Bayesian Analysis (0000) **00**, Number 0, pp. 1

Efficient and Scalable Bipartite Matching with Fast Beta Linkage (fabl)

Brian Kundinger[*], Jerome P. Reiter[*] and Rebecca C. Steorts[†]

Abstract. Within the field of record linkage, Bayesian methods have the crucial advantage of quantifying uncertainty from imperfect linkages. However, current implementations of Bayesian Fellegi Sunter models are computationally intensive, making them challenging to use on larger-scale record linkage tasks. To address these computational considerations, we propose fast beta linkage (`fabl`), an extension to the Beta Record Linkage (`BRL`) method of Sadinle (2017). Specifically, we use independent prior distributions over the matching space, allowing us to use hashing techniques that reduce computational overhead. This also allows us to complete pairwise record comparisons over large datasets through parallel computing and to reduce memory costs through a new technique called storage efficient indexing. Through simulations and two case studies, we show that `fabl` has markedly increased speed with minimal loss of accuracy when compared to `BRL`.

Keywords: data fusion, distributed computing, entity resolution, hashing, record linkage.

1 Introduction

In many data analysis tasks, analysts seek to identify duplicate records across two datafiles. This is an increasingly important task in "data cleaning," and is used for inferential and predictive analyses in fields such as statistics, computer science, machine learning, political science, economics, precision medicine, official statistics, and others (Christen, 2012; Gutman et al., 2013; Dalzell and Reiter, 2018; Tang et al., 2020). In this article, we consider bipartite record linkage, which merges two datafiles that contain duplications across, but not within, the respective datafiles.

Many statistical record linkage methods are extensions of the seminal work of Fellegi and Sunter (1969b) and Newcombe et al. (1959). Specifically, Fellegi and Sunter (1969b) created comparison vectors for each pair of records in the data and independently classified each pair as a match or a non-match using a likelihood ratio test. Recent work in the statistical literature has extended this approach for a wide variety of applications (Winkler and Thibaudeau, 1990; Fair, 2004; Wagner et al., 2014; Gill and Goldacre, 2003; Enamorado et al., 2019; Aleshin-Guendel and Sadinle, 2022). Additionally, some methods model records directly (Steorts et al., 2016; Marchant et al., 2021; Betancourt et al., 2022), but in this paper, we build on the contributions to the comparison vector approach.

The independent pairwise matching assumption from Fellegi and Sunter (1969b) is popular mainly for its mathematical simplicity, but can be unreasonable in practice. In many situations, we know that there are no duplications within a datafile, meaning that

[*]Department of Statistical Science, Duke University, P.O. Box 90251, Durham, NC 27708, USA brian.kundinger@duke.edu, jreiter@duke.edu

[†]Departments of Statistical Science and Computer Science, Duke University, P.O. Box 90251, Durham, NC 27708, USA beka@stat.duke.edu

each record in one file should be linked with at most one other record in the other file. Thus, when the procedure results in many-to-one matches, some of these links must be false. Many extensions to Fellegi and Sunter (1969b) resolve these false matches as a post-processing step (Jaro, 1989), but this model misspecification can still lead to poor results (Sadinle, 2017).

Alternatively, one can embed one-to-one matching requirements into the model specification itself (Gutman et al., 2013; Tancredi and Liseo, 2011), at an additional computational cost. Larsen (2005) employed a Metropolis-Hastings algorithm to only allow sampling matches that respected one-to-one assumptions, but such algorithms exhibit slow mixing due to the combinatorial nature of the constrained matching space. Fortunato (2010) used simulated annealing to target the space of matches permitted under the one-to-one constraint, but the method is computationally intensive and, to our knowledge, has not been applied on datafiles with more than 100 records. Sadinle (2017) proposed the Beta Record Linkage model (`BRL`), using a prior distribution over the space of bipartite matchings to strictly enforce one-to-one requirements throughout a Gibbs sampler. Additionally, he introduced a class of loss functions that allows for a flexible estimation of the linkage structure, such that the modeler can weigh the relative importance of false positives and false negatives, and identify records pairings to be decided through clerical review. `BRL` was shown to work on larger tasks than previous one-to-one methods, but in our experience, it becomes slow when applied to files with more than a few thousand records.

In this paper, we propose fast beta linkage (`fabl`), which extends the `BRL` model for increased efficiency and scalability. Following the suggestion in Wortman (2019), we relax the one-to-one matching requirement of `BRL` and use independent priors over the matching space. This allows us to (1) employ hashing techniques that speed up calculations and reduce computational costs, (2) compute the pairwise record comparisons over large datasets via parallel computing, and (3) reduce memory costs through what we call storage efficient indexing. These contributions allow `fabl` to perform record linkage on much larger datafiles than previous Bayesian Fellegi Sunter models at significantly increased speed with minimal loss of accuracy. In particular, computation time under `BRL` grows quadratically, with the size of each datafile, while computation time under `fabl` grows linearly, only with the size of the smaller datafile.

In what follows, Section 2 reviews the work of Fellegi and Sunter (1969b) and Sadinle (2017). Section 3 proposes the `fabl` model, provides the Gibbs sampler for posterior inference, and shows the loss function used to calculate the Bayes estimate for the bipartite matching. Section 4 introduces the hashing technique and storage efficient indexing used to increase the speed of calculations and the scale of linkage tasks amenable to `fabl`. Sections 5 and 6 demonstrate the speed and accuracy of `fabl` through simulation studies and case studies of homicides from the El Salvadoran Civil War and the National Long Term Care Study.

2 Review of Prior Work

Consider two datafiles $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ containing records $\{x_{1i}\}_{i=1}^{n_1}$ and $\{x_{2j}\}_{j=1}^{n_2}$ respectively. Without loss of generality, denote files such that $n_1 \geq n_2$. We follow the convention established by Sadinle (2017) and say "record $i \in X_1$" rather than the more compact x_{1i} for ease of readability. Under bipartite matching, the set of matches across datafiles can be represented in two equivalent ways. First, we may use a matrix $\Delta \in \{0,1\}^{n_1 \times n_2}$, where

$$\Delta_{ij} = \begin{cases} 1, & \text{if records } i \in \boldsymbol{X}_1 \text{ and } j \in \boldsymbol{X}_2 \text{ refer to the same entity;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Though intuitive, this sparse matrix representation can become cumbersome for large linkage tasks. More compactly, bipartite matching also can be viewed as a labeling $\boldsymbol{Z} = (Z_1, \dots, Z_{n_2})$ for the records in $\boldsymbol{X}_2$ such that

$$Z_j = \begin{cases} i, & \text{if records } i \in \boldsymbol{X}_1 \text{ and } j \in \boldsymbol{X}_2 \text{ refer to the same entity;} \\ n_1 + j, & \text{if record } j \in \boldsymbol{X}_2 \text{ does not have a match in } \boldsymbol{X}_1. \end{cases} \tag{2}$$

We can go back and forth between the two using $\Delta_{ij} = I(Z_j = i)$, where $I(\cdot) = 1$ when the expression inside the parentheses is true, and $I(\cdot) = 0$ otherwise.

Denote the set of matches by $\boldsymbol{M} = \{(i,j) : i \in \boldsymbol{X}_1, j \in \boldsymbol{X}_2, \Delta_{ij} = 1\}$, and the set of non-matches by $\boldsymbol{U} = \{(i,j) : i \in \boldsymbol{X}_1, j \in \boldsymbol{X}_2, \Delta_{ij} = 0\}$. The record linkage task can be viewed as identifying $\boldsymbol{M}$ and $\boldsymbol{U}$. We refer to record pairs that are estimated as matches as "links" and to record pairs that are estimated as non-matches as "non-links."

Intuitively, matching records (those that refer to the same entity) should be similar; records that are non-matching should be dissimilar. Fellegi and Sunter (1969b) proposed encoding this using a comparison vector $\boldsymbol{\gamma}_{ij}$ computed for each record pair (i,j) in $\boldsymbol{X}_1 \times \boldsymbol{X}_2$. Denote the number of criteria for comparing records by F, such that $\boldsymbol{\gamma}_{ij} = (\gamma_{ij}^1, \gamma_{ij}^2, \dots, \gamma_{ij}^f, \dots, \gamma_{ij}^F)$. In most cases, $\boldsymbol{\gamma}_{ij}$ consists of one comparison for each feature shared between the two datasets. We refer to set of all comparison vectors $\boldsymbol{\gamma}_{ij}$ as Γ.

The simplest way to compare any particular feature for two records is to check for exact agreement, and this is commonly used for categorical features. For numerical features, we can use absolute difference between the two feature values. For text features, such as names, we can use string distance metrics such as Levenstein or Jaro-Winkler distance (Cohen et al., 2003). We then set thresholds that allow us to represent comparisons through discrete levels of disagreement (Bilenko and Mooney, 2006; Elmagarmid et al., 2007).

Let $\mathcal{S}_f(i,j)$ denote a general similarity measure for feature f of records i and j, where the range of $\mathcal{S}_f$ can be divided into $L_f + 1$ intervals denoted by $I_{f0}, I_{f1}, \dots, I_{fL_f}$. Following convention, I_{f0} represents the highest level of agreement (inclusive of complete agreement) and I_{fL_f} represents the highest level of disagreement (including complete disagreement). Thus, we can construct comparison vectors such that $\gamma_{ij}^f = l$ if $\mathcal{S}_f(i,j) \in I_{fl}$. The choices of I_{fl} are application specific, as we discuss in the simulation and case studies.

In the construction of comparison vectors, we note it is common to encounter missing information in datasets used for record linkage. Whether information is missing in record $i \in \boldsymbol{X}_1$ or $j \in \boldsymbol{X}_2$, the comparison vector γ_{ij} will have missing values. We assume that this missingness occurs at random (MAR, by Little and Rubin (2002)). With the MAR, we can marginalize over the missing data, and do all computation simply using the observed data. To notate this, we use the indicator $I_{obs}(\cdot)$, which takes on the value 1 when the argument inside the parentheses is observed, and 0 otherwise.

2.1 Fellegi-Sunter Models

The seminal Fellegi and Sunter (1969a) model employs two independence assumptions: first, that comparison vectors are conditionally independent given their matching status, and second, that the matching status of the record pairs are independent. These allow us to model the comparison data using a mixture model, where

$$\begin{aligned}\Gamma_{ij} = \gamma_{ij} \mid \Delta_{ij} = 1 &\overset{iid}{\sim} \mathcal{M}(\boldsymbol{m}), \\ \Gamma_{ij} = \gamma_{ij} \mid \Delta_{ij} = 0 &\overset{iid}{\sim} \mathcal{U}(\boldsymbol{u}), \\ \Delta_{ij} &\overset{iid}{\sim} \text{Bernoulli}(\lambda).\end{aligned} \tag{3}$$

Here $\mathcal{M}$ and $\mathcal{U}$ are the distributions for matching and non-matching record pairs, $\boldsymbol{m}$ and $\boldsymbol{u}$ are their respective sets of parameters, and λ is the marginal probability that a record pair is a match. When using comparison vectors with discrete agreement levels, $\mathcal{M}$ and $\mathcal{U}$ are collections of independent multinomial distributions for each linkage feature. Accordingly, $\boldsymbol{m} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_F)$, where $\boldsymbol{m}_f = (m_{f1}, \ldots, m_{fL_f})$ and $m_{fl} = P(\gamma_{ij}^f = l|\Delta_{ij} = 1)$ for all fields f and agreement levels l. The $\boldsymbol{u}$ parameters are defined similarly, with $u_{fl} = P(\gamma_{ij}^f = l|\Delta_{ij} = 0)$.

Each record pair is then independently classified as a match or nonmatch according to the likelihood ratio

$$w_{ij} = \frac{P(\gamma_{ij} \mid \Delta_{ij} = 1)}{P(\gamma_{ij} \mid \Delta_{ij} = 0)}. \tag{4}$$

This ratio is large when there is strong evidence of the pair being a match, and small otherwise. All record pairs with weights above a specified threshold are considered a match. Such independent classifications often leads to matchings that violate one-on-one matching assumptions, which prompted Jaro (1989) to formulate an optimization problem that produces a bipartite matching maximizing the sum of the Fellegi Sunter weights among matched pairs.

To estimate a one-to-one matching without using a post-processing step, Sadinle (2017) incorporated this constraint directly into the model through a Bayesian framework. In addition to standard Dirichlet priors for the $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$ parameters, he proposed the "beta distribution for bipartite matching" for the linkage parameter $\boldsymbol{Z}$. To construct this prior, he assigned a prior to the probability of the indicator that a record in $\boldsymbol{X}_2$ had a match in $\boldsymbol{X}_1$, so that $I(Z_j \leq n_1) \sim \text{Bernoulli}(\pi)$, where π itself is taken to be distributed

$\text{Beta}(\alpha_\pi, \beta_\pi)$. It follows that the number of records in $\boldsymbol{X}_2$ that have matches, denoted $n_{12}(\boldsymbol{Z}) = \sum_{j=1}^{n_2} I(Z_j \leq n_1 + j)$, is distributed according to a Beta-Binomial$(n_2, \alpha_\pi, \beta_\pi)$. Conditioning on the set of records in $\boldsymbol{X}_2$ that have matches, formally denoted $\{I(Z_j \leq n_1)\}_{j=1}^{n_2}$, all $n_1!/(n_1 - n_{12}(\boldsymbol{Z}))!$ bipartite matchings are taken to be equally likely. Thus the beta distribution for bipartite matching, is given by

$$P(\boldsymbol{Z}|\alpha_\pi, \beta_\pi) = \frac{(n_1 - n_{12}(\boldsymbol{Z}))!}{n_1!} \frac{\mathrm{B}(n_{12}(\boldsymbol{Z}) + \alpha_\pi, n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi)}{\mathrm{B}(\alpha_\pi, \beta_\pi)}, \tag{5}$$

where $\mathrm{B}(\cdot, \cdot)$ represents the Beta function. This prior strictly enforces one-to-one matching, inducing a Gibbs sampler that removes previously matched records from the set of candidate records when sampling Z_j. This makes the sampler inherently serial, which can be slow when working on linkage tasks with more than a few thousand records.

3 Fast Beta Linkage

In contrast to the prior over the vector $\boldsymbol{Z}$ from Sadinle (2017), we follow Wortman (2019) and use independent priors for each component Z_j. However, unlike Wortman (2019) who proposes a flat prior for Z_j, we use a proper prior. We denote the fast Beta prior as follows:

$$Z_j|\pi = \begin{cases} \frac{1}{n_1}\pi, & z_j \leq n_1; \\ 1 - \pi, & z_j = n_1 + j; \end{cases} \tag{6}$$
$$\pi \sim \text{Beta}(\alpha_\pi, \beta_\pi).$$

We can interpret equation 6 as follows: record $j \in \boldsymbol{X}_2$ has some match in $\boldsymbol{X}_1$ with probability π, and each record $i \in \boldsymbol{X}_1$ is equally likely to be that match. The hyperparameters α_π and β_π encode prior beliefs about the proportion of records in $\boldsymbol{X}_2$ that have matches in $\boldsymbol{X}_1$.

Previously, Wortman (2019) proposed a flat prior over the labeling of $\boldsymbol{Z}$, meaning that each value $\{1, \ldots, n_1, n_1 + j\}$ was considered equally likely for Z_j. This however amounts to a prior probability of $\frac{n_1}{n_1+1}$ that record j has a match in $\boldsymbol{X}_1$, and thus tended to over match, leading to poor precision when used in practice. Indeed we find that the flat prior is equivalent to a special case of the fast Beta prior where π fixed at the mean of a Beta $\left(1, \frac{1}{n_1}\right)$ random variable.

Note that linkage in this setting is conducted at the record level, rather than at the record pair level as in the Fellegi Sunter model. That is, π under `fabl` estimates the proportion of records in $\boldsymbol{X}_2$ that have matches, while λ in the Fellegi Sunter model estimates the proportion of record pairs that are matches. We find π to be more interpretable parameter than λ in the bipartite case. In this setting, there are at most n_2 matching pairs out of $n_1 n_2$ total pairs, meaning that λ is bounded above by $\frac{1}{n_1}$ and tends towards 0 as the size of the linkage task grows. Additionally, while the Fellegi Sunter model makes $n_1 \times n_2$ independent matching decisions and `BRL` makes n_2 dependent matching decisions, `fabl` strikes a middle ground between the two, making

Symbol	Description
$\boldsymbol{X}_1, \boldsymbol{X}_2$	datasets
$i \in 1, \dots, n_1$	index over records in $\boldsymbol{X}_1$
$j \in 1, \dots, n_2$	index over records in $\boldsymbol{X}_2$
$f \in 1, \dots F$	index over fields used for comparisons
$l \in 1, \dots L_f$	index over agreement levels for feature f
n_{12}	number of entities in common between $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$
$\boldsymbol{\gamma}_{ij}$	comparison vector for records $i \in \boldsymbol{X}_1$ and $j \in \boldsymbol{X}_2$
$Z_j = i$	records $i \in \boldsymbol{X}_1$ and $j \in \boldsymbol{X}_2$ match
$Z_j = n_1 + j$	record $j \in \boldsymbol{X}_2$ has no match in $\boldsymbol{X}_1$
m_{fl}	$P(\gamma_{ij}^f = l \mid Z_j = i)$
u_{fl}	$P(\gamma_{ij}^f = l \mid Z_j \neq i)$
π	probability that a record in $\boldsymbol{X}_2$ has a match in $\boldsymbol{X}_1$

Table 1: Summary of model notation

n_2 independent matching decisions. As shown in Sections 5 and 6, this allows `fabl` to achieve the benefits from `BRL` while making efficiency gains possible by exploiting independence.

To obtain an estimate $\hat{\boldsymbol{Z}}$ of the linkage structure, we use the loss functions and Bayes estimate from Sadinle (2017), reproduced in Appendix . Since the prior does not strictly enforce one-to-one matching, it is possible for this Bayes estimate to link multiple records in $\boldsymbol{X}_2$ to one record in $\boldsymbol{X}_1$. To achieve a Bayes estimate that fulfills one-to-one matching requirement, we minimize the expected loss subject to the constraint that $\hat{Z}_j \neq \hat{Z}_{j'}$ for all $j \neq j'$. In the event that we have two records j and j' such that both $P(\hat{Z}_j = i|\Gamma) > \frac{1}{2}$ and $P(\hat{Z}_{j'} = i|\Gamma) > \frac{1}{2}$, we accept the match with the highest posterior probability, and declare the other to have no match. A similar approach appears in the most probable maximal matching sets used by Steorts et al. (2016) to match records to latent entities.

For clarity, we present our full model below. A summary of notation is provided in Table 1.

$$\mathcal{L}(\boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{u} \mid \boldsymbol{\Gamma}) = \prod_{j=1}^{n_1} \prod_{i=1}^{n_2} \prod_{f=1}^{F} \prod_{l=1}^{L_f} \left[m_{fl}^{I(Z_j = i)} u_{fl}^{I(Z_j \neq i)} \right]^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)}, \tag{7}$$

$$\boldsymbol{m}_f \sim \text{Dirichlet}(\alpha_{f0}, \dots, \alpha_{fL_f}),$$
$$\boldsymbol{u}_f \sim \text{Dirichlet}(\beta_{f0}, \dots, \beta_{fL_f}),$$
$$Z_j | \pi = \begin{cases} \frac{1}{n_1}\pi, & z_j \leq n_1; \\ 1 - \pi, & z_j = n_1 + j; \end{cases}$$
$$\pi \sim \text{Beta}(\alpha_\pi, \beta_\pi).$$

3.1 Gibbs Sampler

We initialize $\boldsymbol{m}$ and $\boldsymbol{u}$ from random draws from their prior distributions, and initialize $\boldsymbol{Z}$ to reflect no matches across datasets; that is, $\boldsymbol{Z} = (n_1 + 1, \dots, n_1 + n_2)$. We then sample $\boldsymbol{m}$ and $\boldsymbol{u}$ from their full conditionals:

$$\boldsymbol{m}_f | \Gamma, \boldsymbol{Z} \sim \text{Dirichlet}(\alpha_{f1}(\boldsymbol{Z}), \dots, \alpha_{fL_f}(\boldsymbol{Z})),$$

$$\boldsymbol{u}_f | \Gamma, \boldsymbol{Z} \sim \text{Dirichlet}(\beta_{f1}(\boldsymbol{Z}), \dots, \beta_{fL_f}(\boldsymbol{Z})),$$

where $\alpha_{fl}(\boldsymbol{Z}) = \alpha_{fl} + \sum_{i,j} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(z_j = i)$, and $\beta_{fl}(\boldsymbol{Z}) = \beta_{fl} + \sum_{i,j} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(z_j \neq i)$.

Next, we sample $\boldsymbol{Z}$ componentwise from the full conditionals for each Z_j:

$$p\left(Z_j^{(s+1)} = i | \Gamma, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{Z}^{(s)}\right) \propto \begin{cases} w_{z_j,j}, & z_j \leq n_1; \\ n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi}{n_{12}(\boldsymbol{Z}) + \alpha_\pi}, & z_j = n_1 + j, \end{cases} \tag{8}$$

where

$$w_{ij} = \prod_{f=1}^{F} \prod_{l=1}^{L_f} \left(\frac{m_{fl}}{u_{fl}} \right)^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)}.$$

Derivations for these full conditionals can be found in Appendix 7.2.

4 Efficient and Scalable Implementation

The scale of linkage tasks possible through `BRL` is limited by (1) the memory costs of storing $n_1 \times n_2$ comparison vectors for every pair of records across the two datafiles, and (2) the speed of the linkage algorithm over those comparison vectors. One common solution to these challenges is to utilize blocking, which places similar records into partitions, or "blocks," to reduce this computational burden (Christen, 2019). In deterministic blocking, the modeler chooses a field thought to be highly reliable, and only compares records that agree on that field. The record linkage method is then applied independently across all blocks, which can be done in parallel for additional speed gains. DELETE? However, blocking on an unreliable field can lead to missed matches, making this form of blocking undesirable in some situations (Steorts et al., 2014).

Another technique is indexing, in which the modeller decides a priori certain criteria that all linked pairs must satisfy, and labels any record pairs that do not meet that criteria as non-links. For example, one might only consider pairs with a certain similarity score on a field deemed to be important, like first name, or only pairs that exactly match on a specified number of fields. DELETE? However, this kind of indexing requires knowledge of the application and invites room for human error (Murray, 2016).

With `fabl`, we introduce two techniques further expand the scalability of probabilistic record linkage in addition to blocking and indexing. First, we propose hashing methods that allow us to compute sufficient statistics which reduce the computational complexity of the Gibbs sampler. Second, we introduce storage efficient indexing, which reduces the memory costs associated with unlikely matches.

4.1 Data Representation, Hashing, and Storage

Following insight from Enamorado et al. (2019), we recognize that record pairs contribute to posterior calculations only through the agreement pattern γ_{ij}. To make this more precise, let $h_1, \ldots, h_P$ denote the unique agreement patterns, and collect these unique patterns in the set $\mathcal{P} = \{h_1, \ldots, h_P\}$. Here, $P = |\mathcal{P}|$ is the total number of unique agreement patterns. P is bounded above by $\prod_{f=1}^{F} L_f$ which does not depend on n_1 or n_2. In this context, the integers $\{1, \ldots, P\}$ serve as hashed values that encode the same information as the original vectors themselves. Whenever possible, we conduct calculations over these P agreement patterns, instead of the typical $n_1 \times n_2$ record pairs. Additionally, we store "one-hot encodings" of these patterns rather than the original γ_{ij} to aid in vectorized computations. See Appendix 7.3 for details.

First, we hash record pairs of the same agreement pattern to unique integer values. Enamorado et al. (2019) accomplished this efficiently through the hashing function

$$h^*(i,j) = \sum_{f=1}^{F} I(\gamma_{ij}^f > 0) 2^{\gamma_{ij}^f + I(k>1) \sum_{e=1}^{k-1} (L_e - 1)}.$$

This function maps each agreement pattern to a unique integer, allowing us to store a scalar quantity instead of an entire vector for each record pair. For computational ease, we then map these integers to sequential integers from $\{1, \ldots, P\}$ corresponding to the enumerated patterns in $\mathcal{P}$. When the (i,j) pair exhibits the p^{th} pattern, we say $(i,j) \in h_p$. When j has no match in $\boldsymbol{X}_1$, we write $(n_1 + j, j) \in h_{P+1}$.

With all record pairs converted to hashed values, we identify the records in $\boldsymbol{X}_1$ with comparison vectors corresponding to each pattern p for each record $j \in \boldsymbol{X}_2$. We denote this set $r_{j_p} = \{i \in \boldsymbol{X}_1 | (i,j) \in h_p\}$, and collect all such sets in the nested list $\mathcal{R} = \{r_{j_p} | j \in \boldsymbol{X}_1, p \in 1, \ldots, P\}$.This representation is useful because it allows us to easily compute two sets of sufficient statistics. First, we compute the number of records in $\boldsymbol{X}_1$ that share agreement pattern p with record $j \in \boldsymbol{X}_2$, given by

$$N_{j_p} = |r_{j_p}| = \sum_{i=1}^{n_1} I((i,j) \in h_p).$$

Second, we compute the number of comparison vectors in Γ that exhibit pattern p, given by

$$H_p = \sum_{j=1}^{n_2} N_{j_p}.$$

We collect these sufficient statistics in the sets $\mathcal{N} = \{N_{j_p} | j \in \boldsymbol{X}_1, p \in 1, \ldots, P\}$ and $\mathcal{H} = \{H_p | p \in 1, \ldots, P\}$. As we will show in Section 4.3, all posterior calculations are conducted with these sufficient statistics. Note that $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{H}$ fully characterize the comparison matrix Γ with no loss of information, so we can use $\tilde{\Gamma} = \{\mathcal{P}, \mathcal{R}, \mathcal{N}, \mathcal{H}\}$ for posterior inference.

4.2 Scaling to Large Linkage Tasks

With storage efficient indexing (SEI), we can calculate $\mathcal{N}$ and $\mathcal{H}$ for all $n_1 \times n_2$ record pairs, but greatly reduce the memory costs associated with unlikely matches, allowing us to conduct all-to-all comparisons for substantially larger linkage tasks. Observe that all records $i \in \boldsymbol{X}_1$ that share agreement pattern p with record $j \in \boldsymbol{X}_2$ have the same w_{ij}. Therefore, these records have the same probability to be identified as a link when sampling Z_j. Thus, we know that records $i \in r_{j_p}$ such that N_{j_p} is large are very unlikely to be sampled consistently enough to be deemed a match through the Bayes estimate. We know this regardless of the form of the agreement pattern itself, or its associated probabilities. Therefore, rather than store all of these unlikely record labels, we choose to store only a small number S of them in a new nested list $\mathcal{R}^{\text{SEI}}$. Instead of storing $n_1 \times n_2$ record labels, SEI allows us to store at most $n_2 \times P \times S$ labels, regardless of how large n_1 might be. In practice, we recommend $S = 10$, as this reduces the number of stored indices for highly unlikely record pairings, but does is not likely to eliminate any of the indices for plausible matches. Choosing S too low, like $S = 1$ or $S = 2$, can concentrate undue mass on unlikely matches and distort linkage results. Posterior calculations still attribute the appropriate weight to all records through the summary statistics in $\mathcal{H}$, and thus we can proceed with posterior inference through the memory reduced $\tilde{\Gamma}^{\text{SEI}} = \{\mathcal{P}, \mathcal{R}^{\text{SEI}}, \mathcal{N}, \mathcal{H}\}$.

For datafiles with large amounts of records, we can partition the two datasets $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ into smaller chunks $\{\boldsymbol{X}_{1m}\}$ and $\{\boldsymbol{X}_{2n}\}$ for more manageable computations. On a single machine, we can read-in data and construct comparison vectors sequentially. Then we can conduct hashing, compress information through SEI, and delete those comparison vectors from memory before continuing with the next chunk of data. With multiple cores or multiple machines, this can be done in parallel. Summary statistics from each pairwise chunk comparison can be easily synthesized to recover sufficient statistics for the Γ. Thus, the combination of partitioning, hashing, and SEI allows us to conduct linkage tasks over much larger datasets.

4.3 Efficient Posterior Inference

Updating $\alpha_{fl}(\boldsymbol{Z})$ and $\beta_{fl}(\boldsymbol{Z})$ for each field and level in the linkage task constitutes $2 \times \sum_f L_f$ many summations over $n_1 \times n_2$ quantities. These are simple calculations, but become computationally burdensome when working on large linkage tasks.

Instead, we use one-hot encodings of the agreement patterns $\mathcal{P}$ for more efficient calculations. Denote $H_p^m = \sum_{j=1}^{n_2} I\left((Z_j, j) \in h_p, Z_j \leq n_1\right)$ to be the number of matching record pairs with agreement pattern p. It follows that the number of non-matching record pairs with agreement pattern p is $H_p^u = H_p - H_p^m$. Then, if $\boldsymbol{\alpha_0}$ and $\boldsymbol{\beta_0}$ are vectors of length $\sum_f L_f$ of prior parameters for the $\boldsymbol{m}$ and $\boldsymbol{u}$ distributions respectively, the posterior update becomes

$$\boldsymbol{\alpha}(\boldsymbol{Z}) = \boldsymbol{\alpha_0} + \sum_{p=1}^{P} H_p^m \times h_p,$$

Symbol	Description
h_p	one hot encoding of agreement pattern p
$(i, j) \in h_p$	comparison vector between records $i \in \boldsymbol{X}_1$ and $i \in \boldsymbol{X}_2$ exhibits pattern p
r_{j_p}	list of records in $\boldsymbol{X}_1$ that share agreement pattern p with record $j \in \boldsymbol{X}_2$
N_{j_p}	number of records in $\boldsymbol{X}_1$ that share agreement pattern p with record $j \in \boldsymbol{X}_2$
H_p	number of total comparison vectors that exhibit pattern p
H_p^m	number of matching comparison vectors that exhibit pattern p
H_p^u	number of non-matching comparison vectors that exhibit pattern p

Table 2: Summary of hashing notation.

$$\boldsymbol{\beta}(\boldsymbol{Z}) = \boldsymbol{\beta_0} + \sum_{p=1}^{P} H_p^u \times h_p.$$

Although sampling Z_j from the full conditional provided in (8) is conceptually straightforward, it becomes computationally burdensome when n_1 is large. This is because sampling a value from n_1 options with unequal weights requires normalizing the weights to probabilities, which has a computational cost that scales with n_1. To speed up computation, we break this sampling step into two. First, we calculate the Fellegi Sunter weight w_p associated with each unique pattern and sample the agreement pattern between j and its potential link, according to (9).

$$P\left(\left(Z_j^{(s+1)}, j\right) \in h_p \mid \tilde{\Gamma}, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{Z^{(s)}}\right) \propto \begin{cases} w_p \times N_{j_p}, & p \leq P; \\ n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi}{n_{12}(\boldsymbol{Z}) + \alpha_\pi}, & p = P + 1. \end{cases} \quad (9)$$

Since all posterior updates are governed by the agreement patterns of the record pairs rather than the record labels themselves, we complete the entire Gibbs sampler first at the level of the P agreement patterns. Since all records in $\boldsymbol{X}_1$ sharing the same agreement pattern with $j \in \boldsymbol{X}_2$ are equally likely, we can sample among candidate records uniformally through (10).

$$P\left(Z_j^{(s+1)} = i \mid \left(Z_j^{(s+1)}, j\right) \in h_p\right) = \begin{cases} \frac{1}{N_{j_p}}, & (z_j, j) \in h_p; \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

We emphasize the computational gains of this split sampler: the first step is a sample from $P + 1$ options, where P does not scale with the size of the linkage task; and the second step is sampling uniformly at random, which is computationally simple even for large sets of candidate records. These changes greatly improve the speed of the sampler, and each can be parallelized if desired for additional computational gains. We provide a summary of the notation used for the hashing procedure in Table 2. The computational complexity of `fabl` is given in Lemma 1.

Lemma 1. *Recall that n_1 and n_2 are the number of records in $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, respectively. Let F be the number of fields used for comparisons across records, and P be the number* *of patterns that comparison vectors can exhibit. We assume B processors available for parallelization. Then, the overall computational complexity of* `fabl` *is* $O(\frac{F}{B}n_1n_2) + O(n_2P)$.

Proof. We consider two steps: constructing the comparison vectors and the Gibbs sampler. The computational complexity of all pairwise comparisons across $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ is $O(Fn_1n_2)$. The hashing procedure for all pairwise comparisons is also $O(Fn_1n_2)$. With B processors available, we can split these computations across B equally sized partitions and compute these comparisons in parallel, so the complexity becomes $O(\frac{F}{B}n_1n_2)$. There are then trivial computational costs associated with synthesizing summary statistics across these partitions.

Without hashing, the computational complexity of updating the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters is $O(Fn_1n_2)$. However, by doing calculations over the agreement patterns rather than the individual records, hashing reduces the overall complexity to $O(P)$. The complexity of updating $\boldsymbol{Z}$ sequentially at the record level is $O(n_1n_2)$. With hashing, we split this sampling into two steps. First, we sample the agreement pattern of the match with complexity $O(n_2P)$, and then we sample the record exhibiting that pattern with complexity $O(n_2)$. Thus, the complexity of sampling $\boldsymbol{Z}$ is $O(n_2P)$. Since $P << n_1$ in most applications, we have reduced the complexity of the Gibbs sampler from $O(Fn_1n_2)$ under `BRL` to $O(n_2P)$ under `fabl`. In summary, the total computational complexity is $O(\frac{F}{B}n_1n_2) + O(n_2P)$. □

5 Simulation Studies

We demonstrate the speed and accuracy of `fabl` as compared to `BRL` through several simulation studies.

5.1 Speed

In our first simulation, we generate comparison vectors from pre-specified distributions so that we can easily increase the size of the linkage problem. We use $F = 5$ binary comparisons with probabilities for matching and non-matching pairs shown in Table 3. For each record in $\boldsymbol{X}_2$, we simulate n_1 comparison vectors, resulting in a comparison matrix $\Gamma \in \mathbb{R}^{n_1n_2 \times F}$. For $n_2/2$ of these records, there is no match in $\boldsymbol{X}_1$, so we simulate n_1 comparison vectors from the $\boldsymbol{u}$ distribution. For the other $n_2/2$ of these records, there is one match in $\boldsymbol{X}_1$, so we simulate 1 comparison vector from the $\boldsymbol{m}$ distribution, and $n_1 - 1$ comparison vectors from the $\boldsymbol{u}$ distribution. We compare the run-time of `fabl` against `BRL` as we increase n_1 and n_2. Note that since we have five binary comparison fields, the number is unique patterns P is bounded above by $2^5 = 32$, a bound which is consistently attained in simulations with more records.

We note here that the implementation of `BRL` that we use is coded in C (Sadinle, 2017). In contrast, we use non-optimized code written in `R` for `fabl`. While this complicates comparisons, and indeed disfavors `fabl`, the computational speed gains for `fabl` are

	m		u	
	Agree	Disagree	Agree	Disagree
First Name	$\frac{19}{20}$	$\frac{1}{20}$	$\frac{1}{100}$	$\frac{99}{100}$
Last Name	$\frac{19}{20}$	$\frac{1}{20}$	$\frac{1}{100}$	$\frac{99}{100}$
Day	$\frac{19}{20}$	$\frac{1}{20}$	$\frac{1}{30}$	$\frac{29}{30}$
Month	$\frac{19}{20}$	$\frac{1}{20}$	$\frac{1}{12}$	$\frac{11}{12}$
Year	$\frac{19}{20}$	$\frac{1}{20}$	$\frac{1}{12}$	$\frac{11}{12}$

Table 3: Probabilities used for m and u distributions in simulation study in Section 5.1.

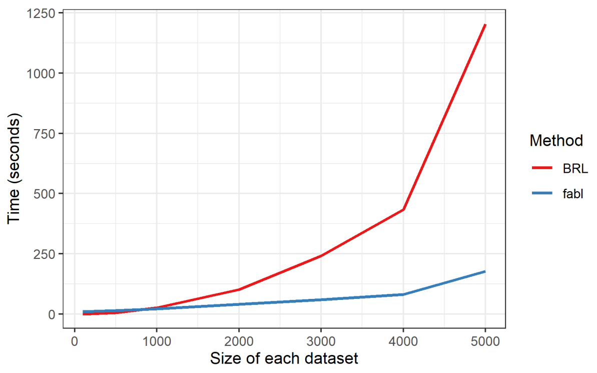

Figure 1: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including the hashing step for `fabl`, for increasing values of both n_1 and n_2, as described in Section 5.1. We see near quadratic growth in run-time for `BRL`, and near linear growth for `fabl`.

still evident, especially for larger sample sizes. Additionally, although `fabl` is amenable to parallelization, this simulation was run on a single core. Implementing `fabl` in C++ with paralellization for the hashing step and sampling the matching status of the record pairs should lead to even more computational gains.

In Figure 1, where we increase both n_1 and n_2, `BRL` is faster than `fabl` for low sample sizes, but `fabl` is significantly faster at handling larger data. In particular, run-time for `BRL` grows quadratically (or linearly with the size of both $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$) while run-time for `fabl` grows linearly (in the size of only $\boldsymbol{X}_2$).

In Figure 2, where we fix $n_2 = 500$, we see linear growth for the run-time under `BRL` as n_1 increases, and much more static run-time under `fabl`. The slight increases in run-time for `fabl` are due primarily to the hashing step, which again can be run in parallel for large data. To illustrate that these trends are generalizeable to other specifications of the comparison vectors, we have included the run-time results for an additional simulation study, under different comparison vector settings, in Appendix 7.6.

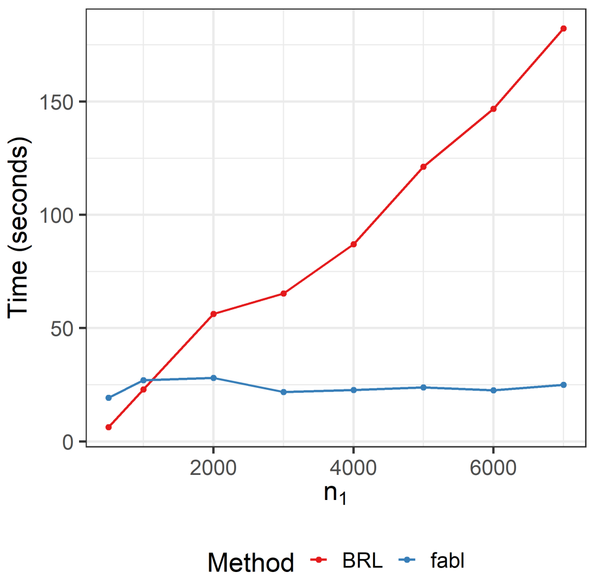

Figure 2: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including hashing step for `fabl`, with n_2 fixed at 500, as described in Section 5.1. We see linear growth in run-time for `BRL`, and near constant run-time for `fabl`.

5.2 Accuracy under Full Estimates

We replicate a simulation study from Sadinle (2017) used to illustrate the effectiveness of `BRL`. We adapt it to compare the accuracy of `fabl` to `BRL`. The simulations employ a collection of synthetic synthetic datasets with varying amounts of error and overlap (the number of records in common across files). Following methods proposed by Christen and Pudjijono (2009) and Christen and Vatsalan (2013), clean records are first simulated from frequency tables for first name, last name, age, and occupation in Australia. Fields are then chosen for distortion uniformly at random. Names are subject to string insertions, deletions and substitutions, as well as common keyboard, phonetic, and optical recognition errors. Age and occupation are distorted through keyboard errors and missingness.

We create comparison vectors according to the default settings of the `compareRecords` function from the `BRL` package, shown in Table 4. Each simulation identifies matched individuals between two datasets, each with 500 records. We conduct linkage when matching records exhibit 1, 2, and 3 errors across the four fields, and when there are 50, 250, and 450 individuals in common across datasets. Under each of these settings, we have 100 pairs of simulated datasets in order to obtain uncertainty quantification on our performance metrics. We use flat priors for all m and u parameters, run the Gibbs Sampler for 1000 iterations, and discard the first 100 as burn-in. We calculate Bayes

		Level of Disagreement			
Fields	Similarity	1	2	3	4
First and Last Name	Levenstein	0	(0, .25]	(.25, .5]	(.5, .1]
Age and Occupation	Binary	Agree	Disagree		

Table 4: Construction of comparison vectors for accuracy study with simulated datasets of Section 5.2.

estimates of the linkage structure using the losses $\theta_R = \infty, \theta_{10} = 1, \theta_{01} = 1, \theta_{11} = 2$. Traceplots for parameters of interest for one example simulation are provided in Appendix 7.5; they show no obvious concern over MCMC convergence. We replicate this simulation under partial estimates, allowing the `fabl` to leave some components of the linkage structure undetermined and left for clerical review, in Appendix 7.4.

We compare `fabl` to `BRL` in terms of recall, precision and F-measure, as defined in Christen (2012). Recall is the proportion of true matches found by the model, that is, $\sum_{j=1}^{n_2} I(\hat{Z}_j = Z_j, Z_j \leq n_1)/\sum_{j=1}^{n_2} I(Z_j \leq n_1)$. Precision is the proportion of links found by the model that are true matches, that is, $\sum_{j=1}^{n_2} I(\hat{Z}_j = Z_j, Z_j \leq n_1)/\sum_{j=1}^{n_2} I(\hat{Z}_j \leq n_1)$. The F-measure balances the two metrics to provide an overall measure of accuracy, and is defined as $2 \times (\text{Recall} + \text{Precision})/(\text{Recall} \times \text{Precision})$. In Figure 3, we see that the two methods have comparable performance at all levels of error and overlap. In the specific case of high error and low overlap, widely regarded as the most difficult linkage scenario, we see that `fabl` performs slightly worse on average; however, the overall accuracy level remains high.

6 Case Studies

In our first case study, we revisit data from the El Salvadoran Civil War analyzed by Sadinle (2017). Though the data files used in this case study are small, it shows how the computational complexity of `fabl` depends on the number of unique agreement patterns found in the data, and how significant computational gains can be achieved by simplifying the construction of the comparison vectors. In the second case study, we apply `fabl` to link records from the National Long Term Care Study (NLTCS), a larger linkage task that is not feasible in reasonable time under `BRL` with typical computing setups.

6.1 Civilian Casualties from the El Salvadoran Civil War

The country of El Salvador was immersed in civil war from 1980 to 1991, and we are interested in estimating the total number of casualties from the war. We utilize lists of casualties from the war, one collected by El Rescate - Tutela Regal (ERTL) and another from the Salvadoran Human Rights Commission (CDHES, by its acronym in Spanish).[1] The ERTL dataset consists of digitized denunciations that had been published throughout the conflict, and the CDHES dataset consists of casualties that had been

[1]We thank the Human Rights Data Analysis Group (HRDAG) for granting access to this data.

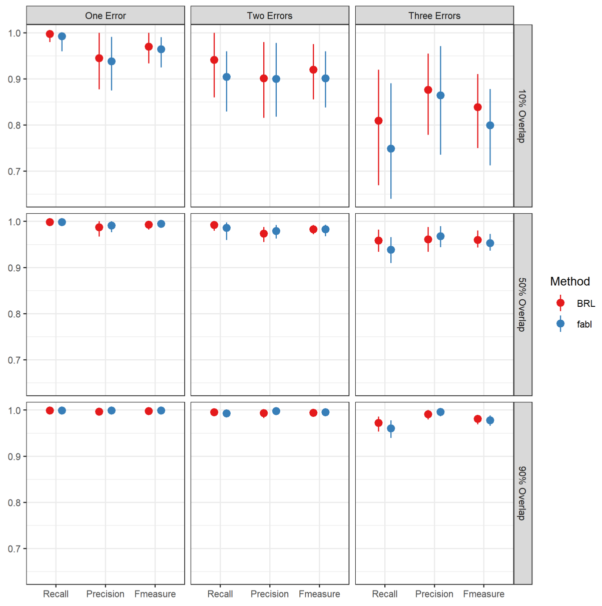

Figure 3: Posterior means and credible intervals for accuracy metrics under the replication of simulation study from Sadinle (2017). For each level of overlap and each level of error, we have 100 paired sets of 500 records. Thus this table represents results for 900 datasets. We see comparable performance for all levels of error and overlap.

reported directly to the organization (Howland, 2008; Ball, 2000). The ERTL required additional investigation before recording denunciations as human rights abuses, and reports to the CHDES were made shortly after the events occurred; thus, both datasets are thought to be fairly reliable. When estimating the total number of casualties, one cannot simply sum the numbers recorded by each organization, as it is likely that the same individuals are recorded in multiple casualty lists. Instead, record linkage techniques must be used to merge datasets before analyzing the data (Lum et al., 2013).

There are several challenges with these data. First, both datasets have been automatically digitized, which inherently leads to some degree of typographical error. Second, the only fields recorded are given name, last name, date of death, and place of death. It is relatively common for a parent and child to share the same given name, resulting in indistinguishable records for two different individuals.

Following Sadinle (2017), we utilize records that have non-missing entries for given and last name, which results in $n_1 = 4420$ records in CHDES and $n_2 = 1323$ records in ERTL. We standardize names to account for common misspellings and use a modified Levenstein distance when comparing names to account for the fact that second names are often omitted in Spanish. Place of death is recorded by municipality and department within that municipality; however, since department is missing in 95% of records in

		Level of Disagreement			
Fields	Similarity	1	2	3	4
First and Last Name	Modified Levenstein	0	(0, .25]	(.25, .5]	(.5, 1]
Year of Death	Absolute Difference	0	1	2	3+
Month of Death	Absolute Difference	0	1	2-3	4+
Day of Death	Absolute Difference	0	1-2	3-7	8+
Municipality	Binary	Agree	Disagree		

Table 5: Construction of comparison vectors for El Salvador data resembling original implementation from (Sadinle, 2017). This set up leads to 2048 possible agreement patterns in total.

		Level of Disagreement		
Fields	Similarity	1	2	3
First and Last Name	Modified Levenstein	0	(0, .25]	(.25, 1]
Year of Death	Binary	Agree	Disagree	
Month of Death	Binary	Agree	Disagree	
Day of Death	Absolute Difference	0	1	2+
Municipality	Binary	Agree	Disagree	

Table 6: Construction of comparison vectors for El Salvador for increased speed under `fabl`. This set up leads to 216 possible agreement patterns in total.

CHDES and 80% of records in ERTL, we exclude department from our analysis. Thus, we conduct record linkage using given name, last name, municipality, and day, month, and year of death. We flat priors for the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters.

We initially followed the comparison vector constructions set by Sadinle (2017), using four levels of agreement for each field, according to the thresholds provided in Table 5. This results in $4^5 \times 2 = 2048$ possible agreement patterns, with 1173 patterns realized in the data. However, we noticed that the posterior distributions of several levels of the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters were nearly identical in an initial run of `BRL`, suggesting that these levels were unnecessary.

Therefore, we perform our analysis with the agreement levels for each field according to Table 6. Among the 216 possible agreement patterns, 159 are realized in the data. With this revised comparison specification, `fabl` ran in 61 seconds, approximately 4 times faster than the `BRL` run time of 239 seconds. The estimates of the $\boldsymbol{m}$ parameters under each method are similar, as shown in Figure 5. Estimates of $\boldsymbol{u}$ are indistinguishable, and thus omitted. Traceplots for parameters of interest are provided in Appendix 7.7.

For completeness, we note that linkage with the more detailed comparison vectors required 240 seconds for `BRL`, and 261 seconds for `fabl`. Apparently, the number of patterns was sufficiently many that the computational savings from `fabl` does not overcome the inherent speed differences of C as opposed to `R`.

Through `fabl`, we arrive at a Bayes estimate of 179 individuals recorded in both datasets. We calculate posterior samples of the size of the overlap across files by finding the number of matches found in each iteration of the Gibbs sampler, and subtracting

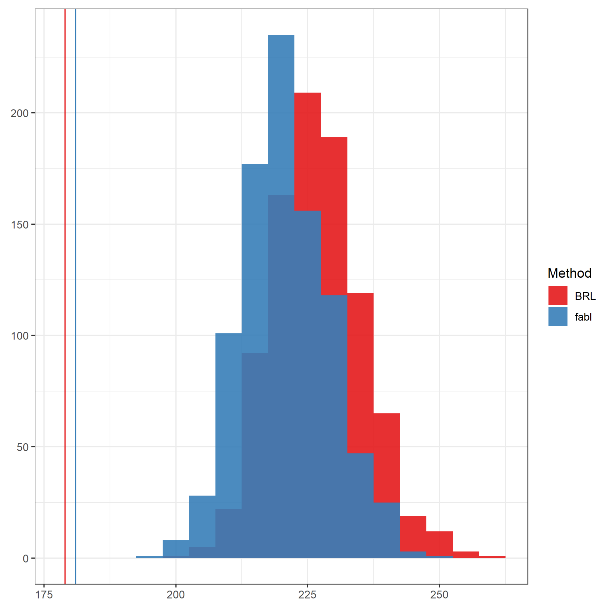

Figure 4: Posterior distribution and Bayes estimate of overlap across the two files. We note they are quite similar under both methods.

the number of matches that violate one-to-one matching. The posterior 95% credible interval for the overlap across files is (206, 238), indicating that the Bayes estimate identifies fewer matches than the Gibbs sampler identifies on average. This is because a large number of records in ERTL have multiple plausible matches in CDHES; `fabl` recognizes that a match exists among the several options, but is unable to definitely declare a specific pair as a match in the Bayes estimate. We also compute a partial estimate of the linkage structure, using $\theta_{10} = \theta_{01} = 1$, $\theta_{11} = 2$, and $\theta_R = 0.1$ as in the simulation study in Appendix 7.4. Here, the Bayes estimate provides 136 matches of which the model is quite confident, and 175 records to verify manually. This means that after clerical review, the number of individuals replicated across datasets would fall in the interval (136, 311), encapsulating the posterior credible interval. More or fewer records could be identified for clerical review by decreasing or increasing θ_R.

We see similar results under `BRL`, with a Bayes estimate of 181 individuals recorded in both datasets, a posterior 95% credible interval of (211, 244), and a range of (140, 294) after the partial estimate and clerical review. See Figure 4 for a visual comparison of the Bayes estimate and posterior credible intervals for the two methods. We note that Bayes estimates falling outside of posterior credible intervals has been observed previously in the record linkage literature (Sadinle, 2017; Steorts et al., 2016), and remains a topic for future research.

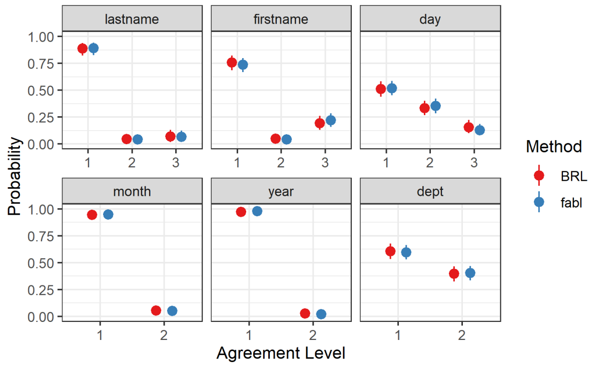

Figure 5: Posterior estimates of $\boldsymbol{m}$ parameters with 95% credible intervals for El Salvador case study. They are quite similar across the two methods.

6.2 National Long Term Care Study

The National Long Term Care Study (NLTCS) is a longitudinal study tracking the health outcomes of Medicare recipients. The initial survey began in 1982, with follow-up surveys taken approximately every five years. As such, patients are surveyed at most once in a given year, and many patients are surveyed across multiple years. In addition, patients can either drop out of the study, pass away, or enter as new patients (Steorts et al., 2016). Hence, the assumptions of our model hold for this study. We seek to link records over the $n_1 = 20485$ records from 1982 to the $n_2 = 17466$ records from 1989. The NLTCS data have longitudinal links, so that in reality one does not need to conduct record linkage. However, following the strategy in Guha et al. (2022), we break the longitudinal links and treat the data from 1982 and 1989 as stand-alone datafiles.

We link records using sex, date of birth, and location using the thresholds shown in Table 7. Storing three comparison scores for each of $20485 \times 17466 \approx 400,000,000$ record pairs would require approximately 8GB of memory. Standard settings on a 16GB personal computer do not allow storage of an object of this size, and thus `BRL` is unable to perform this linkage task on such a machine. However, through the `fabl` framework, we compute comparisons over 30 smaller comparison tasks, hash results, and compress information through storage efficient indexing. The resulting data object is just 10 MB, approximately 0.1% of what is required for the raw comparisons. Constructing the comparisons sequentially took approximately 40 minutes, which could be reduced considerably through parallel computing.

We run a Gibbs sampler for 1000 iterations, taking about 235 seconds. As shown in Figure 6, the Bayes estimate of the linkage structure consists of 9634 matches, with a 95% credible interval of (9581, 9740). Since we have access to the true linkage structure, we can calculate recall to be 0.89 and precision to be 0.98, resulting in an F-measure of 0.94. Traceplots do not suggest convergence issues, and are similar to those seen in Appendix 7.5 and 7.7

		Level of Disagreement		
Fields	Similarity	1	2	3
Sex	Binary	Agree	Disagree	
Year of Birth	Binary	Agree	Disagree	
Month of Birth	Binary	Agree	Disagree	
Day of Birth	Binary	Agree	Disagree	
Location	Custom	Same State and Office	Same State	Otherwise

Table 7: Construction of comparison vectors for NTLCS data

7 Conclusion

In this paper, we have proposed `fabl`, a Bayesian record linkage method that extends the work of Sadinle (2017) to scale to large data sets. We have proven that the proposed hashing method and model assumptions allow for a linkage procedure whose computational complexity does not scale with the size of the larger dataset. This makes `fabl` computationally advantageous in many linkage scenarios, particularly when one datafile is substantially smaller than the other. We have also shown that storage efficient indexing, in tandem with hashing, greatly reduces the memory costs required for all-to-all comparisons, giving practictioners an option for larger record linkage tasks without the use of blocking or indexing. We have demonstrated the speed and accuracy of `fabl` by replicating a simulation study and a case study in Sadinle (2017), and through an additional case study that is computationally infeasible under `BRL`.

Although the `fabl` method greatly reduces the memory costs for all-to-all comparisons, the computing all $n_1 \times n_2$ record pairs still can be infeasible for larger linkage tasks. Indeed, constructing the comparison vectors for the NLTCS linkage task involving around 40,000 records in Section 6.2 took around 40 minutes. Due to the quadratic nature of the comparison space, this computation time would grow quickly with the size of the linkage task, and would be infeasibly slow when dealing with millions of records. Although it is common to use deterministic blocking to reduce the comparison space and then apply probabilistic record linkage within each block, issues arise when sizes of blocks vary across the linkage task. Thus in future work, we seek to extend `fabl` to account for such deterministic blocking, making the framework amenable to arbitrarily large linkage tasks.

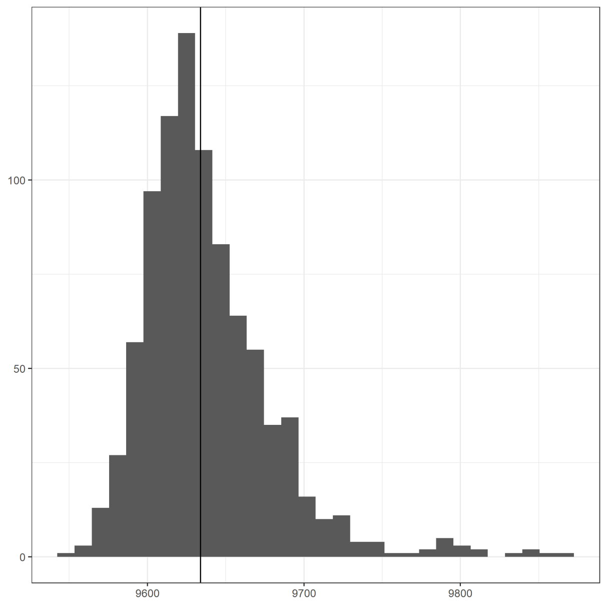

Figure 6: Posterior distribution and Bayes estimate of overlap across years 1982 and 1989 of NLTCS data.

References

Aleshin-Guendel, S. and Sadinle, M. (2022), “Multifile Partitioning for Record Linkage and Duplicate Detection,” *Journal of the American Statistical Association*, 0, 1–10. 1

Ball, P. (2000), “The Salvadoran Human Rights Commission: Data Processing, Data Representation, and Generating Analytical Reports,” in *Making the Case: Investigating Large Scale Human Rights Violations Using Information Systems and Data Analysis*, eds. P. Ball, H. F. Spirer, and L. Spirer, pp. 15–24, American Association for the Advancement of Science. 15

Betancourt, B., Sosa, J., and Rodríguez, A. (2022), “A Prior for Record Linkage Based on Allelic Partitions,” *Computational Statistics and Data Analysis*, 172, 107474. 1

Bilenko, M. and Mooney, R. (2006), “Riddle: Repository of Information on Duplicate Detection, Record Linkage, and Identity Uncertainty,” Online; retrieved July 29, 2020. 3

Christen, P. (2012), “A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication,” *IEEE Transactions on Knowledge and Data Engineering*, 24, 1537–1555. 1, 14

Christen, P. (2019), “Data Linkage: The Big Picture,” *Harvard Data Science Review*, 1, https://hdsr.mitpress.mit.edu/pub/8fm8lo1e. 7

Christen, P. and Pudjijono, A. (2009), “Accurate Synthetic Generation of Realistic Personal Information,” in *Advances in Knowledge Discovery and Data Mining*, eds. T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, pp. 507–514, Berlin, Heidelberg, Springer Berlin Heidelberg. 13

Christen, P. and Vatsalan, D. (2013), “Flexible and Extensible Generation and Corruption of Personal Data,” in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM ’13, p. 1165–1168, New York, NY, USA, Association for Computing Machinery. 13

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003), “A Comparison of String Distance Metrics for Name-Matching Tasks,” in *Proceedings of the 2003 International Conference on Information Integration on the Web*, p. 73–78, AAAI Press. 3

Dalzell, N. M. and Reiter, J. P. (2018), “Regression Modeling and File Matching Using Possibly Erroneous Matching Variables,” *Journal of Computational and Graphical Statistics*, 0, 1–11. 1

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007), “Duplicate Record Detection: A Survey,” *IEEE Transactions on Knowledge and Data Engineering*, 19, 1–16. 3

Enamorado, T., Fifield, B., and Imai, K. (2019), “Using a Probabilistic Model to Assist Merging of Large-Scale Administrative Records,” *American Political Science Review*, 113, 353–371. 1, 8

Fair, M. (2004), “Generalized Record Linkage System–Statistics Canada’s Record Linkage Software,” *Austrian Journal of Statistics*, 33, 37–53. 1

Fellegi, I. and Sunter, A. (1969a), "A Theory for Record Linkage," *Journal of the American Statistical Association*, 64, 1183–1210. 4

Fellegi, I. P. and Sunter, A. B. (1969b), "A Theory for Record Linkage," *Journal of the American Statistical Association*, 64, 1183–1210. 1, 2, 3

Fortunato, S. (2010), "Community Detection in Graphs," *Physics Reports*, 486, 75–174. 2

Gill, L. and Goldacre, M. (2003), "English National Record Linkage of Hospital Episode Statistics and Death Registration Records," *Report to the Department of Health*. 1

Guha, S., Reiter, J. P., and Mercatanti, A. (2022), "Bayesian Causal Inference with Bipartite Record Linkage," *Bayesian Analysis*, p. in press. 18

Gutman, R., Afendulis, C. C., and Zaslavsky, A. M. (2013), "A Bayesian Procedure for File Linking to Analyze End-of-Life Medical Costs," *Journal of the American Statistical Association*, 108, 34–47. 1, 2

Howland, T. (2008), "How El Rescate, a Small Nongovernmental Organization, Contributed to the Transformation of the Human Rights Situation in El Salvador," *Human Rights Quarterly*, 30, 703–757. 15

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. 2, 4

Larsen, M. D. (2005), "Advances in Record Linkage Theory: Hierarchical Bayesian Record Linkage Theory," in *Proceedings of the Joint Statistical Meetings, Section on Survey Research Methods*, pp. 3277–3284, The American Statistical Association. 2

Little, R. and Rubin, D. (2002), *Statistical Analysis with Missing Data*, Wiley, Hoboken, New Jersey. 4

Lum, K., Price, M. E., and Banks, D. (2013), "Applications of Multiple Systems Estimation in Human Rights Research," *The American Statistician*, 67, 191–200. 15

Marchant, N. G., Kaplan, A., Elazar, D. N., Rubinstein, B. I. P., and Steorts, R. C. (2021), "d-blink: Distributed End-to-End Bayesian Entity Resolution," *Journal of Computational and Graphical Statistics*, 30, 406–421. 1

Murray, J. S. (2016), "Probabilistic Record Linkage and Deduplication after Indexing, Blocking, and Filtering," *Journal of Privacy and Confidentiality*, 7, 3–24. 7

Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. (1959), "Automatic Linkage of Vital Records," *Science*, 130, 954–959. 1

Sadinle, M. (2017), "Bayesian Estimation of Bipartite Matchings for Record Linkage," *Journal of the American Statistical Association*, 112, 600–612. 1, 2, 3, 4, 5, 6, 11, 13, 14, 15, 16, 17, 19, 24, 26

Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014), "A Comparison of Blocking Methods for Record Linkage," in *Privacy in Statistical Databases*, Lecture Notes in Computer Science, pp. 253–268, Springer, Cham. 7

Steorts, R. C., Hall, R., and Fienberg, S. E. (2016), "A Bayesian Approach to Graphical Record Linkage and Deduplication," *Journal of the American Statistical Association*, 111, 1660–1672. 1, 6, 17, 18

Tancredi, A. and Liseo, B. (2011), "A Hierarchical Bayesian Approach to Record Linkage and Size Population Problems," *Annals of Applied Statistics*, 5, 1553–1585. 2

Tang, J., Reiter, J. P., and Steorts, R. C. (2020), "Bayesian Modeling for Simultaneous Regression and Record Linkage," in *Privacy in Statistical Databases*, eds. J. Domingo-Ferrer and K. Muralidhar, pp. 209–223, Cham, Springer International Publishing. 1

Wagner, D., Lane, M., et al. (2014), "The Person Identification Validation System (PVS): Applying the Center for Administrative Records Research and Applications' (CARRA) Record Linkage Software," Tech. rep., Center for Economic Studies, U.S. Census Bureau. 1

Winkler, W. and Thibaudeau, Y. (1990), "An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 US Decennial Census," *U.S. Census Research Report*, pp. 1–22. 1

Wortman, J. P. H. (2019), "Record Linkage Methods with Applications to Causal Inference and Election Voting Data," Ph.D. thesis, Duke University. 2, 5

Appendix

7.1 Bayes Estimate

We calculate a Bayes estimate $\hat{\boldsymbol{Z}}$ for the linkage parameter $\boldsymbol{Z}$ by assigning different positive losses to different types of errors, and minimizing posterior expected loss. We adopt the loss function proposed in Sadinle (2017) in which $\hat{Z}_j \in \{1, \dots, n_1, n_1 + j, R\}$, with R representing the option to leave the matching undetermined by the model. Specifically, we have

$$L(\hat{Z}_j, Z_j) = \begin{cases} 0, & \text{if } Z_j = \hat{Z}_j; \\ \theta_R, & \text{if } \hat{Z}_j = R; \\ \theta_{10}, & \text{if } Z_j \leq 1, \hat{Z}_j = n_1 + j; \\ \theta_{01}, & \text{if } Z_j = n_1 + j, \hat{Z}_j \leq n_1; \\ \theta_{11}, & \text{if } Z_j \leq n_1, \hat{Z}_j \leq n_1, Z_j \neq \hat{Z}_j. \end{cases}$$

Here, θ_R is the loss from not making a decision on the linkage status, θ_{10} is the loss from a false non-match, θ_{01} is the loss from a false match, and θ_{11} is the loss from the special case of a false match in which the record has a true match other than the one estimated by the model.

In general, we set $(\theta_{10}, \theta_{01}, \theta_{11}, \theta_R) = (1, 1, 2, \infty)$ inducing the decision rule

$$\hat{Z}_j = \begin{cases} i, & \text{if } P(Z_j = i|\Gamma) > \frac{1}{2}; \\ 0, & \text{otherwise.} \end{cases}$$

When we seek a partial estimate of the linkage structure, leaving a portion of record pairs to be classified manually in clerical review, we adopt losses $(\theta_{10}, \theta_{01}, \theta_{11}, \theta_R) = (1, 1, 2, .1)$. For a more in-depth explanation of this function and the induced Bayes estimate, see Sadinle (2017).

7.2 Derivations of Full Conditionals

We provide detailed derivations of the full-conditionals provided in Section 3.1. The $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters are updated through standard multinomial-Dirichlet distributions. For a particular m_{fl}, we have

$$\begin{aligned} p(m_{fl}|\Gamma, \mathbf{Z}) &\propto \prod_{i=1}^{n_1} \prod_{j=1}^{n_2} m_{fl}^{I(Z_j = i) I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)} \times m_{fl}^{\alpha_{fl} - 1} \\ &= m_{fl}^{\alpha_{fl}(\boldsymbol{Z}) - 1}, \end{aligned}$$

where $\alpha_{fl}(\boldsymbol{Z}) = \alpha_{fl} + \sum_{i,j} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(z_j = i)$. Analogous procedures lead to the posterior distribution $p(u_{fl}|\Gamma, \mathbf{Z}) \propto u_{fl}^{\beta_{fl}(\boldsymbol{Z}) - 1}$, where $\beta_{fl}(\boldsymbol{Z}) = \beta_{fl} + \sum_{i,j} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(z_j \neq i)$. Thus for the vectors of parameters $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$, we have

$$\boldsymbol{m}_f|\boldsymbol{Z},\Gamma \sim \text{Dirichlet}(\alpha_{f1}(\boldsymbol{Z}),\ldots,\alpha_{fL_f}(\boldsymbol{Z})),$$
$$\boldsymbol{u}_f|\boldsymbol{Z},\Gamma \sim \text{Dirichlet}(\beta_{f1}(\boldsymbol{Z}),\ldots,\beta_{fL_f}(\boldsymbol{Z})).$$

In order to define the full conditional for $\boldsymbol{Z}$, we must provide the posterior distribution for π, and provide a more tractable expression for the portion of the likelihood relevant to a particular record $j \in \boldsymbol{X}_2$. Since π encodes the rate of matching across the two datasets, the posterior distribution $p(\pi|\Gamma,\boldsymbol{Z},\boldsymbol{m},\boldsymbol{u},\alpha_\pi,\beta_\pi)$ depends only on the number of links $n_{12}(\boldsymbol{Z}) = \sum_{i=1}^{n_2} I(z_j < n_1 + j)$ encoded by $\boldsymbol{Z}$ (and hyperparameters). Thus, we use $p(\pi|\mathbf{Z},\alpha_\pi,\beta_\pi)$ and have

$$\begin{aligned}
p(\pi|\boldsymbol{Z},\alpha_\pi,\beta_\pi) &\propto p(\boldsymbol{Z}|\pi)p(\pi)\\
&\propto \pi^{n_{12}(\boldsymbol{Z})}(1-\pi)^{n_2-n_{12}(\boldsymbol{Z})}\pi^{\alpha_\pi-1}(1-\pi)^{\beta_\pi-1}\\
&\propto \pi^{n_{12}(\boldsymbol{Z})+\alpha_\pi-1}(1-\pi)^{n_1-n_{12}(\boldsymbol{Z})+\beta_\pi-1}.
\end{aligned}$$

584 Thus $\pi^{(s+1)}|\boldsymbol{Z}^{(s+1),\alpha_\pi,\beta_\pi}$ has a Beta$(n_{12}(\boldsymbol{Z})+\alpha_\pi, n_2 - n_{12}(\boldsymbol{Z})+\beta_\pi)$ distribution.

Let $\Gamma_{.j}$ denote the set of n_1 comparison vectors with $j \in \boldsymbol{X_2}$. We have

$$\begin{aligned}
p(\Gamma_{.j}|Z_j = z_j,\boldsymbol{m},\boldsymbol{u}) &\propto \prod_{i=1}^{n_1}\left[\prod_{f=1}^{F}\prod_{l=1}^{L_f} m_{fl}^{I(Z_j=i)}u_{fl}^{I(Z_j\neq i)}\right]^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)}\\
&\propto \prod_{i=1}^{n_1}\frac{\left[\prod_{f=1}^{F}\prod_{l=1}^{L_f} m_{fl}^{I(Z_j=i)}u_{fl}^{I(Z_j\neq i)}\right]^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)}}{\prod_{f=1}^{F}\prod_{l=1}^{L_f} u_{fl}^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)}}\\
&\propto \prod_{i=1}^{n_1}\left(\prod_{f=1}^{F}\prod_{l=1}^{L_f}\frac{m_{fl}}{u_{fl}}\right)^{I(z_j=i)I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)}\\
&= \begin{cases} w_{z_j,j} & z_j \le n_1;\\ 1 & z_j = n_1 + j,\end{cases}
\end{aligned}$$

where

$$w_{ij} = \prod_{f=1}^{F}\prod_{l=1}^{L_f}\left(\frac{m_{fl}}{u_{fl}}\right)^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)}.$$

With the likelihood in this form, we derive an expression for the posterior distribution of $\boldsymbol{Z}$. Because we sample each Z_j independently of all other $Z_{j'}$, we use only the full conditional for an individual Z_j:

$$p\left(Z_j^{(s+1)}|\Gamma_{.j},\boldsymbol{m},\boldsymbol{u},\boldsymbol{Z^{(s)}},\pi\right)$$

$$\begin{aligned}
&\propto p(\Gamma_{.j}|Z_j^{(s+1)}, \boldsymbol{m}, \boldsymbol{u})P(Z_j^{(s+1)}|\pi) \\
&\propto \left(\sum_{i=1}^{n_1} w_{z_j,j} I(z_j = i) + I(z_j = n_1 + j)\right)\left(\pi \sum_{i=1}^{n_1} \frac{1}{n_1} I(z_j = i) + (1-\pi) I(z_j = n_1 + j)\right) \\
&= \frac{\pi}{n_1} \sum_{i=1}^{n_1} w_{z_j,j} I(z_j = i) + (1-\pi) I(z_j = n_1 + j). \\
&= \begin{cases} \frac{\pi}{n_1} w_{z_j,j} & z_j \leq n_1; \\ 1-\pi & z_j = n_1 + j. \end{cases}
\end{aligned}$$

For more direct comparability with the method from Sadinle (2017), we avoid sampling π directly, and instead integrate over it in the final full conditional for Z_j.

$$\begin{aligned}
&p\left(Z_j^{(s+1)} = i|\Gamma, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{Z^{(s)}}\right) \\
&\quad = \int_\pi p\left(Z_j^{(s+1)}|\Gamma, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{Z^{(s)}}, \pi\right) p\left(\pi|\boldsymbol{Z^{(s)}}\right) d\pi \\
&\quad = \int_\pi \left[\frac{\pi}{n_1} w_{z_j,j} I(z_j \leq n_1) + (1-\pi) I(z_j = n_1 + j)\right] p\left(\pi|\boldsymbol{Z^{(s)}}\right) d\pi \\
&\quad = \frac{\int_\pi \pi p\left(\pi|\boldsymbol{Z^{(s)}}\right) d\pi}{n_1} w_{z_j,j} I(z_j \leq n_1) + \left(1 - \int_\pi \pi p\left(\pi|\boldsymbol{Z^{(s)}}\right) d\pi\right) I(z_j = n_1 + j) \\
&\quad = \frac{n_{12}(\boldsymbol{Z}) + \alpha_\pi}{n_1(n_2 + \alpha_\pi + \beta_\pi)} w_{z_j,j} I(z_j \leq n_1) + \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi}{n_2 + \alpha_\pi + \beta_\pi} I(z_j = n_1 + j) \\
&\quad \propto w_{z_j,j} I(z_j \leq n_1) + n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi}{n_{12}(\boldsymbol{Z}) + \alpha_\pi} I(z_j = n_1 + j) \\
&\quad = \begin{cases} w_{z_j,j} & z_j \leq n_1; \\ n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi}{n_{12}(\boldsymbol{Z}) + \alpha_\pi} & z_j = n_1 + j. \end{cases}
\end{aligned}$$

7.3 One Hot Encoding Transformation

As described in Section 4, `fabl` makes use of one-hot encodings to aid in vectorized computations. For γ_{ij}^f with L_f levels, define e_{ij}^f to be an $L_f \times 1$ vector. When $\gamma_{ij}^f = l$, we set the l^{th} element of e_{ij}^f to be 1, and set the other $L_f - 1$ elements of e_{ij}^f to be 0. We then concatenate the e_{ij}^f for all $f \in \{1, \dots, F\}$, resulting in the one-hot encoded comparison vector e_{ij} of length $\sum_{f=1}^F L_f$.

For example, consider comparing the toy records shown in Table 8 with $L = (3, 3, 2, 2)$ levels of agreement for last name, first name, DOB, and city respectively. Since the first name differs by only one letter, a reasonable comparison vector for this pair would be $\gamma_{ij} = (1, 2, 1, 2)$. The one hot encoding representation of this vector is $e_{ij} = (1, 0, 0, 0, 1, 0, 1, 0, 0, 1)$.

Last Name	First Name	DOB	City
Smith	Taylor	01/01/2000	Durham
Smith	Tayler	01/01/2000	Raleigh

Table 8: Example records for one hot encoding.

7.4 Accuracy under Partial Estimates

By leaving $\theta_{10} = \theta_{01} = 1$ and $\theta_{11} = 2$, but setting $\theta_R = 0.1$, we allow the model to decline to decide a match for certain records, with nonassignment being 10% as costly as a false match. In this context, we are no longer focused on finding all true matches, but rather protecting against false matches. Thus, instead of recall, we use the negative predictive value (NPV), defined as the proportion of non-links that are actual non-matches. Mathematically, NPV $= \sum_{j=1}^{n_2} I(\hat{Z}_j = Z_j = n_1 + j) / \sum_{j=1}^{n_2} I(\hat{Z}_j = n_1 + j)$. We continue to use the precision, which is renamed the positive predictive value (PPV) in this context. Lastly, we also examine the rejection rate (RR), or how often the model declines to make a linkage decision, defined as RR $= \sum_{j=1}^{n_2} I(\hat{Z}_j = R)/n_2$. To convey this information alongside NPV and PPV, for which values close to 1 indicate strong performance, we report the decision rate (DR), defined as DR $= 1 - RR$.

In Figure 7, we see that `fabl` maintains equivalently strong PPV as `BRL` across all linkage settings. However, with high amounts of error, and thus fewer accurate and discerning fields of information, the rejection rate under `fabl` rises, leading to a decrease in NPV. Since `fabl` does not remove previously matched records from consideration for a new record, posterior probabilities of matches at times can be split across more records; in contrast, `BRL` is able to maintain higher confidence in matches in this setting. If one wishes to use partial estimates, `fabl` will possibly leave more linkages for the modeller to match by hand than would be left under `BRL`, but the decisions made by each method will have nearly equal accuracy.

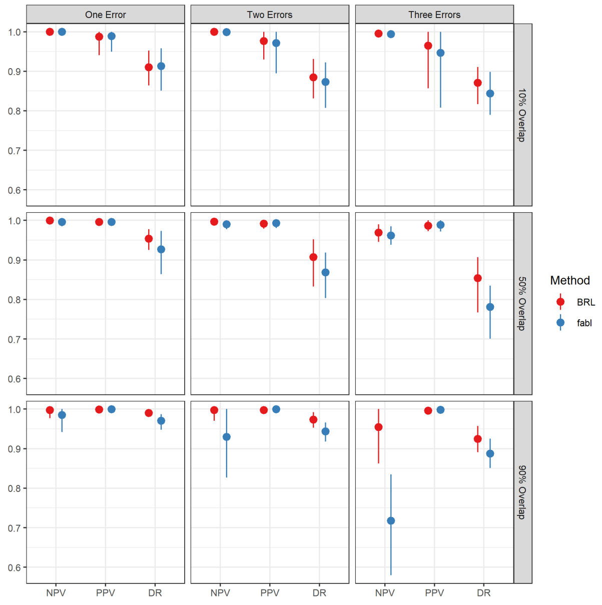

Figure 7: Negative predictive value (NPV), positive predictive value (PPV), and decision rate (DR) on simulated datasets. We see poorer performance for `fabl` only in situations with high overlap.

7.5 Traceplots for Simulation Study

Below are traceplots for one of the 900 linkage tasks that comprise the simulation in Section 5.2. It is set up with one error across the linkage fields and 50 duplicates across files. Traceplots across other settings exhibit similar behavior. Note that traceplots for $\mathbf{u}$ parameters show very little variation because the overwhelming majority of record pairs are nonmatching.

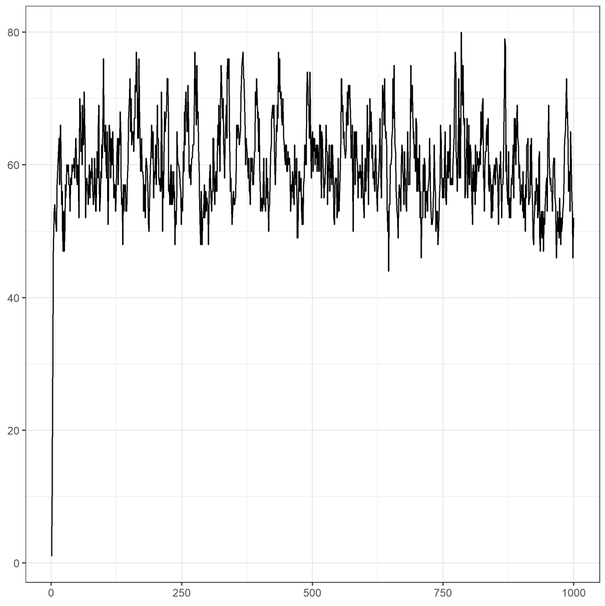

Figure 8: Representative traceplot of overlap between files from simulation study in Section 5.2

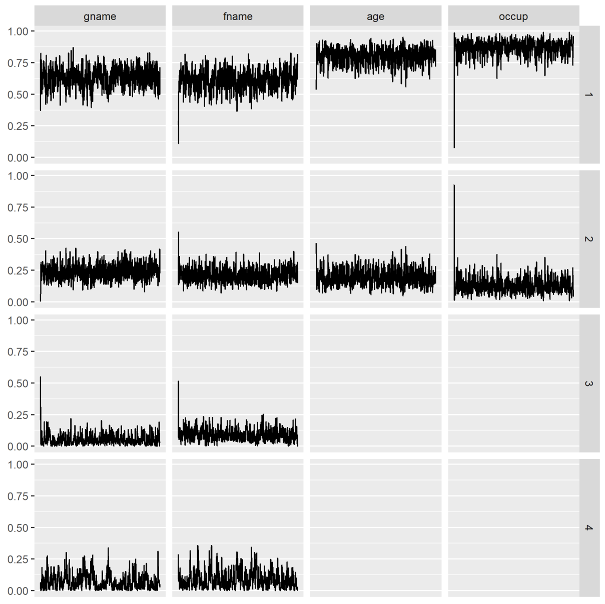

Figure 9: Representative traceplot of m parameter from simulation study in Section 5.2

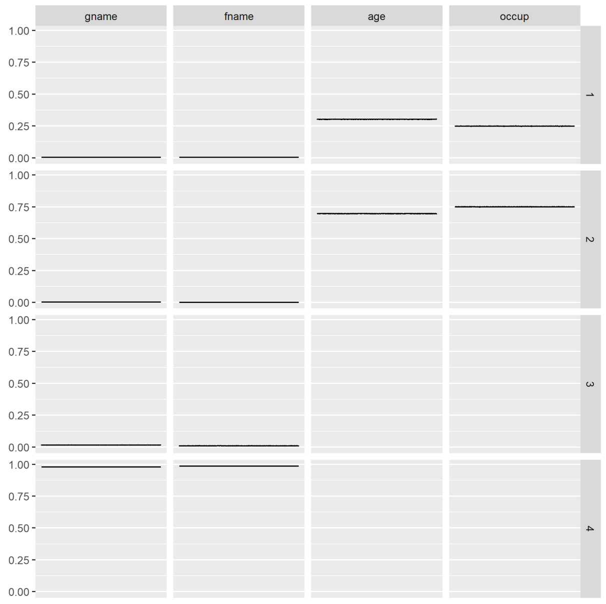

Figure 10: Representative traceplot of u parameters from simulation study in Section 5.2

	m			u		
	Agree	Partial	Disagree	Agree	Partial	Disagree
Feature 1	$\frac{9}{10}$	$\frac{9}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{3}{100}$	$\frac{96}{100}$
Feature 2	$\frac{9}{10}$	$\frac{9}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{3}{100}$	$\frac{96}{100}$
Feature 3	$\frac{9}{10}$	$\frac{9}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{3}{100}$	$\frac{96}{100}$
Feature 4	$\frac{9}{10}$	$\frac{9}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{3}{100}$	$\frac{96}{100}$

Table 9: Probabilities used for m and u distributions in simulation study in Appendix 7.6.

7.6 Additional Speed Simulation Study

To illustrate that different constructions of the comparison vectors lead to similar speed gains, we replicate the speed study of Section 5.1 under different settings. Here, we use four fields of comparison, each with three possible levels of agreement, resulting in $3^4 = 81$ possible patterns. The $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters for this simulation are shown Table 9.

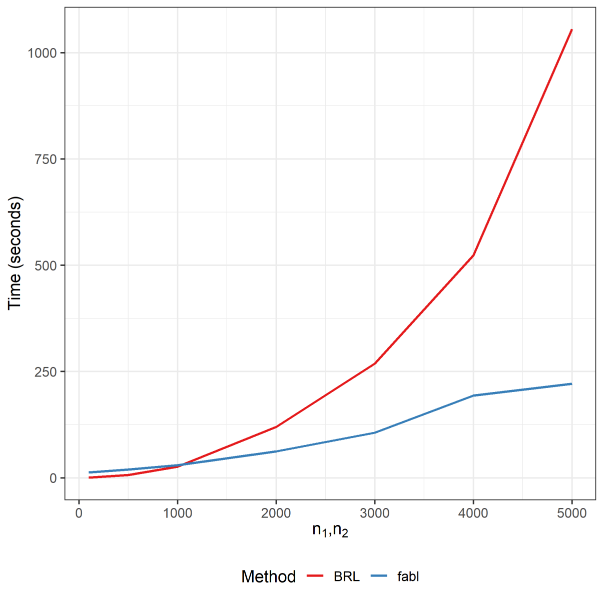

Figure 11: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including hashing step for `fabl`, for increasing values of both n_1 and n_2. We see near quadratic growth in run-time for `BRL`, and near linear growth for `fabl`.

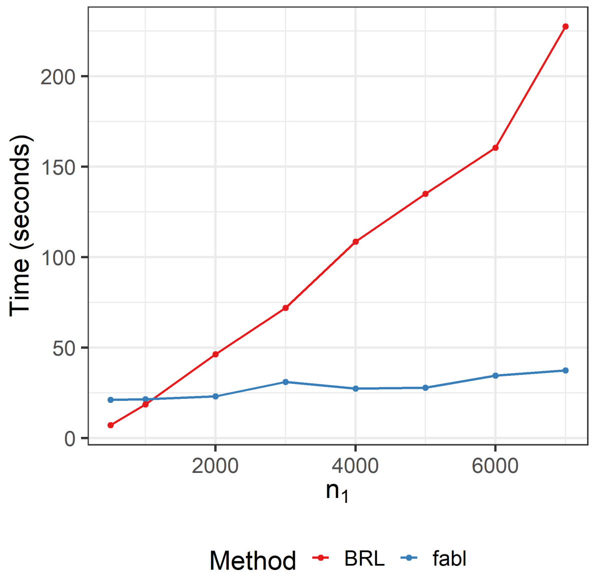

Figure 12: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including hashing step for `fabl`, with increasing n_1 and n_2 fixed at 500. We see linear growth in run-time for `BRL`, and near constant run-time for `fabl`.

7.7 Traceplots for El Salvador Case Study

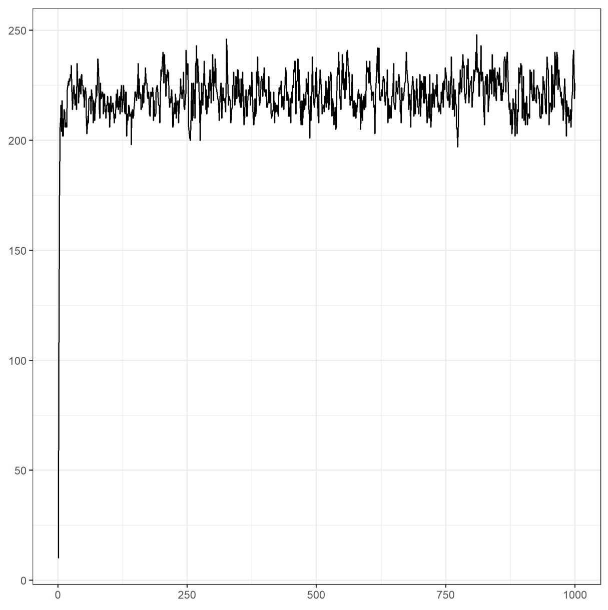

Figure 13: Traceplot for number of matches found across datasets in El Salvador case study.

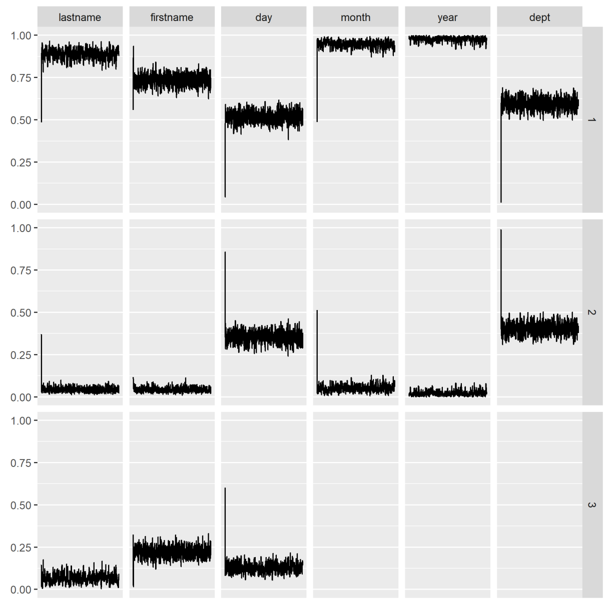

Figure 14: Traceplot for m parameter in El Salvador case study.

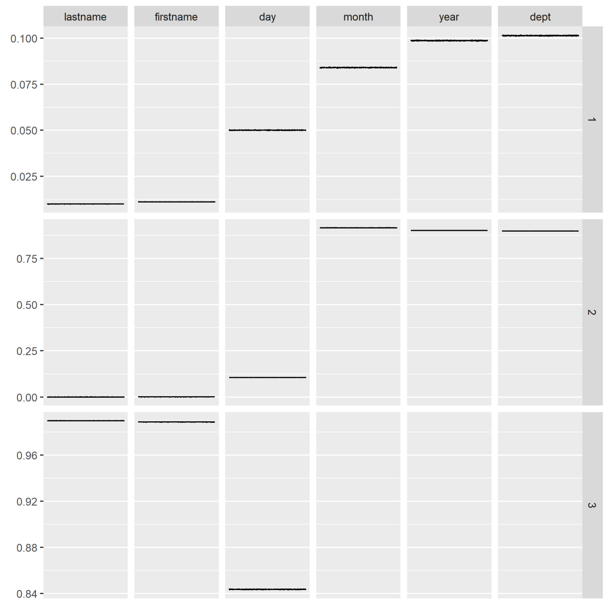

Figure 15: Traceplot for u parameter in El Salvador case study.