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 data files 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: Bayesian methods, distributed computing, entity resolution, hashing, record linkage.

1 Introduction

In many data analysis tasks, analysts seek to identify duplicate records across two data files. 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 data files that contain duplications across, but not within, the respective data files.

Many statistical record linkage methods are extensions of the seminal work of Fellegi and Sunter (1969) and Newcombe et al. (1959). Specifically, Fellegi and Sunter (1969) created comparison vectors for each pair of records in the data files 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 directly model the linkage variables (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 (1969) 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 data file, meaning

*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

that 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 (1969) 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 data files 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 data files 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 data files 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 data file, while computation time under `fabl` grows linearly, only with the size of the smaller data file.

In what follows, Section 2 reviews the work of Fellegi and Sunter (1969) 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. Finally, Section 7 summarizes our contributions and highlights areas for further research.

2 Review of Prior Work

Consider two data files $\boldsymbol{X}_1$ 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 $n_1 \geq n_2$. For ease of readability, we follow the convention established by Sadinle (2017) and say "record $i \in \boldsymbol{X}_1$" rather than the more compact x_{1i}. Under bipartite matching, the set of matches across data files can be represented in two equivalent ways. First, we may use a matrix $\Delta \in \{0,1\}^{n_1 \times n_2}$, where

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

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 (1969) 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 $\boldsymbol{\gamma}_{ij} = (\gamma_{ij}^1, \gamma_{ij}^2, \dots, \gamma_{ij}^f, \dots, \gamma_{ij}^F)$. 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 example, if zip code is linking feature f, we can set $\gamma_{ij}^f = 1$ when the zip codes for record i and j agree exactly, and set $\gamma_{ij}^f = 2$ when they do not. For numerical features, we can use absolute difference between the two feature values. For example, if age is linking field f, we can set $\gamma_{ij}^f = 1$ when the ages for record i and j match exactly, $\gamma_{ij}^f = 2$ when the ages for record i and j are within one year but not equal, and $\gamma_{ij}^f = 3$ when the ages are two or more years apart. 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).

More generally, let $\mathcal{S}_f(i,j)$ denote a similarity measure for feature f of records i and j, where the range of $\mathcal{S}_f$ can be divided into L_f intervals denoted by $I_{f1}, \dots, I_{fL_f}$. Here, I_{f1} 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, it is common to encounter missing information in record $i \in \boldsymbol{X}_1$ or $j \in \boldsymbol{X}_2$. As a result, the comparison vector γ_{ij} will have missing values. We assume that this missingness occurs at random (MAR, per Little and Rubin (2002)). With the MAR assumption, we can marginalize over the missing data, and do all computation simply using the observed data. To notate a missing value in any γ_{ij}^f, we use $I_{obs}(\gamma_{ij}^f) = 1$ when γ_{ij}^f is observed and $I_{obs}(\gamma_{ij}^f) = 0$ otherwise.

2.1 Fellegi-Sunter Models

The seminal Fellegi and Sunter (1969) 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. Using these assumptions, Winkler (1999); Jaro (1989); Larsen and Rubin (2001); Enamorado et al. (2019) and others model the comparison data through mixture models of the form

$$\Gamma_{ij} = \gamma_{ij} \mid \Delta_{ij} = 1 \overset{iid}{\sim} \mathcal{M}(\boldsymbol{m}), \tag{3a}$$

$$\Gamma_{ij} = \gamma_{ij} \mid \Delta_{ij} = 0 \overset{iid}{\sim} \mathcal{U}(\boldsymbol{u}), \tag{3b}$$

$$\Delta_{ij} \overset{iid}{\sim} \text{Bernoulli}(\lambda). \tag{3c}$$

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, $\boldsymbol{m} = (\boldsymbol{m}_1, \dots, \boldsymbol{m}_F)$, where $\boldsymbol{m}_f = (m_{f1}, \dots, m_{fL_f})$ 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 $u_{fl} = P(\gamma_{ij}^f = l|\Delta_{ij} = 0)$.

Within this framework, record pairs are independently classified as matches and non-matches based on the estimated parameters. However, such independent classifications often leads to links that violate one-on-one matching assumptions, requiring post-processing to achieve desirable results. To address this issue, Jaro (1989) proposed an optimization technique using the estimated model parameters to produce a bipartite matching. Sadinle (2017) later showed that this method is equivalent to a maximum likelihood estimate.

To estimate a one-to-one matching without using a post-processing step, Sadinle (2017) incorporates 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 proposes the "beta distribution for bipartite matching" for the linkage parameter $\boldsymbol{Z}$. He assigns a prior distribution for the probability of the indicator that a record in $\boldsymbol{X}_2$ has 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

Beta(α_π, β_π). 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

$$P(\boldsymbol{Z}|\alpha_\pi, \beta_\pi) = \frac{(n_1 - n_{12}(\boldsymbol{Z}))!}{n_1!} \frac{\mathrm{B}(n_{12}(\boldsymbol{Z}) + \alpha_\pi, n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi)}{\mathrm{B}(\alpha_\pi, \beta_\pi)}, \tag{4}$$

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 each 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. For each Z_j, we have

$$p(Z_j = z_j|\pi) = \begin{cases} \frac{1}{n_1}\pi, & z_j \leq n_1; \\ 1 - \pi, & z_j = n_1 + j; \end{cases} \tag{5}$$
$$\pi \sim \text{Beta}(\alpha_\pi, \beta_\pi).$$

We can interpret (5) 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 $\boldsymbol{X}_1$.

In the Wortman (2019) flat prior, each value $\{1, \dots, n_1, n_1 + j\}$ is a priori equally likely for Z_j. This however amounts to a prior probability of $n_1/(n_1 + 1)$ that record j has a match in $\boldsymbol{X}_1$. In our preliminary studies, the flat prior results in poor precision; hence, we prefer (5). We also note that the flat prior is equivalent to a special case of the fast Beta prior with π fixed at the mean of a Beta$(1, 1/n_1)$ random variable.

Note that linkage with `fabl` 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, whereas λ in the Fellegi-Sunter model estimates the proportion of record pairs that are matches. We find π to be more a 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 n_2 independent matching decisions. As shown in Sections 5 and 6, this allows `fabl` to fit a Bayesian record linkage model like `BRL` while making computational efficiency gains possible by exploiting independence.

Symbol	Description
$\boldsymbol{X}_1, \boldsymbol{X}_2$	data files
$i \in 1, \ldots, n_1$	index over records in $\boldsymbol{X}_1$
$j \in 1, \ldots, n_2$	index over records in $\boldsymbol{X}_2$
$f \in 1, \ldots F$	index over fields used for comparisons
$l \in 1, \ldots L_f$	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.

To obtain an estimate $\hat{\boldsymbol{Z}}$ of the linkage structure, we use the loss functions and Bayes estimate from Sadinle (2017). 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 the one-to-one matching requirement, we minimize the expected loss subject to the constraint that $\hat{Z}_j \neq \hat{Z}_{j'}$ for all $j \neq j'$. Details for the initial Bayes estimate and the post-processing procedure are provided in Appendix 8.1.

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

$$\mathcal{L}(\boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{u} \mid \boldsymbol{\Gamma}) = \prod_{j=1}^{n_1} \prod_{i=1}^{n_2} \prod_{f=1}^{F} \prod_{l=1}^{L_f} \left[m_{fl}^{I(Z_j = i)} u_{fl}^{I(Z_j \neq i)} \right]^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)}, \tag{6a}$$

$$\boldsymbol{m}_f \sim \text{Dirichlet}(\alpha_{f1}, \ldots, \alpha_{fL_f}), \tag{6b}$$

$$\boldsymbol{u}_f \sim \text{Dirichlet}(\beta_{f1}, \ldots, \beta_{fL_f}), \tag{6c}$$

$$p(Z_j = z_j \mid \pi) = \begin{cases} \frac{1}{n_1}\pi, & z_j \leq n_1; \\ 1 - \pi, & z_j = n_1 + j; \end{cases} \tag{6d}$$

$$\pi \sim \text{Beta}(\alpha_\pi, \beta_\pi). \tag{6e}$$

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 data files; that is, $\boldsymbol{Z} = (n_1 + 1, \ldots, n_1 + n_2)$. To take the $(s+1)$th sample of each $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$ given $\boldsymbol{Z}^s$, we use the full conditionals,

$$\boldsymbol{m}_f^{(s+1)} \mid \Gamma, \boldsymbol{Z}^{(s)} \sim \text{Dirichlet}(\alpha_{f1}(\boldsymbol{Z}^{(s)}), \ldots, \alpha_{fL_f}(\boldsymbol{Z}^{(s)})), \tag{7a}$$

$$\boldsymbol{u}_f^{(s+1)} \mid \Gamma, \boldsymbol{Z}^{(s)} \sim \text{Dirichlet}(\beta_{f1}(\boldsymbol{Z}^{(s)}), \ldots, \beta_{fL_f}(\boldsymbol{Z}^{(s)})), \tag{7b}$$

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

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

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

where, for all $i \in \boldsymbol{X}_1$,

$$w_{ij} = \prod_{f=1}^{F} \prod_{l=1}^{L_f} \left(\frac{m_{fl}^{(s)}}{u_{fl}^{(s)}} \right)^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)} . \tag{9}$$

Derivations for these full conditionals can be found in Appendix 8.2.

4 Efficient and Scalable Implementation

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

After computing all comparison vectors within a block, the modeller can further reduce the number of comparison vectors used in the linkage algorithm through indexing. 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. However, the impact of indexing on model parameters is not well understood; (Murray, 2016) reviewed this issue in the context of the frequentist Fellegi-Sunter model, leaving the effect of indexing on Bayesian record linkage models to future work.

With `fabl`, we introduce two techniques to further expand the scalability of probabilistic record linkage. First, we propose hashing methods that allow us to compute sufficient statistics that reduce the computational complexity of the Gibbs sampler. Second, we introduce storage efficient indexing, which reduces the memory costs associated with unlikely matches. For convenience, Table 2 summarizes the notation introduced throughout this section.

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.

4.1 Data Representation, Hashing, and Storage

Following an insight from Enamorado et al. (2019), we recognize that record pairs contribute to posterior calculations only through their agreement pattern as represented by γ_{ij}. To make this more precise, let $h_1, \ldots, h_P$ denote the unique agreement patterns observed in $\boldsymbol{X}_1 \times \boldsymbol{X}_2$, and let $\mathcal{P} = \{h_1, \ldots, h_P\}$. 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 8.3 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

$$h^*(i,j) = \sum_{f=1}^{F} I(\gamma_{ij}^f > 0) 2^{\gamma_{ij}^f + I(k>1)\sum_{e=1}^{k-1}(L_e - 1)}. \tag{10}$$

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 map these integers to sequential integers from $\{1, \ldots, P\}$ corresponding to the enumerated patterns in $\mathcal{P}$. When the (i,j) pair exhibits pattern p, 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 $\mathcal{R} = \{r_{j_p} | j \in \boldsymbol{X}_2, p \in 1, \ldots, P\}$. 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

$$N_{j_p} = |r_{j_p}| = \sum_{i=1}^{n_1} I((i,j) \in h_p). \tag{11}$$

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

$$H_p = \sum_{j=1}^{n_2} N_{j_p}. \tag{12}$$

We collect these sufficient statistics in the sets $\mathcal{N} = \{N_{j_p} | j \in \boldsymbol{X}_2, p \in 1, \dots, P\}$ and $\mathcal{H} = \{H_p | p \in 1, \dots, P\}$. As we 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

We next introduce storage efficient indexing (SEI), which allows us to compute $\mathcal{N}$ and $\mathcal{H}$ for all $n_1 \times n_2$ record pairs but greatly reduce the memory costs associated with unlikely matches. This allows all-to-all comparisons for substantially larger linkage tasks. 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 the link for record 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 record labels, we 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, with SEI we 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 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 few 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 linkage tasks with large amounts of records, we can partition the two data files $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ into C and D smaller disjoint chunks $\{\boldsymbol{X}_{1c} | c = 1, \dots, C\}$ and $\{\boldsymbol{X}_{1d} | d = 1, \dots, D\}$ for more manageable computations. For each data file $\boldsymbol{X}_{1c}$, we conduct all-to-all comparisons with each $\boldsymbol{X}_{2d}$ to construct the comparison matrix Γ^{cd}. We then conduct hashing, compress information through SEI, and delete those comparison vectors from memory before continuing with the next chunk of data. In detail, we calculate

$$r_{j_p}^{cd} = \{i \in \boldsymbol{X}_{1c} | (i, j) \in h_p, j \in \boldsymbol{X}_{2d}\}, \tag{13a}$$

$$N_{j_p}^{cd} = |r_{j_p}^{cd}|, \tag{13b}$$

$$H_p^{cd} = \sum_{j \in \boldsymbol{X}_{2d}} N_{j_p}. \tag{13c}$$

These can computed serially or in parallel. We reduce memory costs through SEI on each r_{j_p}, while retaining counts of each agreement pattern through $N_{j_p}^{cd}$ and H_p^{cd}. Summary statistics from each pairwise chunk comparison can be easily synthesized to recover sufficient statistics for the full comparison matrix Γ. Suppressing SEI notation, we combine information through

$$r_{j_p} = (r_{j_p}^{11}, \dots, r_{j_p}^{cd}) \text{ for } c = 1, \dots, C \text{ and } d = 1, \dots, D, \tag{14a}$$

$$N_{j_p} = \sum_{c=1}^{C} \sum_{d=1}^{D} N_{j_p}^{cd}, \tag{14b}$$

$$H_p = \sum_{c=1}^{C} \sum_{d=1}^{D} H_p^{cd}. \tag{14c}$$

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 they become computationally burdensome when working on large linkage tasks.

Instead, we use one-hot encodings of the agreement patterns $\mathcal{P}$. Let $H_p^m = \sum_{j=1}^{n_2} I\left((Z_j, j) \in h_p, Z_j \leq n_1\right)$ 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 $H_p^u = H_p - H_p^m$. Let $\boldsymbol{\alpha_0} = (\alpha_{11}, \dots, \alpha_{FL_f})$ be a concatenated vector of prior parameters for the $\boldsymbol{m}$ distributions, and define $\boldsymbol{\beta_0} = (\beta_{11}, \dots, \beta_{FL_f})$ similarly for the $\boldsymbol{u}$ distributions. Then, the terms needed for the posterior updates in (7a) and (7b) are contained in the vectors

$$\boldsymbol{\alpha}(\boldsymbol{Z}) = \boldsymbol{\alpha_0} + \sum_{p=1}^{P} H_p^m \times h_p, \tag{15a}$$

$$\boldsymbol{\beta}(\boldsymbol{Z}) = \boldsymbol{\beta_0} + \sum_{p=1}^{P} H_p^u \times h_p. \tag{15b}$$

Although sampling Z_j from the full conditional provided in (8) is conceptually straightforward, it can become computationally expensive 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 quantity expressed in (9) associated with each unique pattern, and denote this quantity as w_p. We then sample the agreement pattern between j and its potential link, according to

$$P\left(\left(Z_j^{(s+1)}, j\right) \in h_p \mid \tilde{\Gamma}, \boldsymbol{m}, \boldsymbol{u}, \boldsymbol{Z^{(s)}}\right) \propto \begin{cases} w_p \times N_{j_p}, & p \leq P; \\ n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}^{(s)}) + \beta_\pi}{n_{12}(\boldsymbol{Z}^{(s)}) + \alpha_\pi}, & p = P + 1. \end{cases} \tag{16}$$

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 using

$$P\left(Z_j^{(s+1)} = z_j \mid \left(Z_j^{(s+1)}, j\right) \in h_p\right) = \begin{cases} \frac{1}{N_{j_p}}, & (z_j, j) \in h_p; \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

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; the second step is sampling uniformly at random, which is computationally simple even for large sets of candidate records. These changes can greatly improve the speed of the sampler, and each can be parallelized if desired for additional computational speed-ups.

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 exhibit in $\boldsymbol{X}_1 \times \boldsymbol{X}_2$. We assume B processors available for parallelization and a Gibbs sampler with T iterations. Then, the overall computational complexity of `fabl` is $O(\frac{F}{B}n_1n_2) + O(\frac{T}{B}n_2P)$.*

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 $\boldsymbol{Z}$ in a single iteration is $O(n_2P)$. Since $P << n_1$ in most applications, we have reduced the complexity of sampling $\boldsymbol{Z}$ from $O(Fn_1n_2)$ under `BRL` to $O(n_2P)$ under `fabl`. With parallelization, this complexity is further reduced to $O(\frac{1}{B}n_2P)$, and so the entire Gibbs sampler has complexity $O(\frac{T}{B}n_2P)$ In summary, the total computational complexity is $O(\frac{F}{B}n_1n_2) + O(\frac{T}{B}n_2P)$. □

5 Simulation Studies

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

	$\boldsymbol{m}$		$\boldsymbol{u}$	
	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 $\boldsymbol{m}$ and $\boldsymbol{u}$ distributions in simulation study in Section 5.1.

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 $\Gamma \in \mathbb{R}^{n_1 n_2 \times F}$. 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. Since we have five binary comparison fields, the number of unique patterns P is bounded above by $2^5 = 32$, a bound which is consistently attained in simulations with more records.

The Gibbs sampler in the implementation of `BRL` that we use is coded in C (Sadinle, 2017). In contrast, we use non-optimized code written only in `R` for `fabl`. While this complicates comparisons, and indeed disfavors `fabl`, the computational speed gains for `fabl` are still evident, especially for larger sample sizes. Additionally, although `fabl` is amenable to parallelization, this simulation is 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 sample sizes. 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 near 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 8.6.

5.2 Accuracy

Computational speed-ups are only worthwhile if not accompanied by a notable loss of record linkage accuracy. Therefore, we examine the accuracy of `fabl` relative to `BRL` by

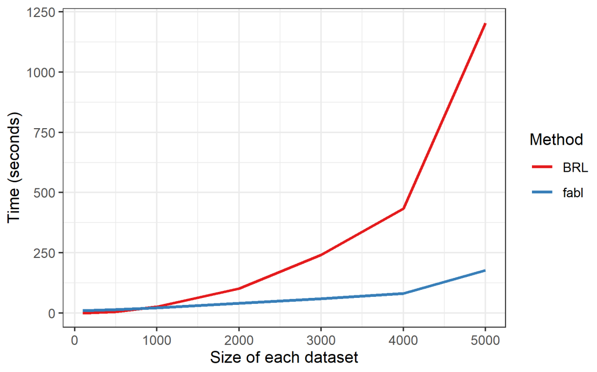

Figure 1: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including the hashing step for `fabl`, for increasing values of both n_1 and n_2, as described in Section 5.1. We see near quadratic growth in run-time for `BRL`, and near linear growth for `fabl`.

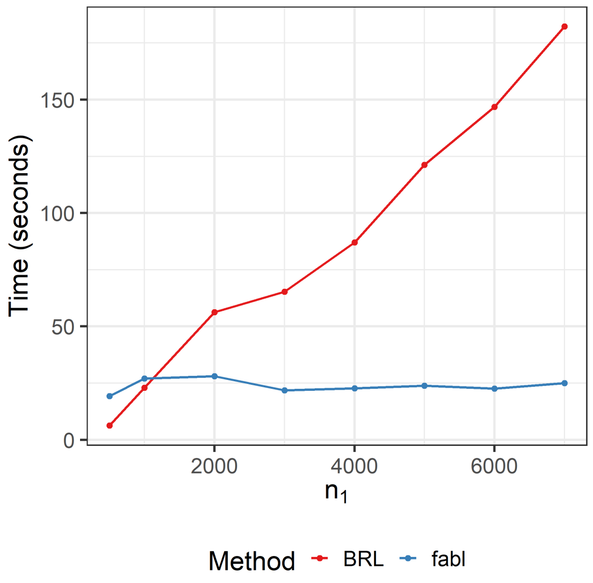

Figure 2: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including hashing step for `fabl`, with n_2 fixed at 500, as described in Section 5.1. We see near linear growth in run-time for `BRL`, and near constant run-time for `fabl`.

replicating a simulation study from Sadinle (2017). The simulations employ a collection of synthetic data files with varying amounts of error and overlap (the number of records

		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 data files of Section 5.2.

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. These synthetic data files are available in the supplement to Sadinle (2017).

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 data files, 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 data files. Under each of these settings, we have 100 pairs of simulated data files in order to obtain uncertainty quantification on our performance metrics. We use uniform priors for all $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters, with $\alpha_{fl} = \beta_{fl} = 1$ for all f and l. We run the Gibbs sampler for 1000 iterations, and discard the first 100 as burn-in. We calculate Bayes estimates $\hat{\boldsymbol{Z}}$ of the linkage structure using the loss function and post-processing procedure described in Appendix 8.1. Traceplots for parameters of interest for one example simulation are provided in Appendix 8.5; they show no obvious concern over MCMC convergence. We also replicate this simulation allowing `fabl` to leave some components of the linkage structure undetermined and left for clerical review; these results are in Appendix 8.4.

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 than `BRL` 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

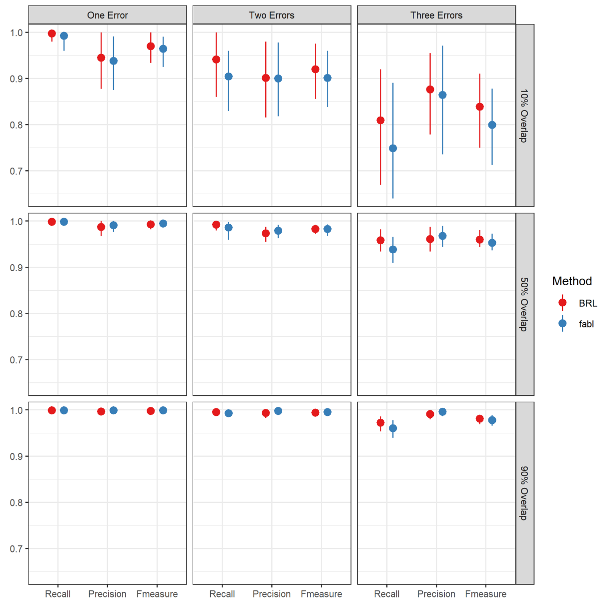

Figure 3: Posterior means and credible intervals for accuracy metrics under the replication of simulation study from Sadinle (2017). For each level of overlap and each level of error, we have 100 paired sets of 500 records. Thus this table summarizes results for 900 data files. We see comparable performance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

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, 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 comprises digitized denunciations published throughout the conflict, and the CDHES dataset comprises casualties 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 data files are thought to be fairly

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

		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.

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 data files before analyzing the data (Lum et al., 2013).

There are several challenges with these data. First, both data files 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 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 use uniform 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` runs 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 8.7.

For completeness, we note that linkage with the more detailed comparison vectors requires 240 seconds for `BRL`, and 261 seconds for `fabl`. Apparently, the number of

Fields	Similarity	Level of Disagreement 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 data for increased speed under `fabl`. This set up leads to 216 possible agreement patterns in total.

Fields	Similarity	Level of Disagreement 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.

patterns is 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 data files. We calculate posterior samples of the size of the overlap across files by finding the number of links in each iteration of the Gibbs sampler, and subtracting 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 see similar results under `BRL`, with a Bayes estimate of 181 individuals recorded in both data files, and a posterior 95% credible interval of (211, 244). See Figure 4 for a visual comparison of the Bayes estimates 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.

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 (Steorts et al., 2016). 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. Hence, the assumptions of our model hold for this study. We seek to link

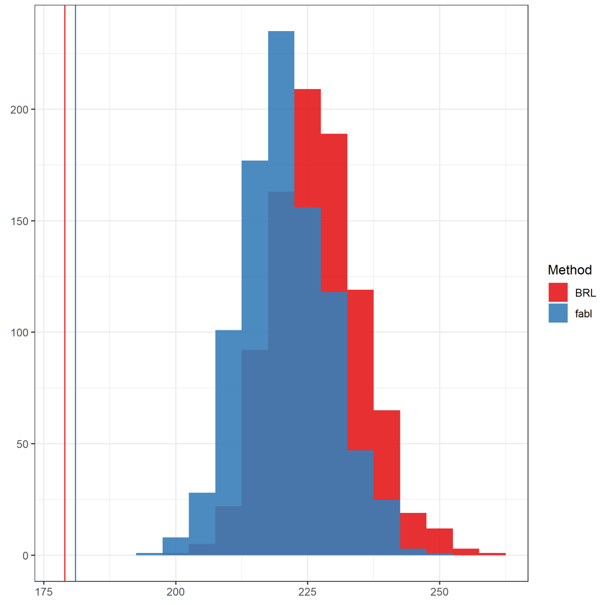

Figure 4: Posterior distribution and Bayes estimate of overlap across the two files. We note they are quite similar under both methods.

records over the $n_1 = 20485$ individuals from 1982 to the $n_2 = 17466$ individuals 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 data files.

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 8.5 and 8.7

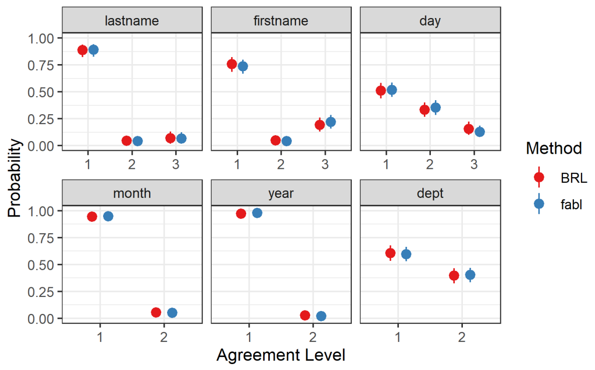

Figure 5: Posterior estimates of $\boldsymbol{m}$ parameters with 95% credible intervals for the El Salvador case study. They are quite similar across the two methods.

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 data file. This makes `fabl` computationally advantageous in many linkage scenarios, particularly when one data file 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 practitioners an option for larger record linkage tasks potentially even 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 impractical under `BRL`.

Although the `fabl` method greatly reduces the memory costs for all-to-all comparisons, computing all $n_1 \times n_2$ record pairs still can be prohibitive 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. In future work, we seek to extend `fabl` to account for such deterministic blocking, making the framework amenable to arbitrarily large linkage tasks.

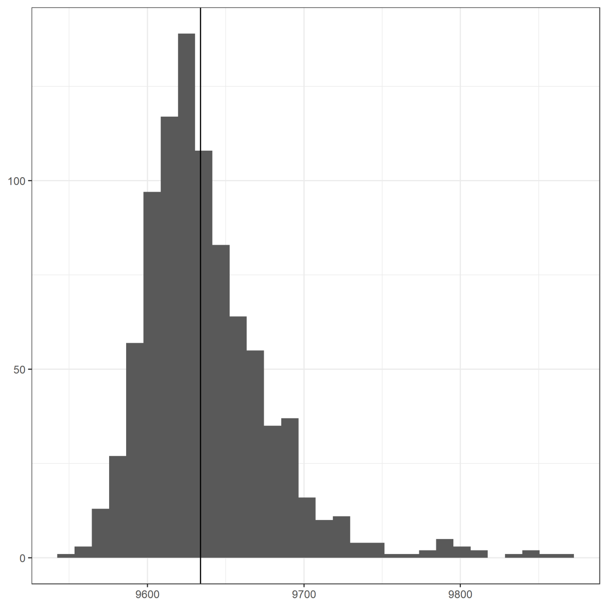

Figure 6: Posterior distribution and Bayes estimate of overlap across years 1982 and 1989 of NLTCS data.

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, Article 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. 14

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. 14

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, 4, 8

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

Fellegi, I. P. and Sunter, A. B. (1969), “A Theory for Record Linkage,” *Journal of the American Statistical Association*, 64, 1183–1210. 1, 2, 3, 4

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

Larsen, M. D. and Rubin, D. B. (2001), “Iterative Automated Record Linkage Using Mixture Models,” *Journal of the American Statistical Association*, 96, 32–41. 4

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. 16

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, 12, 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, 17, 24

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

Winkler, W. E. (1999), "The State of Record Linkage and Current Research Problems," Tech. rep., Statistical Research Division, U.S. Bureau of the Census. 4

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