newcombe and Kennedy 1962]. Therefore, his linkage rule takes the sum of the weights and determines which region it lies in order to determine if he should link, not link, or leave the records undetermined.

Newcombe’s writings are very general in terms of the details of record linkage. In the late 1960s, however, there were several publications that focused on the theory behind record linkage. Some of those publications include Nathan [1967], Tepping [1968], Du Bois [1969], and Fellegi and Sunter [1969]. Nathan’s approach deals with linking a new single record with a master list or file [Nathan 1967]. Tepping addresses optimizing the decision rule so that the expected total cost associated with the possible actions taken is minimized [Tepping 1968]. The focus of Du Bois is on deriving probabilities using binomial modeling of match, no match indicators for record fields [Du Bois 1969]. While each of these methods has important results, the method that seemed to gain the most attention is the one presented by Fellegi and Sunter. Their method is a formalization of the decision rules and weights that much of the later work is based on.

Fellegi and Sunter’s theory parallels that of classical hypothesis testing and leads to linkage rules that are similar to Newcombe’s [Fellegi and Sunter 1969]. To begin, they formally define a comparison vector between two records, A and B, as a set of codes denoting the agreement status of the compared fields. All of the possible realizations of the comparison vector are within the comparison space. Therefore, based on the observed value of the comparison vector, the probabilities of the two

records being linked, not linked, or left undetermined can be found given the linkage rule. Note that the sum of these probabilities is equal to one. As mentioned above, there are two types of errors: linking records that are not a true match and failing to link records that are a true match. Fellegi and Sunter denote the probabilities of these errors as μ and λ respectively. One of the most important findings of Fellegi and Sunter is that given the probabilities of the two errors μ and λ , their linkage rule is optimal [Fellegi and Sunter 1969]. Optimal, in this case, is defined as when the number of record pairs that are left neither linked nor unlinked is minimized.

Rather than spending much as time on procedures that could help optimize the searching step of record linkage as Newcombe did, Fellegi and Sunter spent most of their writing on the procedures of computing weights used for the comparison step in which the linking decision is made. They mention only briefly the practical importance of blocking due to the great amount of comparisons necessary for even medium-sized files. Fellegi and Sunter discuss two methods for computing weights [Fellegi and Sunter 1969]. The first assumes prior knowledge of the distribution of values for each field compared in the file populations. It also assumes that the probabilities of errors in the record-taking process are known. The second method uses information from the data in the files to estimate conditional probabilities used in the computation of the field weights. This method depends strongly on an independence assumption made between the fields.

The independence assumption begins by considering the comparison vector a random variable and defining the conditional distributions of obtaining a particular comparison vector given the records' true matching state. The assumption is that the comparison outcomes for each field are "mutually statistically independent with respect to each of the conditional distributions" [Fellegi and Sunter 1969]. Fellegi and Sunter define a field weight as the log ratio of the conditional probabilities, given the match status, of the comparison outcomes. Using the independence assumption allows the weights to be summed across the fields. It is this sum of weights, or Score, that is used as the test statistic in the linkage rule.

One application of probabilistic record linkage that uses the independence

assumption described by Fellegi and Sunter was developed by the Family History Department of The Church of Jesus Christ of Latter-day Saints [White 1997]. It uses Bayes's Rule and conditional probabilities based on frequencies from the data. The conditional probabilities desired in this method are the probabilities of two fields being the same given that the record pair does or does not represent the same person, written $P(M|S)$ and $P(M|D)$. Using Bayes' Rule these conditionals can be written as $P(M|S) = \frac{P(S|M)P(M)}{P(S)}$ and $P(M|D) = \frac{P(D|M)P(M)}{P(D)}$. The two conditional probabilities, $P(S|M)$ and $P(D|M)$, can be found from the frequency counts of fields that are the same or different in a training data set. This training data set is a group of record pairs that have been compared by hand and determined as representing the same individual. The probability of matches, $P(M)$, is simply the proportion of record pairs that represent the same person. Finally, the probabilities $P(S)$ and $P(D)$ can be estimated by counting the number of times the fields are the same or different in a sample of randomly paired records. This method has been applied in previous projects in the Department of Statistics at Brigham Young University [Price 2000; Yamagata 2001; Francis 2004; Jensen 2004].

The method presented by Fellegi and Sunter also requires careful use of frequencies from the data without the training set. They show that the weights can be computed directly from the data. However, they also state that closed form solutions from their method can only be obtained if there are three fields being compared and a conditional independence assumption is made. With more variables, the Expectation-Maximization, or EM, algorithm can be applied to obtain estimates of the parameters needed in weight computation [Winkler 1993].

After presenting their method, Fellegi and Sunter give two additional cautions regarding the method. The first caution is that the amount of sampling variability may be large if the number of records in the two files being compared is not sufficiently large. The second caution is another reminder that these estimates are based on the independence assumption mentioned above. However, Fellegi and Sunter state that if the amount of identifying information available in the two files is sufficient enough to carry out the record linkage in the first place, they believe that the operation is

robust against departures from this assumption [Fellegi and Sunter 1969].

2.2 EM Algorithm

As mentioned previously, the Fellegi and Sunter method can only give closed solutions when there are only three fields being compared. Generally this is not the case, therefore, the Expectation-Maximization, or EM, algorithm is a good way to estimate the parameters needed in weight computation. The EM algorithm is a general iterative algorithm for maximum likelihood estimation in incomplete data problems [Little and Rubin 1987]. Essentially, this algorithm is a formalization of an ad-hoc idea for handling missing data:

- 1) Replace missing values with estimated values
- 2) Estimate parameters
- 3) Re-estimate values for the missing data with the new parameters
- 4) Repeat until convergence

The EM algorithm is usually discussed in terms of an “E step” and “M step”. In the E step, the conditional expectation of the “missing data” given the observed data and current parameters is found. The key idea of EM is that “missing data” are not the missing observations but functions, such as sufficient statistics, of the observations that appear in the complete data log likelihood. The M step performs the maximum likelihood estimation of the parameters as if there were no missing values, which implies that it is no different than normal maximum likelihood methods [Little and Rubin 1987].

2.3 Modern Applications

As mentioned before, the weights model presented by Fellegi and Sunter is what has been the prevailing model in record linkage. This was shown by Jaro in 1989 when he used the EM algorithm to estimate the two parameters needed for weight computation. Recall that those two parameters are the two conditional

probabilities of obtaining field agreement, assuming that the record pair represents the same person or that the pair does not represent the same person [Jaro 1989]. To use the EM algorithm, Jaro points out that all record pairs are distributed according to a finite mixture of probability models with three unknown parameters: the conditional probability of fields matching given that the record pair represents the same person, the conditional probability of fields matching given the record pair does not represent the same person, and the proportion of record pairs that represent the same person in all record pairs. Mixture models are common in model-based clustering where the main interest is to classify the observed data into different clusters or subpopulations [Pawitan 2001]. In this case, theoretically, the complete data is the field agreement status and an indicator specifying whether the records are of the same person or if the records are not of the same person. Therefore, in practice, the missing data is the indicator and the observed data is the field agreement status.

The E step in Jaro’s application is used to estimate the indicator values. Since these values are unknown, the estimates are interpreted as the probability of being in each population given the conditional probabilities of the field agreement status. This is done using Bayes’ Rule using the current estimates of the probabilities of being in each population, the conditional probability of field agreement given that the record pair represents the same person, and the conditional probability of field agreement given that the record pair does not represent the same person. The M step then updates the values of these three probabilities. Jaro states that the algorithm is not particularly sensitive to starting values, but the initial guess that the conditional probability of field agreement, given that the record pair represents the same person, must be larger than the conditional probability of field agreement given that the record pair does not represent the same person [Jaro 1989].

The EM algorithm used by Jaro can estimate the three parameters mentioned above, but he indicates that there is another way to estimate the conditional probability of agreement given that the record pair does not represent the same person. This is shown based on the cardinality of the set of record pairs that represents the same person, compared to the set of pairs that does not represent the same person.

If two files are of equal size, F , then the set of pairs that truly should be linked would be pF , where p is the proportion of pairs that represents the same person. In comparison, the set of pairs that should not be linked is much larger, $F^2 - pF$. Using this fact, Jaro suggests that you can just ignore the contribution from the set of pairs that represents the same person and estimate the conditional probability with the frequencies of chance agreement of the i th field [Jaro 1989].

2.4 Threshold Values

Up to this point most of the cited literature has focused on the matching and weight computation methods for record linkage. However, once this is done, linkage decisions of link, don't link, or leave for clerical review must still be made to finish the record linkage. This is typically done by setting threshold values to create three regions in order for that weights falling into each region to be classified according to the rule for that region. Errors of this method are false linkages where records are linked when they should not be and missed links where records that should be linked are not linked.

Fellegi and Sunter address this problem by sampling the various configurations of field agreement, computing the weights for the sampled configurations, and computing estimates of error rates using conditional probabilities and sampling probabilities [Fellegi and Sunter 1969]. In 1995, Belin and Rubin show that this method is rather optimistic in its reported error rates [Belin and Rubin 1995]. They suggest a method that uses transformations to normalize the weights for true and false linkages in order for that the distribution of weights to be viewed as a mixture of two transformed normal populations. They use a training sample to estimate global parameters that normalize the weight distributions and the parameter that gives the ratio of variances between the two distributions on the transformed scale. With these parameters they fit a mixture of transformed normal distributions to the observed weights to obtain maximum likelihood estimates and standard errors of the component means, variances and mixing proportions. For each possible threshold value, error rates and standard errors of those rates are computed so that thresholds can be

chosen at levels that give acceptable error rates.

Chapter 3

Methodology

3.1 Introduction

The objective of this project is to show that the EM algorithm presented by Jaro and a multinomial extension of it are useful in record linkage. In order to show this, field weights will be computed for two files of records using the EM algorithm. The first file of records are Church and County Records from Perquimans County, North Carolina. The second is a census index file of heads-of-household records from the 1910 and 1920 Censuses for Ascension Parish of Louisiana. The field weights for these files have previously been computed as a log ratio of frequencies and serve as comparisons [Price 2000; Jensen 2004]. This project requires the following steps. First, the records are blocked according to surname and gender. Second, comparisons are made between the records in a block and the agreement status of the fields is determined. Third, using the methods of the EM algorithm described by Jaro, the conditional probabilities needed to compute the weights for each field are computed. Then, conditional probabilities given the record pair does not represent the same individual are computed with the EM algorithm directly from the data, as suggested by Jaro. Finally, the computed conditional probabilities are compared with those obtained previously by hand to see how well the EM algorithm performs.

3.2 Blocking

The church records of Perquimans County are in one file with potentially more than one record representing each individual. For this file, blocking is done in SAS

by sorting the data by surname and gender, then records are paired in Proc SQL only where the surnames and genders match [SAS Institute Inc. 2006]. If a surname is missing in a record, that record is not blocked and therefore is not used in the project. After blocking there were 59,552 record pairs.

The records from Louisiana are from 1910 and 1920 and the goal is to obtain weights and threshold values from the data that can be used by a search program. These weights and threshold values are determined by comparing 1910 and 1920 records. This means that before blocking is done, two files need to be created, one with 1910 records and the other with 1920 records. The blocking is then done by pairing records with the same last name and gender using SAS and Proc SQL [SAS Institute Inc. 2006]. The Louisiana data produced 88,471 record pairs after blocking was complete.

3.3 Pair Comparisons

The paired comparisons of the fields in each file are done in SAS, but each comparison is done in a different way. The fields in the Perquimans County data are:

- 1) First name
- 2) Second (middle) name
- 3) Birth day
- 4) Birth month
- 5) Birth year
- 6) Birth town
- 7) Birth county
- 8) Birth state
- 9) Death day
- 10) Death year

- 11) Death month
- 12) Death town
- 13) Death county
- 14) Death state

Since these records have previously been matched by hand there were group indicators in the original data to show which records represent the same person. Using this information, an extra field was created to signify pairs that were linked in previous projects. For each block, the fields are compared between records and the agreement status is recorded as 0 for “Different”, 1 for “Same”, and 3 if one or both records have a missing entry for the field.

The comparisons in the Louisiana file are more complicated. Rather than having a simple binomial response for matching, each field has multiple levels of agreement. This means that for each field there is an agreement vector that will indicate how the field matched. For example, if there are three levels (i.e. “the same,” “nearly the same,” and “different”) and the fields match, then the vector is of length three with 1 in the first element and 0 in the other two elements. The fields in this file are Given (First) Name, Age, Race, and Birthplace.

Given name has six levels of agreement: exact match, nickname match, first three letters match, first letter match, last three letters match, and not match. A nickname match means that the name of one record is a nickname of the name in the second record. Nicknames were previously compiled in a table and the program uses that table as its reference [Francis 2004]. The agreement levels for each field are ordered by discriminating power and are mutually exclusive. For example, a pair of records with names of “Jim” and “James” are classified as having a nickname match only. They are not considered for a first letter match since a nickname match has more discriminating power in linking records.

The other fields in the file also have multiple levels but not as many. Age is considered an exact match if the difference between the two records recorded ages

is between 7 and 13 years. Differences of 6 or 14 years are considered “Close” and differences less than 6 or greater than 14 are classified “Different” [Jensen 2004]. The race field has three levels of agreement as well: “Same”, “Close” and “Different”. If one record has “B” for Black and another has “M” for Mulatto, those records are classified “Close”. Finally, birthplace has only two levels, “Same” and “Different”, but there is a lookup table, similar to the nickname table, for finding exact matches. For example, a pair of records with “Prussia” in one record and “Germany” in the other are considered an exact match.

3.4 EM Algorithm

The EM algorithm is used to estimate the values necessary to compute the weights of each field. Recall that the weights are the ratio of two conditional probabilities. The full set of record pairs, \mathbf{F} , is the union of two sets, \mathbf{M} and \mathbf{U} . The set \mathbf{M} contains the record pairs that represent the same individual and set \mathbf{U} is the set of pairs that do not represent the same person. The conditional probability of obtaining the outcome i when comparing field ν assuming the record pair represents the same person, or is in set \mathbf{M} , is denoted as $m_{\nu i}$. Similarly, the conditional probability of obtaining outcome i when comparing field ν assuming the record pair is not the same individual, or is in set \mathbf{U} , is $u_{\nu i}$. With multiple agreement levels in each field, an indicator vector, $\mathbf{y}_{r\nu}$, is needed to denote the agreement status for the r^{th} record pair and the ν^{th} field. To sum across the c_ν agreement levels the log of the ratio is taken. Therefore, the weight of the ν^{th} field is written formally as

$$w_\nu = \sum_{i=1}^{c_\nu} [y_{r\nu i} \log(m_i)] - \sum_{i=1}^{c_\nu} [y_{r\nu i} \log(u_i)]$$

The complete data is $\{\mathbf{y}_r, \mathbf{g}\}$, where $\mathbf{y}_r = (\mathbf{y}_{r1}, \mathbf{y}_{r2}, \dots, \mathbf{y}_{r\nu})$ and $\mathbf{y}_{r\nu}$ is the observed vector of agreement indicators of the ν^{th} field for the r^{th} record pair. The vector \mathbf{g} is the unknown indicator vector, (g_{rm}, g_{ru}) , where $g_{rm} = 1$ if the record pair, r , represents the same person, set \mathbf{M} , and $g_{ru} = 1$ if the record pair, r , does not represent the same person, set \mathbf{U} . The estimates of g_{rm} and g_{ru} are the conditional

probabilities of being in set **M** or **U** given the observed data for the r^{th} record pair. The value \hat{p} is the estimated proportion of record pairs in set **M**. Applying Bayes' Rule, the estimates for the r^{th} record pair are:

$$\hat{g}_{rm} = \frac{\hat{p} \prod_{j=1}^{\nu} \prod_{i=1}^{c_{\nu}} m_{ij}^{y_{rij}}}{\hat{p} \prod_{j=1}^{\nu} \prod_{i=1}^{c_{\nu}} m_{ij}^{y_{rij}} + (1 - \hat{p}) \prod_{j=1}^{\nu} \prod_{i=1}^{c_{\nu}} u_{ij}^{y_{rij}}}$$

and

$$\hat{g}_{ru} = \frac{(1 - \hat{p}) \prod_{j=1}^{\nu} \prod_{i=1}^{c_{\nu}} u_{ij}^{y_{rij}}}{\hat{p} \prod_{j=1}^{\nu} \prod_{i=1}^{c_{\nu}} m_{ij}^{y_{rij}} + (1 - \hat{p}) \prod_{j=1}^{\nu} \prod_{i=1}^{c_{\nu}} u_{ij}^{y_{rij}}}$$

The E step of the EM algorithm is used to get the estimates of g_{rm} and g_{ru} above.

If two records are paired together and one or the other has a missing entry for a field a decision must be made as to how to code the match types for that field. The first option is to code it as “Different” which would have no affect on the equations above. The second is to make the y_{rij} elements missing. This in turn would cause the elements of $\hat{\mathbf{m}}$ and $\hat{\mathbf{u}}$ to be missing for the c_{ν} match types of that field. Therefore, when estimating g_{rm} and g_{ru} the products shown above are only across those elements that have real values. The third option is to replace the missing elements of \mathbf{y} with the current field estimates from the \mathbf{u} vector. This allows the equations to work as normal.

With the estimated conditional probabilities of being in set **M** or **U**, given the data, the M step of the EM algorithm is used to update the conditional probabilities used in the field weights computation and the estimated proportion \hat{p} . The update of the conditional probability of obtaining the observed agreement status, i , for the ν^{th} field, given the record is obtained from set **M** is:

$$\hat{m}_{\nu i} = \frac{\sum_{i=1}^R [\hat{g}_{rm} y_{rvi}]}{\sum_{i=1}^R [\hat{g}_{rm}]}$$

where R is the total number of record pairs. The update for the conditional probability of obtaining the observed agreement status, i , for the ν^{th} field, given the record is obtained from set \mathbf{U} , is similar:

$$\hat{u}_{\nu i} = \frac{\sum_{i=1}^R [\hat{g}_{ru} y_{r\nu i}]}{\sum_{i=1}^R [\hat{u}_{rm}]}$$

The update of the estimate for the proportion of record pairs in set \mathbf{M} is:

$$\hat{p} = \frac{\sum_{i=1}^R \hat{g}_{rm}}{R}$$

In this project, convergence in the EM algorithm means that the sum of squared change of the estimates \hat{m} and \hat{u} is less than 0.0000001.

3.5 Alternate Method for Computing $\hat{\mathbf{u}}$

As shown above, the EM algorithm is used to get estimates of the two conditional probabilities $\hat{m}_{\nu i}$ and $\hat{u}_{\nu i}$; however using Jaro's method $\hat{\mathbf{u}}$ can also be estimated with frequencies taken from a random sample of the records pairs. This is achieved with the Louisiana data using SAS data steps to randomly select a specified number of records from 1910 and an equal number of records from 1920 and then pairing the randomly selected records. The number of sampled records is based on the desired accuracy of $\hat{\mathbf{u}}$. With these randomly paired records, the same field comparisons described above are made and the number of chance agreements for each field agreement type are counted. The resulting estimate for each $\hat{u}_{\nu i}$ is the ratio of chance agreements and number of comparisons made.

3.6 Missing Data

The church records of Perquimans County contain many missing values in the various fields. There is some concern that this will affect the performance of the EM

algorithm even though it is used as a way of handling missing data. Recall that the missing data for this application of the EM algorithm are the indicators g_{rm} and g_{ru} and not the recorded values in the fields. Another reason this is such a concern with the Perquimans County records is because the number of missing values is very high. The first name field only has 1.41% missing but every other field has more than 60% missing with five fields close to 99% missing. Therefore, a simulation with known values of m_{vi} and u_{vi} would be helpful in considering the effect of missing entries in the fields.

The simulation uses the EM algorithm already programmed with new data that is generated with known values of m_{vi} and u_{vi} . For the simulation, five fields are used with missing data ranging from 1% to 95%. The EM algorithm is also run with the simulated data before creating missing values to compare the estimated probabilities with and without missing values. Table 3.1 shows the known values of m and u for each field that are used to generate the data and the percent of missing values created.

m	u	Percent Missing
0.75	0.25	95%
0.80	0.20	90%
0.85	0.15	60%
0.90	0.10	15%
0.95	0.05	1%

Table 3.1: Simulation Parameters

Probabilistic Record Linkage Search Form	
Surname:	ANDRESKI
Given Name:	FRANK
Age:	49
Birthplace:	GERM
County:	HURON
State:	MI
Sex:	MALE
Race:	White
Census Year:	1920
Exact Match Search	
PRL Search	

Figure 3.1: Prototype Family History Search Form

3.7 Weights and Threshold Values

Once the estimated conditional probabilities are found for each field, the weights are computed. For each field the weight is computed as:

$$w_{\nu} = \sum_{i=1}^{c_{\nu}} [y_{ri} \log m_i] - \sum_{i=1}^{c_{\nu}} [y_{ri} \log u_i]$$

The sum of the field weights is then referred to as the record pair’s “Score”. The weights and Score could be particularly valuable in search forms used by family history researchers such as the prototype form shown in Figure 3.1 [Lawson 2006]. With such a form, a family history researcher could enter information known about a person into the given fields. This information is then compared against a database of records and the weight for each field computed. Using the Score of the comparison, records with higher probabilities of representing the searched person would then be returned. To determine if records should be returned, the Score of the comparison is compared to threshold values.

Threshold values are governed by the probability of the two errors associated

with record linkage. The first error probability, μ , is the probability of linking records that do not represent the same individual. The second, λ , is the probability of failing to link records that represent the same person. Scores less than the threshold value T_λ are not linked and scores greater than T_μ are linked. Scores that fall between these two values are left for further investigation.

The methods for computing threshold values described above are used to minimize both errors. However, in this application, the error of failing to link records that represent the same individual is a more severe error. Therefore, the threshold values should be such that λ is made as low as possible, keeping μ at a reasonable level. To compute the threshold values, the scores for each permutation of the agreement status vectors are computed first. Then those scores are ordered. Finally, the probability of each score is found using the estimated conditional probabilities of field agreement assuming the records represent the same individual. The threshold values are then chosen.

Chapter 4

Perquimans County Results

The goal of this project was to develop a program using the EM algorithm to enable probabilistic record linkage without having to use a training subset of record pairs. The nature of the datasets that were available for this project lent themselves to the development of two programs, one for the Perquimans County church and county records that used a binomial method and one for the Ascension Parish of Louisiana census that used a multinomial method. Also, since the Perquimans data was used first, some of the results from that work affected the approach with the Louisiana data. For that reason the following results are presented first for the Perquimans County data and then for the Louisiana data. Since much of this project involved programming there is time devoted to code explanation as well.

4.1 Pairing and Coding


Since weights for this data had already been computed by hand, a program had been previously written in SAS to pair each record [Price 2000]. However, for this project, there was some desire to redo the code so that it was more efficient and adaptable to other datasets. This section will address the new code in SAS and the procedures used to pair records. The procedures used to code the match status of the fields in the paired records will also be addressed.

The data was originally in Excel spreadsheets but was read into SAS with standard import procedures. Once the data was in SAS, it was sorted by last name (variable named SURNAME) and gender (variable named SEX) to prepare for blocking. A new variable, RECORD, which was used to number the records with the

same last name was then added to the data. Using a built in command `first.` and `if/else` condition statements, the first record with a new last name would trigger a `RECORD` to be set to one. Then the variable would increment until a record with a new last name began. The actual pairing of records was done simultaneously with the coding of the match status of the fields which is described next.

```
create table matches as
select n1.SURNAME, n1.name1, n2.name1 as name2,
      n1.record, n2.record as record2
from tmp as n1, tmp as n2
where n1.SURNAME eq n2.SURNAME
      and n1.record < n2.record;
```

RECORD	SURNAME	NAME
1	Albertsen	Ann
2	Albertsen	Anne
3	Albertsen	Hannah
1	Albertson	Peter
2	Albertson	Esau
1	Ames	Sarah
2	Ames	George



SURNAME	NAME1	NAME2	RECORD	RECORD2
Albertsen	Ann	Anne	1	2
Albertsen	Ann	Hannah	1	3
Albertsen	Anne	Hannah	2	3
Albertson	Peter	Esau	1	2
Ames	Sarah	George	1	2

Figure 4.1: Example of SQL pairing

The pairing of records and coding the match status of the fields was done simultaneously within the Proc SQL procedure. SQL itself is not original to SAS but is a database querying language that SAS has incorporated into itself. Typically, SAS executes its code one data row at a time but SQL makes it possible to use all data rows in a dataset at once. This becomes helpful when comparing field entries in all records. In a traditional datastep, pairing records with the same last name and gender would be complicated, requiring many loops and several steps. In SQL you can create a table, or data set, with conditional statements, like `where`, and it will search all records automatically. Figure 4.1 shows an example of creating a table,

MATCHES, from a file of records, *TMP*, that pairs records with the same last name. Notice that the *RECORD* field is used to prevent doubling of pairs.

The technique of pairing described above combines all records with the same last name and gender. For each pairing, the entries in the 14 fields of this dataset were compared and coded according to their match type. Recall that fields that are the same are coded as 1, those that are different are coded as 0, and fields with a missing entry are coded as 3. This was done simply with `case` statements that work like `if/else/then` statements. Programming with Proc SQL means that coding the match type of the fields is done for the entire dataset at once. Also, the pairing of records is done internally, which means that a data set with all pairs is never actually created. Therefore, the program went directly from the original data set to a data set with the blocks' last name, the record numbers of each record in the pair, and 14 fields with numerical codes representing the match status of the fields in the pair.

4.2 Results from the EM Algorithm

The EM algorithm was programmed in Proc IML in SAS. IML is a procedure in SAS that allows matrix coding. The computations described in the Methodology section were done in this program. After the computations were complete, a final data set was created containing the number of iterations, the final value of epsilon, the weights, and the estimates of \mathbf{m} and \mathbf{u} for each field.

The goal of this project was to show that the estimates from the EM algorithm were comparable to the estimates obtained by hand. Table 4.1 shows the estimates of \mathbf{m} and \mathbf{u} obtained from the EM algorithm. Unfortunately, these are not as close as desired to those obtained by hand, also shown in Table 4.1. After looking over the program it seemed there were no critical errors. Therefore, it was determined that a simulation with known vectors \mathbf{m} and \mathbf{u} would help in understanding the EM algorithm.

Field	EM Estimates		Clerical Estimates	
	$\hat{\mathbf{m}}$	$\hat{\mathbf{u}}$	$\hat{\mathbf{m}}$	$\hat{\mathbf{u}}$
First Name	0.80734	0.07509	0.90143	0.03407
Second Name	0.43772	0.02379	0.73332	0.02220
Birth Day	0.95804	0.03284	0.92068	0.03216
Birth Month	1	0.08228	0.95314	0.07821
Birth Year	0.99999	0.00567	0.96913	0.00568
Birth Town	0.74494	0.69954	1	0.65799
Birth County	0.95384	0.95004	1	0.91029
Birth State	0.96712	0.96495	1	0.99514
Death Day	0.25504	0.03424	0.96893	0.02197
Death Month	0.29287	0.08329	0.96872	0.10142
Death Year	0.25262	0.01392	0.96931	0.00476
Death Town	0.08683	0.08715	.	0
Death County	0.85581	0.85587	1	0.92105
Death State	0.85581	0.85587	1	1

Table 4.1: EM and Clerical Estimates of \mathbf{m} and \mathbf{u} for the Perquimans County Records

4.3 Simulation

In effort to understand why the EM algorithm estimates were not closer to those obtained by hand, a simulation with known vectors \mathbf{m} and \mathbf{u} was conducted. A data set was simulated with five fields, each with a unique value of m and u , shown in Table 4.2. The table shows parameters of m and u that sum to one but this is not an indication of a required constraint. A second data set was then created by randomly replacing values in the first data set with missing values. The rate at which the values were made missing varied by field to represent the missing rates in the real data. The proportion of record pairs that were simulated to represent the same individual was also set to represent the real data. The real data had a total of 59,552 pairs with 1,111 pairs where both records represented the same individual. This means that 1.87% of the record pairs were from the set \mathbf{M} . The simulation had a total of 1350 pairs with 25 pairs, or 1.85%, representing the set \mathbf{M} .

One of the main questions the simulation intended to answer is how missing

				EM Estimates			
Field	Simulated Missing Rate	True Values		No Missing		With Missing	
		\mathbf{m}	\mathbf{u}	$\hat{\mathbf{m}}$	$\hat{\mathbf{u}}$	$\hat{\mathbf{m}}$	$\hat{\mathbf{u}}$
1	95%	0.75	0.25	0.86230	0.25731	0.31635	0.29952
2	90%	0.80	0.20	0.88128	0.21623	0.21341	0.23444
3	60%	0.85	0.15	0.95353	0.14552	0.38870	0.16043
4	15%	0.90	0.10	0.84194	0.10834	0.93702	0.08935
5	5%	0.95	0.05	1	0.05492	0.99963	0.05489

Table 4.2: Parameters and Estimates From the Perquimans Simulation

entries in the records affected the ability of the EM algorithm to estimate the parameters. Note that for fields in record pairs that were coded as missing, the algorithm replaces the \mathbf{y} values with the current field estimates from \mathbf{u} . This method is discussed further in the next chapter. The real data had missing values in each field with rates varying from 1.41% to 99.96%. Also, 13 of the 14 fields had more than 60% missing and 9 had more than 90%. To represent these conditions, missing values were created in the data at the rates shown in Table 4.2. The simulation results, also shown in Table 4.2, show that when there are no missing entries the estimates of the \mathbf{m} values are within 0.12 of the parameters and the estimates of the \mathbf{u} elements are within 0.02. When the data with missing values is run through the EM algorithm, the estimates get much worse. For this reason, it was determined that the Perquimans County data is not a good data set for the EM algorithm due to the high rate of missing entries in the records.

Chapter 5

Louisiana Results

After finding that missing values have a large affect on the ability of the EM algorithm to estimate the elements of \mathbf{m} and \mathbf{u} , it was decided that the algorithm should be tried on a new dataset. The census index file of heads-of-household records from the 1910 and 1920 Censuses for Ascension Parish of Louisiana provides a good alternative because it has already been analyzed and it contains less missing values in the records. This chapter presents the results obtained from analysis of the Louisiana data and the associated simulations.

5.1 Pairing and Coding

A few aspects of this data made its preparation more difficult. The first was that the fields' matching outcomes had more than two possibilities. The best example of this was the given name field, which had six possible outcomes. Second, some of the outcomes, like matching nicknames, required the program to search reference tables to determine the match status. Lastly, the data required more data scrubbing, such as removing question marks or N/A from field entries. Much of the coding for this section was done by Nathan Orien under supervision through an undergraduate mentoring program.

The first thing done in this code was the data scrubbing. Then the records from 1910 were put into one data set and the 1920 records put in another. Those two files were then paired by last name and gender with Proc SQL to make one dataset. This data set has a duplicate of each field, except the blocking fields. The Perquimans County data did not need to be split and paired like this because the program was

looking for duplicate records without regard to when the record was collected. There is no reason to look for duplicates within a census year in this data, so the records were split by year to prevent comparisons within the same year.

After pairing the records, most field match outcomes were coded with conditional `if/then/else` statements that assigned a 0 or 1 to a new field which represented the outcome. Coding nicknames and birthplaces, however, are more complicated since they require a reference table to use for comparison. Each reference table had previously been compiled and were brought into SAS as separate data sets [Francis 2004]. The reference tables have the nickname or alternate birthplace in their first field. The other fields in the tables contain names or birthplaces that are potential matches to the first entry. In regard to nicknames, the given name from both years were compared to the first entry in the nickname reference table using Proc SQL. If the name in one year matched the nickname, then the other year was compared to the given names associated with that nickname in the reference table. If it matched one of the names, then that pair was marked as having a nickname match. A similar procedure was used with birthplaces.

The last step in the program is to make sure that two special conditions are met. The first condition is that there can only be one match type for a given field. For example, if a pair has one given name of “Robert” and the other has “Albert”, it is marked as a match on the last three letters only. The other condition that is addressed explains how to deal with missing values. The different ways this is dealt with are discussed later in the chapter.

In addition to pairing records and coding the field match types, a section of code is written to take a specified number of samples from all record pairs. For each year, a random observation is chosen and put into a sample data set. Then, the two sample data sets are merged by their sample order. The set of randomly paired records is then put into the same SAS macro to code the field match outcomes, which are used as an alternate method for estimating \mathbf{u} .

5.2 Results from the EM Algorithm

The coding of the EM algorithm changed very little from the Perquimans County code. There were some improvements made so that it utilized more matrix algebra which in turn made it more efficient. Also, the alternative method of counting the frequency of match status outcomes from a sample of record pairs to estimate \mathbf{u} was added. These alternate estimates were added to the output data set.

Field	EM Estimates		Alternate	Clerical Estimates	
	$\hat{\mathbf{m}}$	$\hat{\mathbf{u}}$	$\hat{\mathbf{u}}$	$\hat{\mathbf{m}}$	$\hat{\mathbf{u}}$
Age Same	0.79841	0.2041	0.1991	0.79252	0.03128
Age Close	0.04817	0.0573	0.05896	0.17177	0.01418
Age Different	0.15342	0.7386	0.74194	0.03571	0.95454
Birthplace Same	0.94956	0.88722	0.80261	0.97227	0.80397
Birthplace Different	0.05044	0.11278	0.19739	0.02773	0.19603
Name Same	0.50556	0.00958	0.00344	0.64262	0.00470
Name Nickname	0.00635	0.00197	0.00101	0.08389	0.00241
Name 1st 3 Letters	0.15493	0.00466	0.00344	0.13423	0.00328
Name 1st Letter	0.27185	0.06203	0.06148	0.13591	0.05791
Name Last 3 Letters	0.01808	0.00451	0.00121	0.000	0.00291
Name Different	0.04324	0.91726	0.92942	0.00336	0.92880
Race Same	0.93203	0.7137	0.42826	0.94128	0.43693
Race Close	0.06243	0.10357	0.07765	0.03523	0.07282
Race Different	0.00554	0.18273	0.4941	0.02349	0.49025

Table 5.1: EM and Clerical Estimates of \mathbf{m} and \mathbf{u} for the Louisiana Data. The inputs into the EM Algorithm are: Epsilon: 0.00001 and Starting Values for \mathbf{m} (starting with Age Same and moving down): 0.85 0.14 0.01 0.99 0.01 0.719 0.05 0.13 0.1 0.0005 0.0005 0.98 0.015 0.005. The alternate \mathbf{u} estimates come from a sample of 5000 record pairs.

The estimates from the EM algorithm for the Louisiana records gave better results than those obtained with the Perquimans County records. They were closer to the values previously obtained with clerical methods [Francis 2004]. The EM and clerical estimates are shown in Table 5.1. Unfortunately, there are a couple of estimates that are not as good as the others. First, the estimates of m for given

name were off, especially those for given name being the same and different. These differences motivated simulations that are discussed later in the chapter. The second set of poor estimates that brought concern were the estimates of u for race. Upon looking into this difference, it was found that the frequencies of the match types for race were different depending on whether the pairs were put in blocks or not. This is reasonable because different races tend to have different surnames. The estimates obtained hand and from the alternative method of estimating \mathbf{u} are close to the frequencies from the unblocked data. The estimates from the EM algorithm were close to the frequencies from the blocked data. For use in ancestral searches, it is more appropriate to use the estimates of \mathbf{u} from the alternate method because the records are not blocked in the search.

5.3 Simulation

The estimates shown in Table 5.1 are the best estimates obtained for this data. The initial run of the EM Algorithm gave similar results with the exception of the estimates of the m values for the given name being exactly the same and the name being completely different. In runs with different initial estimates and epsilon convergence criteria, the estimates of these two values were off by over 0.40. Many of the simulations were run in an effort to understand the inaccuracy of these estimates. To get the estimates shown in Table 5.1, several variables in the algorithm had to be carefully selected. The variables chosen were based on results from simulations that showed certain combinations produced better estimates. The variables that were examined in the simulations included the total number of record pairs, the number of missing entries in the records, the starting values for \mathbf{m} and \mathbf{u} , and the epsilon value in the algorithm. It is not believed that there are any significant interactions between these variables, so the simulations were done by changing one variable at a time, while holding the others constant.

5.3.1 Total Number of Record Pairs

There are a total of 88,471 blocked record pairs in the Louisiana data. Of those pairs, 591, or 0.67%, are from the set **M**. As a result, the question was raised concerning whether the number of record pairs and/or the percentage of the total number of pairs that were in the **M** set affected how well the EM algorithm estimates the parameters. A simulation was set up to obtain 10-15 estimates from a fixed number of record pairs and **M** proportion. Figures 5.1, 5.2, and 5.3 show boxplots of estimates for **m** from simulations where 0.5% were from set **M** and the total number of record pairs ranged from 10,050 to 502,500.

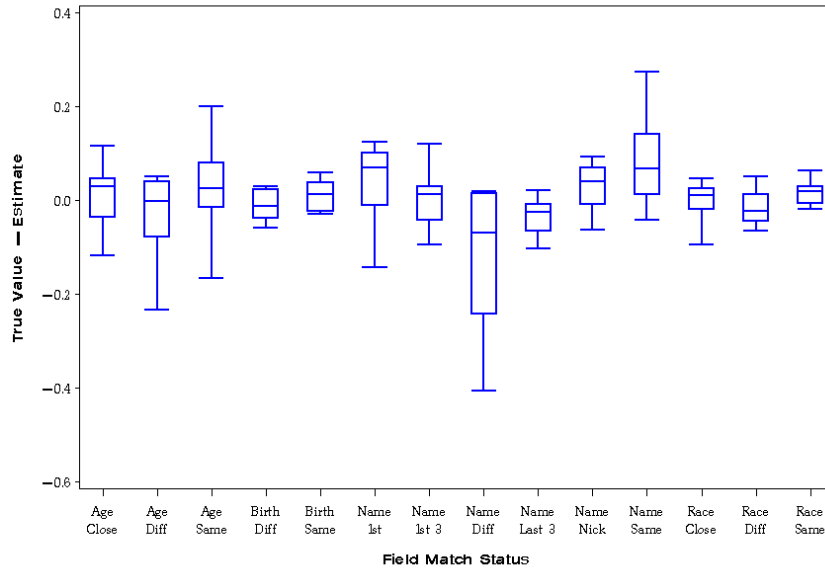


Figure 5.1: Boxplots of Ten Simulation Estimates of **m**. Total Record Pairs: 10,050. No Missing Entries. Epsilon: 0.000001. Starting Values are listed in Table 5.2

The most obvious changes in the estimates are the reductions in variation as the total number of record pairs increase. Also, noticeable are the long tailed boxplots for NAME SAME and NAME DIFF in Figure 5.1 and the outlying estimates for the same match types in Figure 5.2. These figures show that occasionally simulations

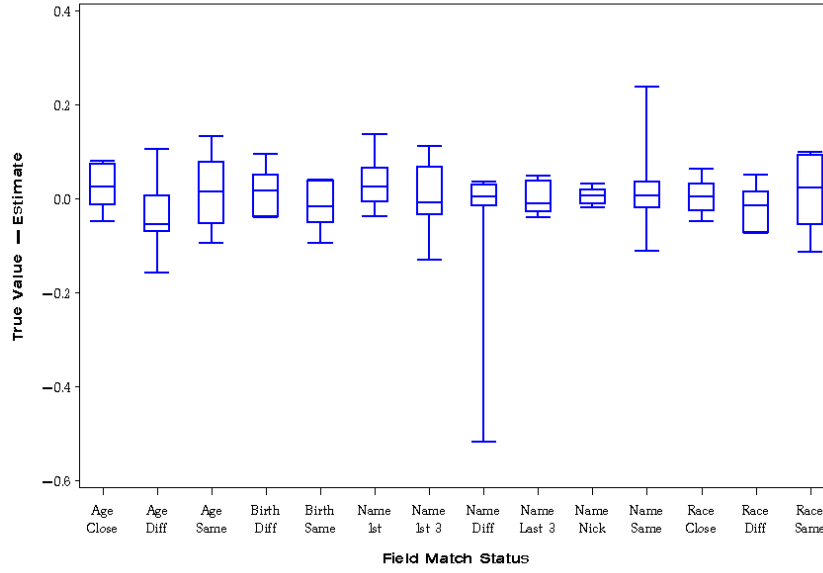


Figure 5.2: Boxplots of Ten Simulation Estimates of \mathbf{m} . Total Record Pairs: 100,500. No Missing Entries. Epsilon: 0.000001. Starting Values are listed in Table 5.2

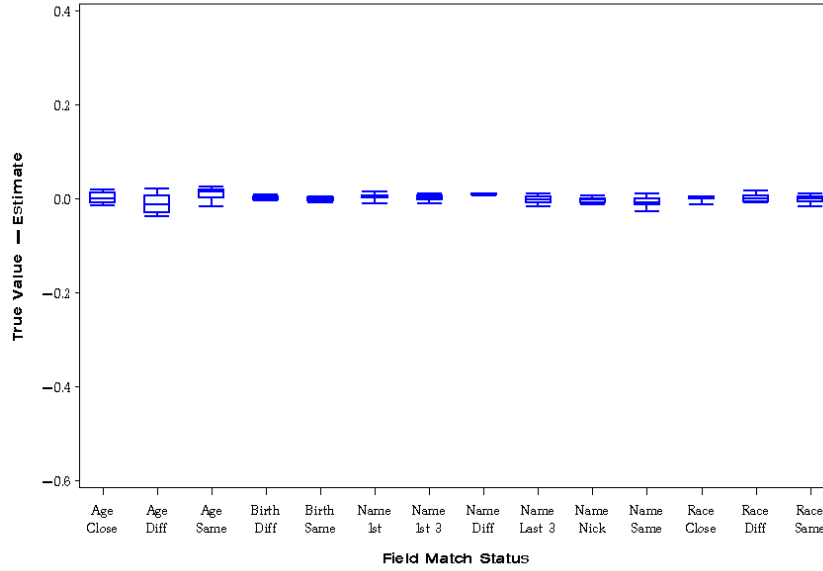


Figure 5.3: Boxplots of Ten Simulation Estimates of \mathbf{m} . Total Record Pairs: 502,500. No Missing Entries. Epsilon: 0.000001. Starting Values are listed in Table 5.2

with a total number of record pairs of around 100,000 or less give bad estimates for the two given name match types, “Same” and “Different”. This never occurred with the larger data sets. Additional simulations were run with 1,005,000. The variation in estimates got even smaller and no simulations gave bad or biased results for given name being the same or different. Simulations with a small number of record pairs were also run with the proportion of pairs in the **M** set at 10.5% and there were no cases where the estimates further than 0.1 from the true estimates. It is concluded that the EM algorithm works well if the number of record pairs is around 500,000 or more, regardless of the proportion of pairs in set **M**. However, if the number of records is around 100,000 or less, a higher proportion of record pairs in the **M** set is required.

5.3.2 Missing Values

One of the first things considered with the Louisiana data was how to handle missing record entries because it was determined that missing entries were the reason the EM algorithm failed with the Perquimans County data. Three solutions were tried and incorporated into simulations. The first method of dealing with missing record entries is to code the field with the “Different” match type. The second method is to code the all field match types as missing. This means that pairs where a field entry is missing in one or the other records are not included in the estimation of that field’s elements of **m** or **u**. For example, if race is missing for a record pair then it does not contribute to the estimate of the race elements but it does contribute to the others. The third method is to code the field match types like the second method but in the EM algorithm the missing values are replaced with the current estimates of the corresponding **u** values. If birthplace is missing in a record pair, then each time the EM algorithm starts over, the missing values for the two birthplace match types are replaced with the two current estimates from **u**.

Results from these different methods show very little change in the variation of the estimates but they do affect their bias. Coding missing record entries as different caused the estimates of being the same to be underestimated and estimates of being

different overestimated. When coding the field as missing, method two, the estimates for being different were more accurate but the estimates for being the same were still overestimated. This is easily seen in the estimates of \mathbf{m} for birthplace and age. The third method improved on the second by making the estimates for being the same more accurate. Figures 5.4, 5.5 and 5.6 show boxplots of the difference between the true \mathbf{u} values and their estimates from ten simulated data sets using each of the three methods. This shows that using method three gives the best estimates.

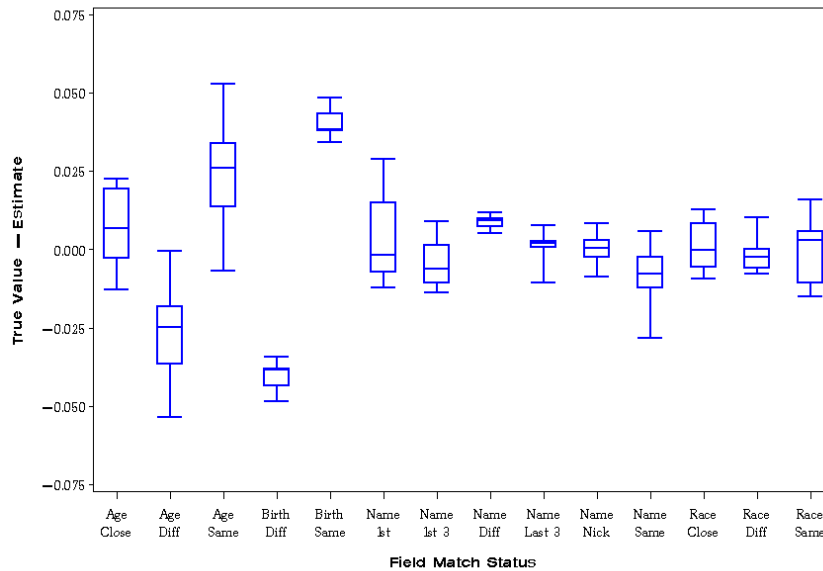


Figure 5.4: Boxplots of Simulation Estimates for Pairs with Missing Entries Coded as Different. Total Record Pairs: 502,500. Missing Entry Rates: Age 2%, Birthplace 4%, Name 1%, Race 0.1%. Epsilon: 0.000001. Starting Values are listed in Table 5.2

5.3.3 Starting Values

The biased estimates of \mathbf{m} for given name being the same or different were also believed to be connected to the starting values for \mathbf{m} and \mathbf{u} . Since it was determined that estimating \mathbf{u} with the alternative method was best, most of the simulations focused on changes in the starting values of \mathbf{m} . Starting values were investigated

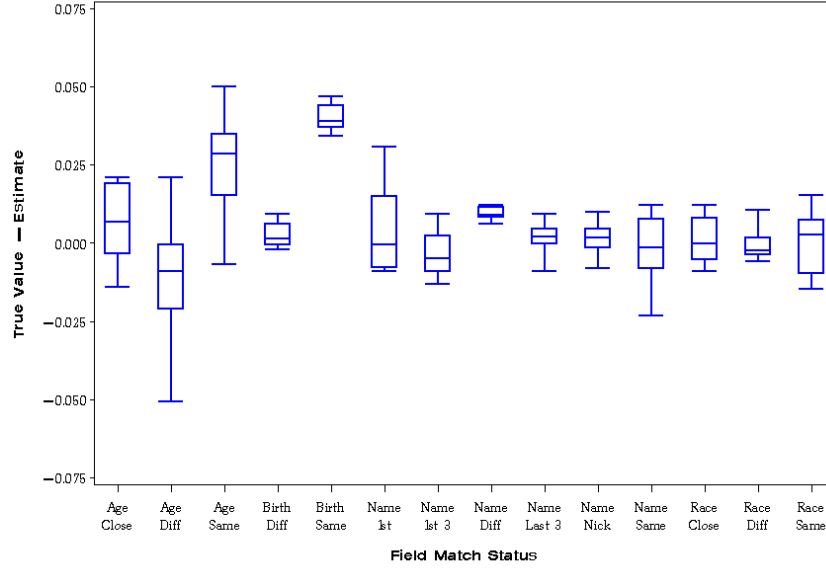


Figure 5.5: Boxplots of Simulation Estimates for Pairs with Missing Entries Coded as Missing. Total Record Pairs: 502,500. Missing Entry Rates: Age 2%, Birthplace 4%, Name 1%, Race 0.1%. Epsilon: 0.000001. Starting Values are listed in Table 5.2

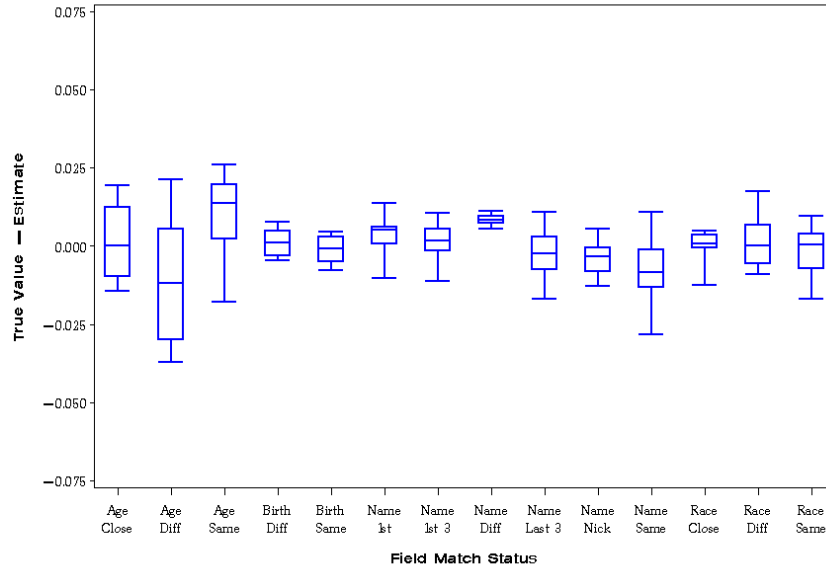


Figure 5.6: Boxplots of Simulation Estimates for Pairs with Missing Entries Substituted with Estimates from \mathbf{u} . Total Record Pairs: 502,500. Missing Entry Rates: Age 2%, Birthplace 4%, Name 1%, Race 0.1%. Epsilon: 0.000001. Starting Values are listed in Table 5.2

in two different ways. The first method kept the same starting values but changed the parameters used to simulate the data. Table 5.2 shows the starting values and the three sets of parameters used to simulate the data. The boxplots from these simulations do not show any large differences in the variation of the estimates.

Field	Starting Values		Parameter Set					
			1		2		3	
	m	u	m	u	m	u	m	u
Age Same	0.8	0.03	0.7	.03	0.9	0.1	0.5	0.2
Age Close	0.18	0.01	0.25	.01	0.08	0.05	0.3	0.1
Age Different	0.02	0.96	0.05	.98	0.02	0.85	0.2	0.7
Birthplace Same	0.97	0.8	0.97	.85	0.9	0.75	0.7	0.6
Birthplace Different	0.03	0.2	0.03	.15	0.1	0.25	0.3	0.4
Name Same	0.688	0.005	0.6	.003	0.5	0.01	0.4	0.01
Name Nickname	0.08	0.005	0.1	.003	0.13	0.01	0.05	0.015
Name 1st 3 Letters	0.13	0.005	0.12	.005	0.15	0.02	0.2	0.02
Name 1st Letter	0.1	0.05	0.14	.049	0.17	0.1	0.25	0.07
Name Last 3 Letters	0.001	0.005	0.02	.04	0.025	0.1	0.05	0.1
Name Different	0.001	0.93	0.02	.9	0.025	0.76	0.05	0.785
Race Same	0.94	0.44	0.9	.4	0.8	0.3	0.7	0.2
Race Close	0.03	0.07	0.05	.1	0.1	0.1	0.2	0.3
Race Different	0.03	0.49	0.05	.5	0.1	0.7	0.1	0.5

Table 5.2: Starting Values and Simulation Parameters

The second method for investigating starting values, however, showed a change in the accuracy of the estimates based on the relative distance and location of the starting value from the true value. Using data set Finale614 and Finale615, which were simulated using parameter set 1, the starting values for the given name being the same and different were varied while the other given name starting values were set at the parameter values. As a result, it appeared that with a true parameter that is small, having a starting value less than the true value, gives better estimates. Also, if the true parameter was large, starting values greater than the true parameter gave better results. This trend appeared with both data sets. Table 5.3 shows this trend

for epsilon values of 0.00001.

	Same	Diff	Same	Diff	Same	Diff	Same	Diff
True Value	0.6	0.02	0.6	0.02	0.6	0.02	0.6	0.02
Start Value	0.619	0.001	0.61	0.01	0.59	0.03	0.581	0.39
Finale614	0.5771	0.0107	0.5306	0.0853	0.4767	0.1757	0.4752	0.1815
Finale615	0.5623	0.0173	0.4583	0.1998	0.4295	0.2392	0.4117	0.2670

Table 5.3: Estimates for Name Same and Name Different for Changing Starting Values. Datasets Finale614 and Finale615 have 100,500 record pairs with missing entry rates: Age 2%, Birthplace 4%, Name 1%, Race 0.1%, and epsilon: 0.000001.

5.3.4 Epsilon Values

The EM algorithm was set to continue until the sum of squared differences between the new and old estimates of \mathbf{m} and \mathbf{u} went below a set epsilon value. Initially, the algorithm used an epsilon of 0.000001 but a simulation was set up to see how changing this value would effect the estimates. The simulation used epsilon values of 0.0001, 0.00001, and 0.000001 on data sets that were simulated independently. In general, it was found that a smaller epsilon value results in estimates closer to the actual value. However, it was only with the smallest epsilon values that the estimates for given name being the same or completely different were severely wrong.

After seeing that the worst estimates only occurred with the smallest epsilon values, two specific simulated data sets were set up to output estimates at each epsilon value. This way, the data would not change with the epsilon value. The two data sets chosen for this grid had been initially simulated with 100,500 record pairs with 0.5% in set \mathbf{M} , for a EM run with an epsilon value of 0.000001. These two data sets were chosen because they had been simulated with the same parameters but had a large difference in their estimates for first name being the same and different. The first data set, Finale614, gave an estimate for given name being completely different that was no more than 0.01 away from the true value. The second data set, Finale615,

gave an estimate for the same field outcome that was more than 0.2 over the true value.

Parameters:		Finale614		Finale615	
		Same	Different	Same	Different
		0.6	0.02	0.6	0.02
Estimates	0.0001	0.5658	0.0097	0.5522	0.0131
at:	0.00001	0.5771	0.0107	0.5623	0.0173
	0.000001	0.5814	0.0128	0.4290	0.2478

Table 5.4: Estimates for Name Same and Name Different for Changing Epsilon Values. Datasets Finale614 and Finale615 have 100,500 record pairs with missing entry rates: Age 2%, Birthplace 4%, Name 1%, Race 0.1% and starting values listed in Table 5.2

Table 5.4 shows that decreasing epsilon to 0.00001 improved the estimates for both data sets. However, moving epsilon passed 0.00001 severely worsened the estimates with the Finale615 records but continued to improve them with the Finale614 data set. The biased estimates with Finale615 are an example of bad estimates that occasionally occur when the number of record pairs are around 100,000 or smaller. Therefore, if there are around 100,000 record pairs, it may be better to use a larger epsilon to help avoid cases where estimates are extremely different from the true values. However, if there are around 500,000 record pairs, smaller epsilon values improve estimates as they did with the data set Finale614.

Chapter 6

Conclusions and Future Work

The need to link records has become increasingly necessary in today's world due to the vast amounts of information being collected. For those interested in ancestral research, it is especially necessary to continue in their hobby. However, too much time is required for linking records with traditional hand-linking methods. This project focused on showing that an application of the EM algorithm could help eliminate some of the hand work required to link records by estimating the probabilities needed for the weights used in probabilistic record linkage. It was found that the EM algorithm can be useful when estimating probabilities. Also, much was learned about the EM algorithm's sensitivities to inputs and data characteristics. Specifically, the ability of the EM algorithm to accurately estimate the probability of two records representing the same person, given that the fields in those records match, is sensitive to the total number of record pairs, the number of missing entries in the records, the starting values used, and the epsilon value in the algorithm.

Fellegi and Sunter warned that sampling variation is high when the number of records pairs is not sufficiently large. This warning was confirmed by the simulations conducted in this project. The number of record pairs for the real data used in this project was less than 100,000. This turned out to be a small amount of records. It was also observed that when a field has many match types, the EM algorithm can give estimates that are far from the true values with data sets around 100,000 or less. In addition to the total number of record pairs used, it was found that the proportion of pairs that represent the same individual affects the algorithm's performance. If the proportion is as high as 10.5% it appears to cancel the negative affects of having a

smaller set of record pairs. For this reason, it is recommended to use the EM algorithm for data sets with more than 100,000 record pairs or that have a high proportion of pairs representing the same person.

For records in a data set with a high percentage of missing entries in the fields, it was found that the algorithm did not perform well. Noticable differences in the performance of the estimates started as low as 4% but drastic declines in performance were not observed until the missing percentage was over 60%. This is believed to be the reason that the EM algorithm did not perform well on the Perquimans County church and county records. It is likely that the results found with missing entries are also related to the results found with small sets of record pairs since a large percentage of missing entries can significantly reduce the useful number of record pairs in data sets. In regard to how to address missing entries, it was also found that the best way to code the match status of fields with missing entries is to replace the missing field values with the current estimates from the \mathbf{u} vector. This helps to avoid overestimating the probabilities associated with “Different” match types and underestimating probabilities associated with “Same” match types.

One of the inputs into the EM algorithm is starting values for the vectors \mathbf{m} and \mathbf{u} . It was found that the distance and direction of these starting values relative to the actual values had some impact on the performance of the algorithm’s estimates. If the true parameter was large, the best results came when the starting values were larger. The opposite was found for small parameter values. If the starting values were smaller than the true values, the estimates improved. Also, the estimates became better as the distance between the true value and the starting value increased.

The last result from this project dealt with the epsilon value, which is the value that determines when the algorithm has converged. If at the end of an iteration the sum of squared differences between the new and old estimates of \mathbf{m} and \mathbf{u} was smaller than the epsilon value the algorithm would end. It was found that smaller values of epsilon resulted in better estimates from the algorithm. However, if the total number of records was around 100,000 or less, the smallest value of epsilon occasionally gave worse results. Therefore, it is recommended to use epsilon values

greater than 0.000001 for small data sets and an epsilon of 0.000001 for larger data sets.

6.1 Future Work

While the results of this project are useful in understanding this application of the EM algorithm, the performance with these data sets was not as successful as hoped. This is likely due to the small number of records pairs available and in the case of the Perquimans County data the number of missing entries in the data. In the future it would be more effective to apply this algorithm to a larger data set. With that data, it would also be useful to establish threshold values that could be used as cutoff values when returning records in an ancestral search.

More in-depth questions could also be addressed to improve the algorithm itself. First, the validity of the independence assumption made by this application should be investigated. Also, a method to assign weights to the fields used to block the data could be developed. Further investigation as to why the EM algorithm occasionally gave biased results when decreasing the epsilon value would also be useful. As an alternative to using the EM algorithm, Bayesian methods with priors on the estimated parameters may be investigated.

Bibliography

- Belin, T. R. and Rubin, D. B. (1995), “A Method for Calibrating False-Match Rates in Record Linkage,” *Journal of the American Statistical Association*, 90, 694–707.
- Du Bois, D. N. (1969), “A Solution to the Problem of LInking Multivariate Documents,” *Journal of the American Statistical Association*, 64, 163–174.
- Dunn, H. (1946), “Record Linkage,” *American Journal of Public Health*, 36, 163–174.
- Fellegi, I. P. and Sunter, A. B. (1969), “A Theory for Record Linkage,” *Journal of the American Statistical Association*, 64, 1183–1210.
- Francis, M. A. (2004), “Probabilistic Record Linkage of Census Data,” Master’s project, Brigham Young University.
- 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.
- Jensen, K. P. (2004), “Probabilistic Methodology for Record Linkage: Determining Robustness of Weights,” Master’s project, Brigham Young University.
- Lawson, J. (2006), “Find Them in the Census Records Using Automatic Record-Linkage Techniques,” To be published in *Genealogical Helper*.
- Little, R. J. and Rubin, D. B. (1987), *Statistical Analysis with Missing Data*, John Wiley & Sons, Inc., chap. 7.
- Nathan, G. (1967), “Outcome Probabilities for a Record Matching Process with Complete Invariant Information,” *Journal of the American Statistical Association*, 62, 454–469.

- Newcombe, H., Kennedy, J., Axford, S., and James, A. (1959), “Automatic Linkage of Vital Records,” *Science*, 130, 954–959.
- Newcombe, H. B. (1967), “Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family Histories,” *American Journal of Human Genetics*, 19, 13–37.
- Newcombe, H. B. and Kennedy, J. M. (1962), “Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information,” *Communications of the Association of Computing Machinery*, 5, 563–566.
- Pawitan, Y. (2001), *In All Likelihood: Statistical Modeling and Inferences Using Likelihood*, Oxford Science Publications, chap. 12.
- Price, B. L. (2000), “Probabilistic Methodology for Record Linkage: A Review of Theory and an Example,” Master’s project, Brigham Young University.
- SAS Institute Inc. (2006), *Base SAS 9.1.3 Procedures Guide, Second Edition, Volumes 1, 2, 3, and 4*, SAS Institute Inc., Cary, NC.
- Tepping, B. J. (1968), “A Model for Optimum Linkage of Records,” *Journal of the American Statistical Association*, 63, 1321–1332.
- White, D. (1997), “A Review of the Statistics of Record Linkage for Genealogical Research,” in *Record Linkage Techniques-1997: Proceedings of and International Workshop and Exposition*, pp. 362–373.
- Winkler, W. E. (1993), “Improved Decision Rules in the Fellegi-Sunter Model of Record Linkage,” Statistical research report series 93/12, U.S. Bureau of the Census, Washington DC.
- Yamagata, R. T. (2001), “Probabilistic Methodology for Genealogical Record Linkage: Increasing Classification Rates and Decreasing Unclassified Rates,” Master’s project, Brigham Young University.

Appendix A

SAS Code for Perquimans County Data Pairing and Coding

This SAS code is used to read in, pair, and code match status types for the Perquimans County church and county records.

```
options ls=120;
libname link 'C:/Documents and Settings/John/My Documents/School/
RecordLinkage/Quaker/Data/';

data link.fullldata;
infile 'C:/Documents and Settings/John/My Documents/School/RecordLinkage/
matchedgrouGendat2.csv' dsd dlm=',' firstobs=2;
input GRP MATCH RIN SURNAME :$15. NAME1 :$15. NAME2 :$15. FAMS FAMC SEX :$1. BDAY
BYEAR :$10. BMON :$4. BCONG :$15. BTOWN :$15. BCOUN :$10. BSTATE :$5. DDAY
DMON :$4. DYEAR :$10. DCONG :$5. DTOWN :$15. DCOUN :$15. DSTATE :$5. ROLE :$5.
EVENT :$5. EDAY EMON :$4. EYEAR :$10. ECONG :$5. ETOWN :$15. ECOUN :$15.
ESTATE :$6. GROUP;
run;

data link.nomiss;
set link.fullldata;
if SURNAME not in('','/') ;
run;

proc sort data=link.nomiss;
by SURNAME SEX;
run;

data link.tmp (drop=tmp);
set link.nomiss;
by SURNAME;
retain ;
if first.SURNAME then do;
    record = 1;
    tmp = SURNAME;
end;
if not(first.SURNAME) and (SURNAME = tmp) then record = record + 1;
run;

proc sql;
```

```

create table link.temp as
select *
from link.tmp;
create index myindex
  on link.temp(SURNAME, SEX);

create table link.matches as
select n1.SURNAME, n1.SEX, n1.record, n2.record as record2,
  case
    when n1.MATCH eq n2.MATCH
    then
      case
        when n1.MATCH eq . and n2.MATCH eq . then 3 else 1
      end
    else
      case
        when n1.MATCH eq . or n2.MATCH eq . then 3 else 0
      end
    end as MATCH_M,
  case
    when n1.NAME1 eq n2.NAME1
    then
      case
        when n1.NAME1 eq '' and n2.NAME1 eq '' then 3 else 1
      end
    else
      case
        when n1.NAME1 eq '' or n2.NAME1 eq '' then 3 else 0
      end
    end as NAME1_M,
  case
    when n1.NAME2 eq n2.NAME2
    then
      case
        when n1.NAME2 eq '' and n2.NAME2 eq '' then 3 else 1
      end
    else
      case
        when n1.NAME2 eq '' or n2.NAME2 eq '' then 3 else 0
      end
    end as NAME2_M,
  case
    when n1.BDAY eq n2.BDAY
    then
      case
        when n1.BDAY eq . and n2.BDAY eq . then 3 else 1
      end
    else
      case
        when n1.BDAY eq . or n2.BDAY eq . then 3 else 0
      end
    end as BDAY_M,
  case
    when n1.BYEAR eq n2.BYEAR

```

```

then
    case
        when n1.BYEAR eq '' and n2.BYEAR eq '' then 3 else 1
    end
else
    case
        when n1.BYEAR eq '' or n2.BYEAR eq '' then 3 else 0
    end
end as BYEAR_M,
case
    when n1.BMON eq n2.BMON
    then
        case
            when n1.BMON eq '' and n2.BMON eq '' then 3 else 1
        end
    else
        case
            when n1.BMON eq '' or n2.BMON eq '' then 3 else 0
        end
    end
end as BMON_M,
case
    when n1.BTOWN eq n2.BTOWN
    then
        case
            when n1.BTOWN eq '' and n2.BTOWN eq '' then 3 else 1
        end
    else
        case
            when n1.BTOWN eq '' or n2.BTOWN eq '' then 3 else 0
        end
    end
end as BTOWN_M,
case
    when n1.BCOUN eq n2.BCOUN
    then
        case
            when n1.BCOUN eq '' and n2.BCOUN eq '' then 3 else 1
        end
    else
        case
            when n1.BCOUN eq '' or n2.BCOUN eq '' then 3 else 0
        end
    end
end as BCOUN_M,
case
    when n1.BSTATE eq n2.BSTATE
    then
        case
            when n1.BSTATE eq '' and n2.BSTATE eq '' then 3 else 1
        end
    else
        case
            when n1.BSTATE eq '' or n2.BSTATE eq '' then 3 else 0
        end
    end
end as BSTATE_M,
case

```

```

when n1.DDAY eq n2.DDAY
then
  case
    when n1.DDAY eq . and n2.DDAY eq . then 3 else 1
  end
else
  case
    when n1.DDAY eq . or n2.DDAY eq . then 3 else 0
  end
end as DDAY_M,
case
  when n1.DYEAR eq n2.DYEAR
  then
    case
      when n1.DYEAR eq '' and n2.DYEAR eq '' then 3 else 1
    end
  else
    case
      when n1.DYEAR eq '' or n2.DYEAR eq '' then 3 else 0
    end
  end as DYEAR_M,
case
  when n1.DMON eq n2.DMON
  then
    case
      when n1.DMON eq '' and n2.DMON eq '' then 3 else 1
    end
  else
    case
      when n1.DMON eq '' or n2.DMON eq '' then 3 else 0
    end
  end as DMON_M,
case
  when n1.DTOWN eq n2.DTOWN
  then
    case
      when n1.DTOWN eq '' and n2.DTOWN eq '' then 3 else 1
    end
  else
    case
      when n1.DTOWN eq '' or n2.DTOWN eq '' then 3 else 0
    end
  end as DTOWN_M,
case
  when n1.DCOUN eq n2.DCOUN
  then
    case
      when n1.DCOUN eq '' and n2.DCOUN eq '' then 3 else 1
    end
  else
    case
      when n1.DCOUN eq '' or n2.DCOUN eq '' then 3 else 0
    end
  end as DCOUN_M,

```

```

case
  when n1.DSTATE eq n2.DSTATE
  then
    case
      when n1.DSTATE eq '' and n2.DSTATE eq '' then 3 else 1
    end
  else
    case
      when n1.DSTATE eq '' or n2.DSTATE eq '' then 3 else 0
    end
  end as DSTATE_M
from link.tmp as n1, link.tmp as n2
where n1.SURNAME eq n2.SURNAME
  and n1.SEX = n2.SEX
  and n1.record < n2.record;
quit;

```


Appendix B

SAS Code for Perquimans County EM Algorithm

This SAS code is the EM algorithm programmed for the Perquimans County church and county records.

```
libname link '.';

proc iml;
use link.matches;
read all var _num_ into mtch;

*number of pairs;
np = nrow(mtch);

*number of fields;
nf = ncol(mtch)-3;

*intitital estimates of m, u, and p;
mi = j(1,nf,0);
ui = j(1,nf,0);
mi = {0.93 0.93 0.93 0.93 0.93 0.93 0.98 0.98 0.98 0.93 0.93 0.93 0.93 0.98 0.98};
ui = {0.05 0.05 0.05 0.05 0.05 0.05 0.85 0.85 0.85 0.05 0.05 0.05 0.05 0.85 0.85};
pin = 0.05;

*Get gamma matrix and match unmatched identifiers from input cells;
gamma = mtch[1:np,4:nf+3];
gma = j(np,1,0);
gmu = j(np,1,0);

p1 = pin;
m1 = mi;
u1 = ui;
epi = 5;
test=0;

do while(epi>.0000001);
*do while(test<3);

*E step;
do j=1 to np by 1;
```

```

product1 = 1;
product2 = 1;
do i=1 to nf by 1;
    if gamma[j,i] ^= 0 & gamma[j,i] ^= 1 then gamma[j,i]=u1[i];
    product1 = product1 * (m1[i]**gamma[j,i])*(1-m1[i]**(1-gamma[j,i]));
    product2 = product2 * (u1[i]**gamma[j,i])*(1-u1[i]**(1-gamma[j,i]));
end;
gma[j] = p1*product1/(p1*product1+(1-p1)*product2);
gmu[j] = (1-p1)*product2/(p1*product1+(1-p1)*product2);
end;

*M step;
b1 = 0;
b2 = 0;
m1 = m1;
u1 = u1;
m1 = j(1,nf,0);
u1 = j(1,nf,0);

do j=1 to np by 1;
    do i=1 to nf by 1;
        m1[i] = m1[i] + gamma[j,i]*gma[j];
        u1[i] = u1[i] + gamma[j,i]*gmu[j];
    end;
    b1 = b1 + gma[j];
    b2 = b2 + gmu[j];
end;

m1 = m1/b1;
u1 = u1/b2;
p1 = b1/np;

epi = 0;
epi = (m1 - m2)*(m1 - m2)' + (u1 - u2)*(u1 - u2)';
test = test+1;

end;

*calculate weights;
wa = log(m1/u1);
wd = log((1-m1)/(1-u1));

*calculate scores;
score = j(np,1,0);
do j=1 to np by 1;
    do i=1 to nf by 1;
        if gamma[j,i] = 1 then score[j] = score[j] + wa[i];
        else score[j] = score[j] + wd[i];
    end;
end;

*create output;
EMiter = test||epi;
EMparm = wa' || wd' || m1' || u1';

```

```

create iter from EMiter;
append from EMiter;

create parm from EMparm;
append from EMparm;
quit;

data iter (rename=(col1=Iterations col2=Epsilon));
set iter;
data parm (rename=(col1=W_Same col2=W_Diff col3=M col4=U));
set parm;

data fields;
input Fields$ @@;
datalines;
F_Name M_Name BDay BYear BMon BTown BCoun BState
DDay DYear DMon DTown DCoun DState
run;

data parm;
merge iter fields parm;
run;

```


Appendix C

SAS Code for Data Generation with the Perquimans County Simulation

This SAS code is used to generate the simulated data for the simulation associated with the Perquimans County church and county records.

```
libname link 'C:\Documents and Settings\John\My Documents\School\
RecordLinkage\Quaker\Data';

data link.sim (drop= i j);
do i=1 to 25;
  d1=rand('uniform');d2=1;d3=1;
  f1=rand('binomial',.75,1);f2=rand('binomial',.80,1);
  f3=rand('binomial',.85,1);f4=rand('binomial',.90,1);
  f5=rand('binomial',.95,1);
  output;
  do j = 1 to 53;
    d1=rand('uniform');d2=2;d3=1;
    f1=rand('binomial',.25,1);f2=rand('binomial',.20,1);
    f3=rand('binomial',.15,1);f4=rand('binomial',.10,1);
    f5=rand('binomial',.05,1);
    output;
  end;
end;
run;

data link.miss;
set link.sim;
if rand('uniform') < .95 then f1 = 3;
if rand('uniform') < .9 then f2 = 3;
if rand('uniform') < .6 then f3 = 3;
if rand('uniform') < .15 then f4 = 3;
if rand('uniform') < .01 then f5 = 3;
run;
```


Appendix D

SAS Code for Perquimans County Simulation EM Algorithm

This SAS code is the EM algorithm programmed for the simulated data associated with the Perquimans County records. It is the same as the algorithm for the real data but has been modified for five fields and is repeated for the simulated record pairs with missing values.

```
libname link '.';

title 'Sim';
proc iml;
use link.sim;
read all var _num_ into mtch;

*number of pairs;
np = nrow(mtch);

*number of fields;
nf = ncol(mtch)-3;

*intitital estimates of m, u, and p;
mi = j(1,nf,0);
ui = j(1,nf,0);
do i=1 to nf by 1;
    mi[i] = .25;
    ui[i] = .15;
end;
pin = 0.05;

*Get gamma matrix and match unmatched identifiers from input cells;
gamma = mtch[1:np,4:nf+3];
gma = j(np,1,0);
gmu = j(np,1,0);

p1 = pin;
m1 = mi;
u1 = ui;
epi = 5;
test=0;
```



```

do while(epi>.0000001);
*do while(test<3);

*E step;
do j=1 to np by 1;
  product1 = 1;
  product2 = 1;
  do i=1 to nf by 1;
    if gamma[j,i] ^= 0 & gamma[j,i] ^= 1 then gamma[j,i]=u1[i];
    product1 = product1 * (m1[i]**gamma[j,i])*(1-m1[i]**(1-gamma[j,i]));
    product2 = product2 * (u1[i]**gamma[j,i])*(1-u1[i]**(1-gamma[j,i]));
  end;
  gma[j] = p1*product1/(p1*product1+(1-p1)*product2);
  gmu[j] = (1-p1)*product2/(p1*product1+(1-p1)*product2);
end;

*M step;
b1 = 0;
b2 = 0;
m2 = m1;
u2 = u1;
m1 = j(1,nf,0);
u1 = j(1,nf,0);
do j=1 to np by 1;
  do i=1 to nf by 1;
    m1[i] = m1[i] + gamma[j,i]*gma[j];
    u1[i] = u1[i] + gamma[j,i]*gmu[j];
  end;
  b1 = b1 + gma[j];
  b2 = b2 + gmu[j];
end;

m1 = m1/b1;
u1 = u1/b2;
p1 = b1/np;

epi = 0;
epi = (m1 - m2)*(m1 - m2)‘ + (u1 - u2)*(u1 - u2)‘;
test = test+1;

end;

*calculate weights;
wa = log(m1/u1);
wd = log((1-m1)/(1-u1));

*calculate scores;
score = j(np,1,0);
do j=1 to np by 1;
  do i=1 to nf by 1;
    if gamma[j,i] = 1 then score[j] = score[j] + wa[i];
    else score[j] = score[j] + wd[i];
  end;
end;
end;

```

```

*create output;
EMiter = test||epi;
EMparm = wa' || wd' || m1' || u1';

create iter from EMiter;
append from EMiter;

create parm from EMparm;
append from EMparm;
quit;

data iter (rename=(col1=Iterations col2=Epsilon));
set iter;
data parm (rename=(col1=W_Same col2=W_Diff col3=M col4=U));
set parm;
data fields;
input Fields$ @@;
datalines;
f1 f2 f3 f4 f5
run;

data parm;
merge iter fields parm;
run;

title 'Miss';
proc iml;
use link.miss;
read all var _num_ into mtch;

*number of pairs;
np = nrow(mtch);

*number of fields;
nf = ncol(mtch)-3;

*intitital estimates of m, u, and p;
mi = j(1,nf,0);
ui = j(1,nf,0);
do i=1 to nf by 1;
    mi[i] = .25;
    ui[i] = .15;
end;
pin = 0.05;

*Get gamma matrix and match unmatched identifiers from input cells;
gamma = mtch[1:np,4:nf+3];
gma = j(np,1,0);
gmu = j(np,1,0);

p1 = pin;
m1 = mi;
u1 = ui;

```

```

epi = 5;
test=0;

do while(epi>.0000001);
*do while(test<3);

*E step;
do j=1 to np by 1;
  product1 = 1;
  product2 = 1;
  do i=1 to nf by 1;
    if gamma[j,i] ^= 0 & gamma[j,i] ^= 1 then gamma[j,i]=u1[i];
    product1 = product1 * (m1[i]**gamma[j,i])*(1-m1[i]**(1-gamma[j,i]));
    product2 = product2 * (u1[i]**gamma[j,i])*(1-u1[i]**(1-gamma[j,i]));
  end;
  gma[j] = p1*product1/(p1*product1+(1-p1)*product2);
  gmu[j] = (1-p1)*product2/(p1*product1+(1-p1)*product2);
end;

*M step;
b1 = 0;
b2 = 0;
m1 = m1;
u1 = u1;
m1 = j(1,nf,0);
u1 = j(1,nf,0);
do j=1 to np by 1;
  do i=1 to nf by 1;
    m1[i] = m1[i] + gamma[j,i]*gma[j];
    u1[i] = u1[i] + gamma[j,i]*gmu[j];
  end;
  b1 = b1 + gma[j];
  b2 = b2 + gmu[j];
end;

m1 = m1/b1;
u1 = u1/b2;
p1 = b1/np;

epi = 0;
epi = (m1 - m2)*(m1 - m2)c + (u1 - u2)*(u1 - u2)c;
test = test+1;

do i=1 to nf by 1;
  if m1[i] = 1 then m1[i] = 1- (1/(2*np));
end;

end;

*calculate weights;
wa = log(m1/u1);
wd = log((1-m1)/(1-u1));

*calculate scores;

```

```

score = j(np,1,0);
do j=1 to np by 1;
  do i=1 to nf by 1;
    if gamma[j,i] = 1 then score[j] = score[j] + wa[i];
    else score[j] = score[j] + wd[i];
  end;
end;

*create output;
EMiter = test||epi;
EMparm = wa'||wd'||m1'||u1';

create iter from EMiter;
append from EMiter;

create parm from EMparm;
append from EMparm;
quit;

data iter (rename=(col1=Iterations col2=Epsilon));
set iter;
data parm (rename=(col1=W_Same col2=W_Diff col3=M col4=U));
set parm;
data fields;
input Fields$ @@;
datalines;
f1 f2 f3 f4 f5
run;

data parm;
merge iter fields parm;
run;

```


Appendix E

SAS Code for Louisiana Data Pairing and Coding

This SAS code is used to read in, pair, and code match status types for the census index file of heads-of-household records from the 1910 and 1920 Censuses for Ascension Parish of Louisiana.

*GENEALOGICAL RECORD LINKAGE PROJECT

OUTLINE OF STEPS:

- 1) Assign library and read-in data
- 2) Separate 1910 records from 1920 records and scrub/prepare data
- 3) Generate two random samples and merge results
- 4) Create table blocked by last name and sex
- 5) Explanation of comparison vectors
- 6) Create intermediate macros for gname and bplace
- 7) Use macro to create comparison vectors for both tables

*Assign library and read-in data;

%let disk = C:/Documents and Settings/John/My Documents/School/RecordLinkage/

LA Files/Missing;

libname records "&disk/";

options nodate formdlim = '_';

PROC IMPORT OUT= records.DATAIN

DATAFILE= "&disk/PrototypePRL.xls"

DBMS=EXCEL REPLACE;

SHEET="'Census Index Data\$'";

GETNAMES=YES;

MIXED=NO;

SCANTEXT=YES;

USEDATE=YES;

SCANTIME=YES;

RUN;

proc sql;

create table records.census as

select unqid, surname, gname, age, gender, race, bplace, state, county, locale,
matchid, year

from records.datain

order by unqid;

quit;

```

PROC IMPORT OUT= records.nicknames
      DATAFILE= "&disk/PrototypePRL.xls"
      DBMS=EXCEL REPLACE;
      SHEET="'Nicknames$'";
      GETNAMES=NO;
      MIXED=NO;
      SCANTEXT=YES;
      USEDATE=YES;
      SCANTIME=YES;
RUN;

```

```

data records.nicknames;
set records.nicknames;
f1 = upcase(f1);
f2 = upcase(f2);
f3 = upcase(f3);
f4 = upcase(f4);
f5 = upcase(f5);
f6 = upcase(f6);
f7 = upcase(f7);
f8 = upcase(f8);
f9 = upcase(f9);
f10 = upcase(f10);
f11 = upcase(f11);
f12 = upcase(f12);
f13 = upcase(f13);
f14 = upcase(f14);
f15 = upcase(f15);
f16 = upcase(f16);
f17 = upcase(f17);
run;

```

```

PROC IMPORT OUT= records.birthplaces
      DATAFILE= "&disk/PrototypePRL.xls"
      DBMS=EXCEL REPLACE;
      SHEET="'Birthplace Index$'";
      GETNAMES=NO;
      MIXED=NO;
      SCANTEXT=YES;
      USEDATE=YES;
      SCANTIME=YES;
RUN;

```

```

*Separate 1910 records from 1920 records and scrub/prepare data;
proc sql;
create table LA_only as
select * from records.census
where state = "LA";
quit;

proc sql;
alter table LA_only
modify bplace char(20) format = $20.;

```

```

alter table LA_only
modify race char(4) format = $4.;

alter table LA_only
modify gender char(4) format = $4.;

update LA_only
set gname = ' ',
where prxmatch('/\w*[\?]+\w*/',gname) or prxmatch("/(N\A)/",gname);
quit;

proc sql;
drop table LA_1910;
create table LA_1910 as
select * from LA_only
where year = 1910
order by unqid;

drop table LA_1920;
create table LA_1920 as
select * from LA_only
where year = 1920
order by unqid;
quit;

data LA_1910;
set LA_1910;
informat bplace $20. race $4. gender $4.;
if bplace = 'UNKN' then bplace = ' ';
if bplace = '----' then bplace = ' ';
if race = ' ' then race = ' ';
if race = '-' then race = ' ';
if gender = '-' then gender = ' ';
run;

data rename1920;
set LA_1920;
rename unqid = unqid2;
rename surname = surname2;
rename gname = gname2;
rename age = age2;
rename gender = gender2;
rename race = race2;
rename bplace = bplace2;
rename state = state2;
rename county = county2;
rename locale = locale2;
rename matchid = matchid2;
rename year = year2;
run;

data rename1920;
set rename1920;
informat bplace2 $20. race2 $4. gender2 $4.;

```



```

if bplace2 = 'UNKN' then bplace2 = ' ';
if bplace2 = '----' then bplace2 = ' ';
if race2 = ' ' then race2 = ' ';
if race2 = '-' then race2 = ' ';
if gender2 = '-' then gender2 = ' ';
if matchid2 = 'none' then matchid2 = 'none2';
run;

data LA_1910;
set LA_1910;
if bplace = 'BELG' then bplace = 'BELGIUM';
if bplace = 'CANA' then bplace = 'CANADA';
if bplace = 'CHIN' then bplace = 'CHINA';
if bplace = 'ENGL' then bplace = 'ENGLAND';
if bplace = 'FRAN' then bplace = 'FRANCE';
if bplace = 'GERM' then bplace = 'GERMANY';
if bplace = 'IREL' then bplace = 'IRELAND';
if bplace = 'ITAL' then bplace = 'ITALY';
if bplace = 'POLA' then bplace = 'POLAND';
if bplace = 'PORT' then bplace = 'PORTUGAL';
if bplace = 'RUSS' then bplace = 'RUSSIA';
if bplace = 'SCOT' then bplace = 'SCOTLAND';
if bplace = 'SPAI' then bplace = 'SPAIN';
if bplace = 'SWIT' then bplace = 'SWITZERLAND';
if bplace = 'SYRI' then bplace = 'SYRIA';
if bplace = 'TURK' then bplace = 'TURKEY';
run;

data rename1920;
set rename1920;
if bplace2 = 'ALSA' then bplace2 = 'ALSACE';
if bplace2 = 'AUST' then bplace2 = 'AUSTRIA';
if bplace2 = 'BAVA' then bplace2 = 'BAVARIA';
if bplace2 = 'CANA' then bplace2 = 'CANADA';
if bplace2 = 'ENGL' then bplace2 = 'ENGLAND';
if bplace2 = 'FRAN' then bplace2 = 'FRANCE';
if bplace2 = 'GERM' then bplace2 = 'GERMANY';
if bplace2 = 'HOLL' then bplace2 = 'HOLLAND';
if bplace2 = 'HOND' then bplace2 = 'HONDURAS';
if bplace2 = 'IREL' then bplace2 = 'IRELAND';
if bplace2 = 'ITAL' then bplace2 = 'ITALY';
if bplace2 = 'RUSS' then bplace2 = 'RUSSIA';
if bplace2 = 'SPAI' then bplace2 = 'SPAIN';
if bplace2 = 'SWIT' then bplace2 = 'SWITZERLAND';
if bplace2 = 'SYRI' then bplace2 = 'SYRIA';
run;

*Generate two random samples and merge results;
*1910 data;
data sample1 (drop=i sampsize);
    sampsize=5000;
    do i=1 to sampsize;
        pickit=ceil(ranuni(0)*totobs);
        set LA_1910 point=pickit

```

```

        nobs=totobs;
        output;
    end;
    stop;
run;
*1920 data;
data sample2 (drop=i sampsize);
    sampsize=5000;
    do i=1 to sampsize;
        pickit=ceil(ranuni(0)*totobs);
        set rename1920 point=pickit
            nobs=totobs;
        output;
    end;
    stop;
run;

data seq1;
set sample1;
num = _n_;
run;

data seq2;
set sample2;
num = _n_;
run;

data samples;
merge seq1 seq2;
by num;
run;

*Create table blocked by last name and sex;
proc sql;
drop table block;
create table block as
select a.unqid, a.surname, a.gender, a.gname, a.age, a.race, a.bplace, a.state,
    a.county, a.locale, a.matchid, a.year, b.unqid2, b.gname2, b.age2, b.race2,
    b.bplace2, b.state2, b.county2, b.locale2, b.matchid2, b.year2
from LA_1910 as a, rename1920 as b
where a.surname = b.surname2 and a.gender = b.gender2
order by a.unqid;
quit;

data blockmiss;
set block;
num = _n_;
run;

*Explanation of comparison vectors

MATCHID
ym = 1 if unqid (1910) = matchid2 (1920) or unqid2 = matchid, otherwise ym = 0

```

RACE

```
(yr1,yr2,yr3) = (1,0,0) if race = race2  
(yr1,yr2,yr3) = (0,1,0) if race is close to race2  
(yr1,yr2,yr3) = (0,0,1) if race is not equal to race2
```

AGE

```
(ya1,ya2,ya3) = (1,0,0) if age = age2  
(ya1,ya2,ya3) = (0,1,0) if age is close to age2  
(ya1,ya2,ya3) = (0,0,1) if age is not equal to age2
```

GNAME

```
(yg1,yg2,yg3,yg4,yg5,yg6) = (1,0,0,0,0,0) if gname = gname2  
(yg1,yg2,yg3,yg4,yg5,yg6) = (0,1,0,0,0,0) if gname is nickname of gname2,  
                                         or vice versa  
(yg1,yg2,yg3,yg4,yg5,yg6) = (0,0,1,0,0,0) if first three letters of gname = first  
                                         three letters of gname2  
(yg1,yg2,yg3,yg4,yg5,yg6) = (0,0,0,1,0,0) if first letter of gname = first letter  
                                         of gname2  
(yg1,yg2,yg3,yg4,yg5,yg6) = (0,0,0,0,1,0) if last three letters of gname = last  
                                         three letters of gname2  
(yg1,yg2,yg3,yg4,yg5,yg6) = (0,0,0,0,0,1) if gname is not equal to gname2
```

BPLACE

```
(yb1,yb2) = (1,0) if bplaces are the same or close (based on lookup table)  
(yb1,yb2) = (0,1) if bplaces are different;
```

*Create intermediate macros for gname and bplace;

*Nickname macro definitions;

```
%macro closename(dataset2,var1,var2,var3,var4);
```

```
proc sql;
```

```
drop table &dataset2;
```

```
create table &dataset2 as
```

```
select * from &dataset1 as a, records.nicknames as b
```

```
where a.&var1 = b.&var2 and a.&var3 = b.&var4;
```

```
quit;
```

```
%mend closename;
```

```
%macro closejoin(dummy,dataset3,dataset4);
```

```
proc sql;
```

```
create table &dummy as
```

```
select * from &dataset3
```

```
outer union corr
```

```
select * from &dataset4
```

```
quit;
```

```
%mend closejoin;
```

*Bplace macro definition;

```
%macro close_bplace(dataset5,var1,var2,var3,var4);
```

```
proc sql;
```

```
drop table &dataset5;
```

```
create table &dataset5 as
```

```
select * from &dataset1 as a, records.birthplaces as b
```

```
where a.&var1 = b.&var2 and a.&var3 = b.&var4;
```

```
quit;
```

```

%mend close_bplace;

*Use macro to create comparison vectors for both tables;
%macro vectors(dataset1,summary,text);
proc sql;
alter table &dataset1
    add ym num
    add yr1 num add yr2 num add yr3 num
    add ya1 num add ya2 num add ya3 num
    add yg1 num add yg2 num add yg3 num add yg4 num add yg5 num add yg6 num
    add yb1 num add yb2 num
    add ctm num
    add ctr num
    add cta num
    add ctg num
    add ctb num;
quit;

data &dataset1;
set &dataset1;
if unqid = matchid2 then ym = 1;
if unqid ne matchid2 then ym = 0;
    yr1=0;yr2=0;yr3=0;
if race = ' ' or race2 = ' ' then do;
    yr1=.;yr2=.;yr3=.;
    /*yr1=0;yr2=0;yr3=1;*/ /**USE WHEN CODING MISSING AS DIFFERENT**/
end;
else do;
    if race = race2 then yr1 = 1;
    if race ne race2 then yr3 = 1;
    if race = 'B' and race2 = 'M' then yr2 = 1;
    if race = 'M' and race2 = 'B' then yr2 = 1;
    if yr2 = 1 then yr3 = 0;
end;
if age = . or age2 = . then do;
    ya1=.;ya2=.;ya3=.;
    /*ya1=0;ya2=0;ya3=1;*/ /**USE WHEN CODING MISSING AS DIFFERENT**/
end;
else do;
    if 7 <= abs(age2-age) <= 13 then ya1 = 1;
    else ya1 = 0;
    if abs(age2 - age) = 6 or abs(age2 - age) = 14 then ya2 = 1;
    else ya2 = 0;
    if abs(age2 - age) < 6 or abs(age2 - age) > 14 then ya3 = 1;
    else ya3 = 0;
end;
run;

data &dataset1;
set &dataset1;
yg1=0;yg2=0;yg3=0;yg4=0;yg5=0;yg6=0;
if gname = ' ' or gname2 = ' ' then do;
    yg1=.;yg2=.;yg3=.;yg4=.;yg5=.;yg6=.;
    /*yg1=0;yg2=0;yg3=0;yg4=0;yg5=0;yg6=1;*/ /**USE WHEN CODING MISSING AS**/

```

```

                                /**DIFFERENT**/

end;
else do;
if gname = gname2 then do;
    yg1 = 1;
end;
else do;
    if length(gname) > 3 and length(gname2) > 3 and substr(gname,1,3) =
        substr(gname2,1,3) then do;
        yg3 = 1;
    end;
    else do;
        yg3 = 0;
        if substr(gname,1,1) = substr(gname2,1,1) then do;
            yg4 = 1;
            yg5 = 0;
            yg6 = 0;
        end;
        else do;
            yg4 = 0;
            if length(gname) > 3 then string1 = substr(gname,length(gname)-2,3);
            if length(gname) < 4 then string1 = substr(gname,1,length(gname));
            if length(gname2) > 3 then string2 = substr(gname2,length(gname2)-2,3);
            if length(gname2) < 4 then string2 = substr(gname2,1,length(gname2));
            if length(string1) = 3 and length(string2) = 3 and string1 = string2
                then do;
                    yg5 = 1;
                    yg6 = 0;
                end;
            else do
                yg5 = 0;
                yg6 = 1;
            end;
        end;
    end;
end;
end;
drop string1 string2;
run;

%closename(combine1,gname,f1,gname2,f2)
%closename(combine2,gname,f1,gname2,f3)
%closename(combine3,gname,f1,gname2,f4)
%closename(combine4,gname,f1,gname2,f5)
%closename(combine5,gname,f1,gname2,f6)
%closename(combine6,gname,f1,gname2,f7)
%closename(combine7,gname,f1,gname2,f8)
%closename(combine8,gname,f1,gname2,f9)
%closename(combine9,gname,f1,gname2,f10)
%closename(combine10,gname,f1,gname2,f11)
%closename(combine11,gname,f1,gname2,f12)
%closename(combine12,gname,f1,gname2,f13)
%closename(combine13,gname,f1,gname2,f14)
%closename(combine14,gname,f1,gname2,f15)

```

```

%closeiname(combine15,gname,f1,gname2,f16)
%closeiname(combine16,gname,f1,gname2,f17)
%closeiname(combine17,gname2,f1,gname,f2)
%closeiname(combine18,gname2,f1,gname,f3)
%closeiname(combine19,gname2,f1,gname,f4)
%closeiname(combine20,gname2,f1,gname,f5)
%closeiname(combine21,gname2,f1,gname,f6)
%closeiname(combine22,gname2,f1,gname,f7)
%closeiname(combine23,gname2,f1,gname,f8)
%closeiname(combine24,gname2,f1,gname,f9)
%closeiname(combine25,gname2,f1,gname,f10)
%closeiname(combine26,gname2,f1,gname,f11)
%closeiname(combine27,gname2,f1,gname,f12)
%closeiname(combine28,gname2,f1,gname,f13)
%closeiname(combine29,gname2,f1,gname,f14)
%closeiname(combine30,gname2,f1,gname,f15)
%closeiname(combine31,gname2,f1,gname,f16)
%closeiname(combine32,gname2,f1,gname,f17)

```

```

%closejoin(d1,combine1,combine2)
%closejoin(d2,d1,combine3)
%closejoin(d3,d2,combine4)
%closejoin(d4,d3,combine5)
%closejoin(d5,d4,combine6)
%closejoin(d6,d5,combine7)
%closejoin(d7,d6,combine8)
%closejoin(d8,d7,combine9)
%closejoin(d9,d8,combine10)
%closejoin(d10,d9,combine11)
%closejoin(d11,d10,combine12)
%closejoin(d12,d11,combine13)
%closejoin(d13,d12,combine14)
%closejoin(d14,d13,combine15)
%closejoin(d15,d14,combine16)
%closejoin(d16,d15,combine17)
%closejoin(d17,d16,combine18)
%closejoin(d18,d17,combine19)
%closejoin(d19,d18,combine20)
%closejoin(d20,d19,combine21)
%closejoin(d21,d20,combine22)
%closejoin(d22,d21,combine23)
%closejoin(d23,d22,combine24)
%closejoin(d24,d23,combine25)
%closejoin(d25,d24,combine26)
%closejoin(d26,d25,combine27)
%closejoin(d27,d26,combine28)
%closejoin(d28,d27,combine29)
%closejoin(d29,d28,combine30)
%closejoin(d30,d29,combine31)
%closejoin(joined,d30,combine32)

```

```

data joined;
set joined;
close="YES";

```

```

drop f1 - f17;
run;

proc sql;
create table close_matches as
select unique * from joined;
quit;

proc sort data=close_matches;
by num;
run;

data &dataset1;
merge &dataset1 close_matches;
by num;
if gname = ' ' or gname2 = ' ' then do;
    yg1=.;yg2=.;yg3=.;yg4=.;yg5=.;yg6=.;
    /*yg1=0;yg2=0;yg3=0;yg4=0;yg5=0;yg6=1;*/ /**USE WHEN CODING MISSING AS**/
    /**DIFFERENT**/
end;
else do;
if close = "YES" then do;
    yg1=0;yg2=1;yg3=0;yg4=0;yg5=0;yg6=0;
end;
else do;
    yg2 = 0;
end;
end;
drop close;
run;

data &dataset1;
set &dataset1;
if bplace = ' ' or bplace2 = ' ' then do;
    yb1=.;yb2=.;
    /*yb1=0;yb2=1;*/ /**USE WHEN CODING MISSING AS DIFFERENT**/
end;
else do;
    if bplace = bplace2 then yb1 = 1;
    else yb1 = 0;
    if bplace ne bplace2 then yb2 = 1;
    else yb2 = 0;
end;
run;

%close_bplace(combine1,bplace,f1,bplace2,f2)
%close_bplace(combine2,bplace2,f1,bplace,f2)

proc sql;
drop table joined;
create table joined as
select * from combine1
outer union corr
select * from combine2;

```

```

quit;

data joined;
set joined;
close="YES";
drop f1 f2;
run;

proc sql;
create table bplaces as
select unique * from joined;
quit;

proc sort data=bplaces;
by num;
run;

data records.&dataset1;
merge &dataset1 bplaces;
by num;
if close = "YES" then yb1 = 1;
if yb1 = 1 then yb2 = 0;
if ym = 1 then ctm = 1;
else ctm = 2;
if yr1 = 1 then ctr = 1;
if yr2 = 1 then ctr = 2;
if yr3 = 1 then ctr = 3;
if ya1 = 1 then cta = 1;
if ya2 = 1 then cta = 2;
if ya3 = 1 then cta = 3;
if yg1 = 1 then ctg = 1;
if yg2 = 1 then ctg = 2;
if yg3 = 1 then ctg = 3;
if yg4 = 1 then ctg = 4;
if yg5 = 1 then ctg = 5;
if yg6 = 1 then ctg = 6;
if yb1 = 1 then ctb = 1;
if yb2 = 1 then ctb = 2;
drop close;
run;
%mend vectors;

%vectors(samples)
%vectors(blockmiss)

```


Appendix F

SAS Code for Louisiana EM Algorithm

This SAS code is the EM algorithm programmed for the census index file of heads-of-household records from the 1910 and 1920 Censuses for Ascension Parish of Louisiana.

```
libname link '.';

%let myvar = ya1 ya2 ya3 yb1 yb2 yg1 yg2 yg3 yg4 yg5 yg6 yr1 yr2 yr3;

/*****
*Alternate method to get U;
*****/
ods output OneWayFreqs = Freqs;
proc freq data=link.samples;
tables &myvar;
run;
ods output close;

proc sort data=freqs;
by table;
data altu (keep=field u);
set freqs;
by table;
if last.table then do;
    field = substr(table,7,3);
    u = percent*.01;
    output;
end;
run;

proc iml;

use link.blockmiss;
read all var {&myvar} into y;

use altu;
read all var {u} into u;
altu = u';

*number of pairs;
```

```

np = nrow(y);

*number of agreement criteria across all fields;
nf = ncol(y);

*intial estimates of m, u, and p;
m = {.85 .14 .01 .99 .01 .719 .05 .13 .1 .0005 .0005 .98 .015 .005};
u = {.03 .01 .96 .8 .2 .005 .005 .005 .05 .005 .93 .44 .07 .49};
p = 0.05;

*intial estimates of gru and grm;
grm = j(np,1,0);
gru = j(np,1,0);

*Loop stop criteria;
epsilon = 5;
test = 0;

/*****/
*Begin EM Algorithm;
/*****/
do while (epsilon > 0.00001);
*do while (test < 5);

*E Step;
do j=1 to np by 1;
  /**USE THIS DO LOOP WHEN SUBSTITUTING MISSING VALUES WITH U ESTIMATES**/
  *do i=1 to nf by 1;
    *if y[j,i] = . then y[j,i] = u[i];
  *end;
  m_y = m##y[j,];
  u_y = u##y[j,];
  product1 = m_y[,#];
  product2 = u_y[,#];
  grm[j] = (p*product1)/(p*product1+(1-p)*product2);
  gru[j] = ((1-p)*product2)/(p*product1+(1-p)*product2);
end;

*M Step;
m_old = m;
u_old = u;

m = (grm#y)[+,]/grm[+];
u = (gru#y)[+,]/gru[+];
p = grm[+]/np;

*Check stop criteria;
epsilon = (m - m_old)*(m - m_old)' + (u - u_old)*(u - u_old)';
test = test + 1;

do i=1 to nf by 1;
  if m[i] = 0 then m[i] = 1/(2*np);
  if u[i] = 0 then u[i] = 1/(2*np);
end;

```

```

end;
/*****
*End EM Algorithm;
*****/

*Compute Weights;
wa = log(m/altu);
wd = log((1-m)/(1-altu));

*Compute Scores;
score = j(np,1,0);
do j=1 to np by 1;
  do i=1 to nf by 1;
    if y[j,i] = 1 then score[j] = score[j] + wa[i];
    else score[j] = score[j] + wd[i];
  end;
end;

*create output;
iter = test||epsilon;
create EMiter from iter;
append from iter;

parm = wa' || wd' || m' || altu' || u';
create EMparm from parm;
append from parm;
quit;

data EMiter(rename=(col1=Iterations col2=Epsilon));
set EMiter;
data EMparm(rename=(col1=W_Same col2=W_Diff col3=M col4=AltU col5=U));
set EMparm;
data fields;
input Fields$ @@;
datalines;
A_Same A_Close A_Diff
B_Same B_Diff
N_Same N_Nick N_1st3 N_1st N_Last3 N_Diff
R_Same R_Close R_Diff
run;

data EMparm;
merge EMiter fields EMparm;
run;

```


Appendix G

SAS Code for Louisiana Simulations

This SAS code generates data and runs it through the EM algorithm. It is set up in macros so all the simulation parameters are all set at the end.

```
libname link '.';

%macro simdata(mn, am1, am2, bm, gm1, gm2, gm3, gm4, gm5, rm1, rm2,
              un, au1, au2, bu, gu1, gu2, gu3, gu4, gu5, ru1, ru2);

data simm (drop=r u1 u2 u3 u4);
do r=1 to &mn;
  u1 = ranuni(0);u2 = ranuni(0);u3 = round(ranuni(0),.001);u4 = ranuni(0);
  ym=1;

  *Age;
  y11=0;y12=0;y13=0;
  if u1 lt &am1 then y11=1;
    else if &am1 le u1 lt (&am1+&am2) then y12 = 1;
    else y13 = 1;

  *BPlace;
  y21=0;y22=0;
  if u2 lt &bm then y21=1;
    else y22 = 1;

  *GName;
  y31=0;y32=0;y33=0;y34=0;y35=0;y36=0;
  if u3 lt &gm1 then y31=1;
    else if &gm1 le u3 lt (&gm1+&gm2) then y32 = 1;
    else if (&gm1+&gm2) le u3 lt (&gm1+&gm2+&gm3) then y33 = 1;
    else if (&gm1+&gm2+&gm3) le u3 lt (&gm1+&gm2+&gm3+&gm4) then y34 = 1;
    else if (&gm1+&gm2+&gm3+&gm4) le u3 lt (&gm1+&gm2+&gm3+&gm4+&gm5) then y35 = 1;
    else y36 = 1;

  *Race;
  y41=0;y42=0;y43=0;
  if u4 lt &rm1 then y41=1;
    else if &rm1 le u4 lt (&rm1+&rm2) then y42 = 1;
    else y43 = 1;
```

```

output;
end;
run;

data simu (drop=r u1 u2 u3 u4);
do r=1 to &n;
u1 = ranuni(0);u2 = ranuni(0);u3 = ranuni(0);u4 = ranuni(0);
ym=0;

*Age;
y11=0;y12=0;y13=0;
if u1 lt &au1 then y11=1;
    else if &au1 le u1 lt (&au1+&au2) then y12 = 1;
    else y13 = 1;

*BPlace;
y21=0;y22=0;
if u2 lt &bu then y21=1;
    else y22 = 1;

*GName;
y31=0;y32=0;y33=0;y34=0;y35=0;y36=0;
if u3 lt &gu1 then y31=1;
    else if &gu1 le u3 lt (&gu1+&gu2) then y32 = 1;
    else if (&gu1+&gu2) le u3 lt (&gu1+&gu2+&gu3) then y33 = 1;
    else if (&gu1+&gu2+&gu3) le u3 lt (&gu1+&gu2+&gu3+&gu4) then y34 = 1;
    else if (&gu1+&gu2+&gu3+&gu4) le u3 lt (&gu1+&gu2+&gu3+&gu4+&gu5) then y35 = 1;
else y36 = 1;

*Race;
y41=0;y42=0;y43=0;
if u4 lt &ru1 then y41=1;
    else if &ru1 le u4 lt (&ru1+&ru2) then y42 = 1;
    else y43 = 1;
output;
end;
run;

data link.sim;
set simm simu;
run;

data true;
am1 = &am1;am2 = &am2;am3 = 1-(&am1+&am2);
bm1 = &bm;bm2 = 1-&bm;
gm1 = &gm1;gm2 = &gm2;gm3=&gm3;gm4 = &gm4;gm5 = &gm5;
gm6=1-(&gm1+&gm2+&gm3+&gm4+&gm5);
rm1 = &rm1;rm2 = &rm2;rm3=1-(&rm1+&rm2);
au1 = &au1;au2 = &au2;au3=1-(&au1+&au2);
bu1 = &bu;bu2 = 1-&bu;
gu1 = &gu1;gu2 = &gu2;gu3=&gu3;gu4 = &gu4;gu5 = &gu5;
gu6=1-(&gu1+&gu2+&gu3+&gu4+&gu5);
ru1 = &ru1;ru2 = &ru2;ru3=1-(&ru1+&ru2);
run;

```

```

%mend;

data fields;
input Fields$ @@;
datalines;
A_Same A_Close A_Diff
B_Same B_Diff
N_Same N_Nick N_1st3 N_1st N_Last3 N_Diff
R_Same R_Close R_Diff
run;

%macro sim(dataset, iter, epsilon,
            miss, pa, pb, pn, pr,
            mn, am1, am2, bm, gm1, gm2, gm3, gm4, gm5, rm1, rm2,
            un, au1, au2, bu, gu1, gu2, gu3, gu4, gu5, ru1, ru2);

%do k=1 %to &iter %by 1;

%simdata(&mn, &am1, &am2, &bm, &gm1, &gm2, &gm3, &gm4, &gm5, &rm1, &rm2,
        &un, &au1, &au2, &bu, &gu1, &gu2, &gu3, &gu4, &gu5, &ru1, &ru2);

%if &miss = MISS %then %do;
data link.sim (drop=un ua ur ub);
set link.sim;
ua = ranuni(0);ub = ranuni(0);un = ranuni(0);ur = ranuni(0);
if ua lt &pa then do;
    y11=.;y12=.;y13=.;
end;
if ub lt &pb then do;
    y21=.;y22=.;
end;
if un lt &pn then do;
    y31=.;y32=.;y33=.;y34=.;y35=.;y36=.;
end;
if ur lt &pr then do;
    y41=.;y42=.;y43=.;
end;
run;
%end;

%let trum = am1 am2 am3 bm1 bm2 gm1 gm2 gm3 gm4 gm5 gm6 rm1 rm2 rm3;
%let truu = au1 au2 au3 bu1 bu2 gu1 gu2 gu3 gu4 gu5 gu6 ru1 ru2 ru3;
%let simvar = y11 y12 y13 y21 y22 y31 y32 y33 y34 y35 y36 y41 y42 y43;

/*****
*Alternate method to get U;
*****/
data sample (drop=i sampsize);
    sampsize=5000;
    do i=1 to sampsize;
        pickit=ceil(ranuni(0)*totobs);
        set link.sim point=pickit
            nobs=totobs;
        output;
    end;
run;

```



```

        end;
        stop;
    run;

data link.&dataset&k;
set link.sim;
run;

ods output OneWayFreqs = Freqs;
proc freq data=sample;
tables &simvar;
run;
ods output close;

proc sort data=freqs;
by table;

data altu (keep=field u);
set freqs;
by table;
if last.table then do;
    field = substr(table,7,3);
    u = percent*.01;
    output;
end;
run;

proc iml;

use link.sim;
read all var {&simvar} into y;

use true;
read all var {&trum} into trum;

use true;
read all var {&truu} into truu;

use altu;
read all var {u} into u;
altu = u';

*number of pairs;
np = nrow(y);

*number of agreement criteria across all fields;
nf = ncol(y);

*intial estimates of m, u, and p;
m = {.8 .18 .02 .97 .03 .688 .08 .13 .1 .001 .001 .94 .03 .03};
u = {.03 .01 .96 .8 .2 .005 .005 .005 .05 .005 .93 .44 .07 .49};
p = 0.05;

*intial estimates of gru and grm;

```

```

grm = j(np,1,0);
gru = j(np,1,0);

*Loop stop criteria;
epsilon = 5;
test = 0;

/*****/
*Begin EM Algorithm;
/*****/
do while (epsilon > &epsilon);
*do while (test < 5);

*E Step;
do j=1 to np by 1;
  /**USE THIS DO LOOP WHEN SUBSTITUTING MISSING VALUES WITH U ESTIMATES**/
  *do i=1 to nf by 1;
    *if y[j,i] = . then y[j,i] = u[i];
  *end;
  m_y = m##y[j,];
  u_y = u##y[j,];
  product1 = m_y[,#];
  product2 = u_y[,#];
  grm[j] = (p*product1)/(p*product1+(1-p)*product2);
  gru[j] = ((1-p)*product2)/(p*product1+(1-p)*product2);
end;

*M Step;
m_old = m;
u_old = u;

m = (grm#y)[+,]/grm[+];
u = (gru#y)[+,]/gru[+];
p = grm[+]/np;

*Check stop criteria;
epsilon = (m - m_old)*(m - m_old)' + (u - u_old)*(u - u_old)';
test = test + 1;

do i=1 to nf by 1;
  if m[i] = 0 then m[i] = 1/(2*np);
  if u[i] = 0 then u[i] = 1/(2*np);
end;

end;
/*****/
*End EM Algorithm;
/*****/

diffm = trum' - m';

*create output;
diff = trum' || m' || diffm || truu' || u' || altu';
create diff from diff;

```

```

append from diff;
quit;

data diff(rename=(col1=trum col2=m col3=diffm col4=truu col5=u col6=altu));
set diff;
run;

%if &k=1 %then %do;
data link.&dataset;
merge fields diff;
run;
%end;

%else %do;
data tmp;
merge fields diff;
run;
data link.&dataset;
set tmp link.&dataset;
run;
%end;

proc freq data=link.sim;
tables _all_/list;
run;
%end;

proc print data=link.&dataset;
run;
%mend;

%sim(finale1, 15, .0001,
    NO, 1, 1, 1, 1,
    500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
    100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(finale2, 15, .00001,
    NO, 1, 1, 1, 1,
    500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
    100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(finale3, 15, .000001,
    NO, 1, 1, 1, 1,
    500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
    100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(finale4, 15, .0001,
    MISS, .02, .04, .01, .001,
    500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
    100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(finale5, 15, .00001,
    MISS, .02, .04, .01, .001,
    500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,

```

```

100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(finale6, 15, .000001,
MISS, .02, .04, .01, .001,
500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final1, 10, .000001,
NO, 1, 1, 1, 1,
50, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
10000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final2, 10, .000001,
NO, 1, 1, 1, 1,
500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final3, 10, .000001,
NO, 1, 1, 1, 1,
2500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
500000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final4, 10, .000001,
NO, 1, 1, 1, 1,
500, .9, .08, .9, .5, .13, .15, .17, .025, .8, .1,
100000, .1, .05, .75, .01, .01, .02, .1, .1, .3, .1);

%sim(final5, 10, .000001,
NO, 1, 1, 1, 1,
500, .5, .3, .7, .4, .05, .2, .25, .05, .7, .2,
100000, .2, .1, .6, .01, .015, .02, .07, .1, .2, .3);

%sim(final6, 10, .000001,
MISS, .02, .04, .01, .001,
500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final7, 10, .000001,
MISS, .04, .08, .02, .002,
500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final8, 10, .000001,
MISS, .005, .01, .0005, .02,
500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
100000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final9, 10, .000001,
MISS, .02, .04, .01, .001,
2500, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
500000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final10, 10, .000001,
NO, 1, 1, 1, 1,

```

```

5000, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
1000000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

%sim(final11, 10, .000001,
MISS, .02, .04, .01, .001,
5000, .7, .25, .97, .6, .1, .12, .14, .02, .9, .05,
1000000, .03, .01, .85, .003, .003, .005, .049, .04, .4, .1);

```

Appendix H

SAS Code for Louisiana Grid Simulations

This SAS code the EM algorithm macro used for the grid simulations done with datasets Finale614 and Finale615. It is set up in a macro so all the simulation parameters are all set at the end.

```
libname link '.';

data fields;
input Fields$ @@;
datalines;
A_Same A_Close A_Diff
B_Same B_Diff
N_Same N_Nick N_1st3 N_1st N_Last3 N_Diff
R_Same R_Close R_Diff
run;

%let simvar = y11 y12 y13 y21 y22 y31 y32 y33 y34 y35 y36 y41 y42 y43;
%macro mygrid(indata,outdata,iter,epsilon,gs1,gs6);

%do k=1 %to &iter %by 1;
proc iml;

use link.&indata;
read all var {%simvar} into y;

*number of pairs;
np = nrow(y);

*number of agreement criteria across all fields;
nf = ncol(y);

*intial estimates of m, u, and p;
m = {.8 .18 .02 .97 .03 &gs1 .1 .12 .14 .02 &gs6 .94 .03 .03};
u = {.03 .01 .96 .8 .2 .005 .005 .005 .05 .005 .93 .44 .07 .49};
p = 0.05;

*intial estimates of gru and grm;
grm = j(np,1,0);
```

```

gru = j(np,1,0);

*Loop stop criteria;
epsilon = 5;
test = 0;

/*****/
*Begin EM Algorithm;
/*****/
do while (epsilon > &epsilon);
*do while (test < 5);

    *E Step;
    do j=1 to np by 1;
        product1 = 1;
        product2 = 1;
        m_y = m##y[j,];
        u_y = u##y[j,];
        do i=1 to nf by 1;
            product1 = product1 # m_y[i];
            product2 = product2 # u_y[i];
        end;
        grm[j] = (p*product1)/(p*product1+(1-p)*product2);
        gru[j] = ((1-p)*product2)/(p*product1+(1-p)*product2);
    end;

    *M Step;
    m_old = m;
    u_old = u;

    m = (grm'*y)/(grm'*j(np,1,1));
    u = (gru'*y)/(gru'*j(np,1,1));
    p = (grm'*j(np,1,1))/np;

    *Check stop criteria;
    epsilon = (m - m_old)*(m - m_old)' + (u - u_old)*(u - u_old)';
    test = test + 1;

end;
/*****/
*End EM Algorithm;
/*****/

*Compute Weights;
wa = log(m/u);
wd = log((1-m)/(1-u));

*Compute Scores;
score = j(np,1,0);
do j=1 to np by 1;
    do i=1 to nf by 1;
        if y[j,i] = 1 then score[j] = score[j] + wa[i];
        else score[j] = score[j] + wd[i];
    end;
end;

```

```

end;

*create output;
iter = test||epsilon;
parm = wa' || wd' || m' || u';

create EMiter from iter;
append from iter;

create EMparm from parm;
append from parm;
quit;

data EMiter(rename=(col1=Iterations col2=Epsilon));
set EMiter;
data EMparm(rename=(col1=W_Same col2=W_Diff col3=M col4=U));
set EMparm;
run;

%if &k=1 %then %do;
data link.&outdata;
merge EMiter fields EMparm;
run;
%end;

%else %do;
data tmp;
merge EMiter fields EMparm;
run;

data link.&outdata;
set tmp link.&outdata;
run;
%end;

%end;

title "&outdata";

proc print data=link.&outdata;
run;
%mend;

%mygrid(finale615, finalg1, 5, .0001, .619, .001);
%mygrid(finale615, finalg2, 5, .0001, .61, .01);
%mygrid(finale615, finalg3, 5, .0001, .59, .03);
%mygrid(finale615, finalg4, 5, .0001, .581, .039);
%mygrid(finale615, finalg5, 5, .00001, .619, .001);
%mygrid(finale615, finalg6, 5, .00001, .61, .01);
%mygrid(finale615, finalg7, 5, .00001, .59, .03);
%mygrid(finale615, finalg8, 5, .00001, .581, .039);
%mygrid(finale615, finalg9, 5, .000001, .619, .001);
%mygrid(finale615, finalg10, 5, .000001, .61, .01);
%mygrid(finale615, finalg11, 5, .000001, .59, .03);

```



```

%mygrid(finale615, finalg12, 5, .000001, .581, .039);

%mygrid(finale614, finalg1, 1, .0001, .619, .001);
%mygrid(finale614, finalg2, 1, .0001, .61, .01);
%mygrid(finale614, finalg3, 1, .0001, .59, .03);
%mygrid(finale614, finalg4, 1, .0001, .581, .039);
%mygrid(finale614, finalg5, 1, .00001, .619, .001);
%mygrid(finale614, finalg6, 1, .00001, .61, .01);
%mygrid(finale614, finalg7, 1, .00001, .59, .03);
%mygrid(finale614, finalg8, 1, .00001, .581, .039);
%mygrid(finale614, finalg9, 1, .000001, .619, .001);
%mygrid(finale614, finalg10, 1, .000001, .61, .01);
%mygrid(finale614, finalg11, 1, .000001, .59, .03);
%mygrid(finale614, finalg12, 1, .000001, .581, .039);

```

Appendix I

SAS Code for Boxplots

This SAS code was used to create the boxplots in the paper.

```
libname link 'C:\Documents and Settings\John\My Documents\School\RecordLinkage
\LA Files\Sim';
libname linkalt 'C:\Documents and Settings\John\My Documents\School\RecordLinkage
\LA Files\Sim_Alt';
libname linkmiss 'C:\Documents and Settings\John\My Documents\School\RecordLinkage
\LA Files\Miss_Methods';

proc sort data=linkalt.final1;
by fields;
run;

proc sort data=link.final5;
by fields;
run;

proc sort data=linkalt.final3;
by fields;
run;

proc sort data=linkmiss.out1;
by fields;
run;

proc sort data=linkmiss.out2;
by fields;
run;

proc sort data=linkmiss.out3;
by fields;
run;

goptions ftext=swissx ;

symbol interpol=boxt00
        co=blue
        bwidth=4
        width=2;
```

```

axis1 label=(font=swissxb 'Field Match Status')
      value=(font=zapf t=1 'Age' j=c 'Close'
            t=2 'Age' j=c 'Diff'
            t=3 'Age' j=c 'Same'
            t=4 'Birth' j=c 'Diff'
            t=5 'Birth' j=c 'Same'
            t=6 'Name' j=c '1st'
            t=7 'Name' j=c '1st 3'
            t=8 'Name' j=c 'Diff'
            t=9 'Name' j=c 'Last 3'
            t=10 'Name' j=c 'Nick'
            t=11 'Name' j=c 'Same'
            t=12 'Race' j=c 'Close'
            t=13 'Race' j=c 'Diff'
            t=14 'Race' j=c 'Same')
      offset=(3,3);

axis2 label=(angle = 90 font=swissxb 'True Value - Estimate')
      order=(-.6 to .4 by .2)
      value=(font=zapf)
      minor=none;

*title 'Sample Size: 10,050 |M| = 50';
proc gplot data=linkalt.final1;
plot diffm*fields/haxis=axis1 vaxis=axis2;
run;

*title 'Sample Size: 100,500 |M| = 500';
proc gplot data=link.final5;
plot diffm*fields/haxis=axis1 vaxis=axis2;
run;

*title 'Sample Size: 502,500 |M| = 2500';
proc gplot data=linkalt.final3;
plot diffm*fields/haxis=axis1 vaxis=axis2;
run;

axis2 label=(angle = 90 font=swissxb 'True Value - Estimate')
      order=(-.075 to .075 by .025)
      value=(font=zapf)
      minor=none;

*title 'Pairs with Missing Entries Coded as Different';
proc gplot data=linkmiss.out1;
plot diffm*fields/haxis=axis1 vaxis=axis2;
run;

*title 'Pairs with Missing Entries Coded as Missing';
proc gplot data=linkmiss.out2;
plot diffm*fields/haxis=axis1 vaxis=axis2;
run;

*title 'Pairs with Missing Entries Substituted with U';

```

```
proc gplot data=linkmiss.out3;  
plot diffm*fields/haxis=axis1 vaxis=axis2;  
run;
```