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

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

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

Abstract. Recently, researchers have developed Bayesian versions of the Fellegi Sunter model for record linkage. These 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. We propose and investigate a variation on Bayesian Fellegi Sunter models that we call fast beta linkage, or `fabl`. Specifically, in `fabl` we propose 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 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.

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

1 Introduction

In many data analysis tasks, analysts seek to identify duplicate records across two databases. 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 (**????**). In this article, we consider bipartite record linkage, which merges two databases together that contain duplications across but not within the respective databases.

Many statistical record linkage methods are extensions of the seminal work of **?** and **?**. Specifically, **?** 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 (**??????**). Additionally, some methods model records directly (**???**), but in this paper, we build on the contributions to the comparison vector approach.

The independent pairwise matching assumption from **?** 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 database, meaning that each record should be linked with at most one other record. Thus, when the procedure results in many to

*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

© 0000 International Society for Bayesian Analysis DOI: 0000

one matches, some of these links must be false. Many extensions to **?** resolve these false matches as a post-processing step (**?**), but this model misspecification can still lead to poor results (**?**).

Alternatively, one can embed one-to-one matching requirements into the model specification itself (**??**), at an additional computational cost. **?** 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. **?** used simulated annealing to target the space of matches permitted under the one-to-one constraint, but the method is quite computationally intensive and, to our knowledge, has not been applied on databases with more than 100 records. **?** proposed the Beta Record Linkage model (`BRL`), using a prior 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 modeller can weight 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 **?**, we relax the one-to-one matching requirement of `BRL` and propose independent priors over the matching space, creating a "many-to-one" model for record linkage. 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. We demonstrate that even in cases where a bipartite matching is desired, `fabl` (with a simple post-processing procedure) provides accurate estimation of the linkage structure and other parameters, more information through which to asses model misspecification, and greatly enhanced speed. Open source software to use `fabl` in `R` is available through Github.

In what follows, Section 2 reviews the work of **?** and **?**. Section 3, proposes the `fabl` model, derives the Gibbs sampler for posterior inference, and provides 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 calculatoins 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 databases $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ with n_1 and n_2 records respectively. Without loss of generality, denote the files such that $n_1 \geq n_2$. In the context of bipartite matching, we assume that there are duplications across, but not within, each database. Under this framework, the set of matches across databases 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 $\boldsymbol{Z} = (Z_1, \ldots, Z_{n_2})$ for the records in database $\boldsymbol{X}_2$ such that

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

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 the sets of $\boldsymbol{M}$ and $\boldsymbol{U}$. We refer to record pairs that are estimated as matches as "links", and record pairs that are estimated as non-matches as "non-links".

Intuitively, co-referent records (those that refer to the same entity) should be similar; records that are not co-referent should not be similar. **?** proposed encoding this is 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 $\boldsymbol{\gamma}_{ij} = (\gamma_{ij}^1, \gamma_{ij}^2, \ldots, \gamma_{ij}^f, \ldots, \gamma_{ij}^F)$. In most cases, $\boldsymbol{\gamma}_{ij}$ consists of one comparison for each feature shared between the two datasets.

The simplest way to compare two records is to check for agreement or disagreement, and this is commonly used for categorical variables. For more complex measurements, we can take into account partial agreement to more richly characterize the comparison; for numerical data, we can use absolute difference, and for text data, we can use string distance metrics such as Levenstein or Jaro-Winkler distance (**?**). We then can set thresholds that allow us to represent comparisons through discrete levels of disagreement (**??**). 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 $I_{f0}, I_{f1}, \ldots, I_{fL_f}$. 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 in the following way: $\gamma_{ij}^f = \ell$ if $\mathcal{S}_f(i,j) \in I_{f\ell}$. The choice of $I_{f\ell}$ are application specific, which we discuss in our simulation and case studies.

In practice, it is not feasible to make all-to-all record comparisons as the computational complexity is of order $O(n_1 \times n_2)$. The most common solution is to utilize blocking, which places similar records into partitions, or "blocks," to reduce this computational burden (**??**). In deterministic blocking, the modeller 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. However, blocking on an unreliable field can lead to missed matches, making this form of blocking undesirable in some situations (**?**).

In this paper, we use hashing and a new technique called storage efficient indexing to increase the scale of the linkage tasks we can undertake without blocking. This is useful when there is no reliable blocking field available, or one desires estimates of model parameters for the entire sample in question. In practice, `fabl` can be combined with blocking, but all derivations, simulations, and case studies are presented here without blocking.

2.1 Fellegi-Sunter Model

The comparison data $\boldsymbol{\gamma}_{ij}$ are not sufficient to determine whether a record is a match or non-match because of errors that naturally occur in the data. This motivated **?** to consider the likelihood ratio

$$w_{ij} = \frac{P(\gamma_{ij} \mid \Delta_{ij} = 1)}{P(\gamma_{ij} \mid \Delta_{ij} = 0)} \tag{3}$$

as a weight to estimate if record pairs are a match or non-match. This ratio will be large when there is strong evidence of the pair being a match, and small otherwise.

In practice, $P(\cdot \mid \Delta_{ij} = 1)$ and $P(\cdot \mid \Delta_{ij} = 0)$ are unknown to the modeler, and thus must be estimated. Under the Fellegi Sunter model, we assume both that comparison vectors are independent given their matching status, and that the match status of the record pairs are independent. This allows us to model the comparison data using mixture models in the following way:

$$\begin{aligned} \Gamma_{ij} = \gamma_{ij} \mid \Delta_{ij} = 1 &\overset{iid}{\sim} \mathcal{M}(\boldsymbol{m}), \\ \Gamma_{ij} = \gamma_{ij} \mid \Delta_{ij} = 0 &\overset{iid}{\sim} \mathcal{U}(\boldsymbol{u}), \\ \Delta_{ij} &\overset{iid}{\sim} \text{Bernoulli}(p), \end{aligned} \tag{4}$$

where p is the proportion of records that match, and $\boldsymbol{m}$ is the set of parameters for the matching record pairs $m_{fl} = P(\gamma_{ij}^f = l | Z_j = i)$ for all fields f and agreement levels ℓ_f, and $\boldsymbol{u}$ is defined similarly for non-matching pairs. These unknown parameters are often estimated using the EM algorithm (**?**).

After estimating $\boldsymbol{m}$ and $\boldsymbol{u}$, the Fellegi Sunter method uses thresholds T_m and T_u such that all record pairs with $w_{ij} > T_m$ are declared as links, and all those with $w_{ij} < T_u$ are declared as non-links (with those such that $T_u < w_{ij} < T_m$ left undetermined and subject to clerical review). This is a problem in practice as transitive closures can be violated. Since each w_{ij} is calculated independently, there can be situations in which both $w_{ij} > T_m$ and $w_{i'j} > T_m$, so both (i, j) and (i, j') are declared as links, even though such a matching is not allowed by assumption.

To address this issue, **?** proposed solving the following linear sum assignment problem:

$$\max_{\Delta} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_{ij} \Delta_{ij} \quad \text{subject to} \quad \Delta_{ij} \in \{0, 1\}; \tag{5}$$

$$\sum_{i=1}^{n_1} \Delta_{ij} \leq 1, j = 1, 2, \ldots n_2; \text{ and}$$
$$\sum_{i=1}^{n_2} \Delta_{ij} \leq 1, j = 1, 2, \ldots n_1.$$

The solution to this problem, for which several simple algorithms exist, is a bipartite matching that maximizes the sum of the Fellegi Sunter weights among matched pairs. Jaro provided no theoretical justification for using this approach; however **?** recently showed that under certain conditions, (5) is the maximum likelihood estimate for bipartite matching.

2.2 Beta Record Linkage Model

One criticism of the independent matching assumption is that the decision rule leads to "many-to-many assignments," breaking one-to-one matching assumptions. One way of resolving this issue to incorporate the constraint directly into the model as follows:

$$\begin{aligned} \boldsymbol{\Gamma}_{ij} \mid Z_j = i &\overset{iid}{\sim} \mathcal{M}(\boldsymbol{m}) \\ \boldsymbol{\Gamma}_{ij} \mid Z_j \neq i &\overset{iid}{\sim} \mathcal{U}(\boldsymbol{u}) \\ \boldsymbol{Z} &\sim \mathcal{B}, \end{aligned} \tag{6}$$

where $\mathcal{B}$ represents a prior on the space of bipartite matchings, and $\mathcal{M}(\boldsymbol{m})$ and $\mathcal{U}(\boldsymbol{u})$ are models for the comparison vectors of matches and non-matches. For `BRL`, **?** proposed the beta distribution for bipartite matching, given by

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

where $\mathrm{B}(\cdot, \cdot)$ represents the Beta function, and hyperparameters α_π and β_π encode prior beliefs about the proportion of records in $\boldsymbol{X_2}$ that have matches in $\boldsymbol{X_1}$. This prior induces a Gibbs sampler that strictly enforces one-to-one matching, removing previously matched records from the set of candidate records when sampling Z_j. This creates a dependency that makes the sampler inherently serial, which can slow down the sampler when working on large linkage tasks.

3 Fast Beta Linkage

Recall the compact representation of the matching structure $\boldsymbol{Z} = (Z_1, Z_2, \ldots Z_{n_2})$ with $Z_j \in \{1, 2, \ldots, n_1, n_1 + j\}$, where $Z_j < n_1$ indicates a link and $Z_j = n_1 + j$ indicates a nonlink. In constrast to the prior over the vector $\boldsymbol{Z}$ from **?**, we follow **?** and use independent priors for each component Z_j. However, unlike **?** who proposes a flat prior for Z_j, we propose a proper and flexible prior. We denote the fast beta prior as follows:

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

Intuitively, this set of priors says that record $j \in \boldsymbol{X}_2$ has some match in $\boldsymbol{X}_1$ with probability π, and that each record $i \in \boldsymbol{X}_1$ is equally likely to be that match. π itself is given a prior distribution with hyperparameters α_π and β_π to encode prior beliefs about the proportion of records in $\boldsymbol{X_2}$ that have matches in $\boldsymbol{X_1}$. One choice might be $\pi \sim \text{Beta}(1,1)$, which corresponds to a prior belief that non-matches and matches are equally likely, and another might be $\pi \sim \text{Beta}\left(1, \frac{1}{n_1}\right)$, which corresponds to a uniform prior on the labeling of $\boldsymbol{Z}$. We note here that prior from **?** can be viewed as a special case of the fast beta prior, with 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. 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 achieve the benefits from `BRL` while making efficiency gains possible by exploiting independence.

Importantly, this independence relaxes the one-to-one requirement from `BRL`; the Gibbs sampler under `fabl` does ensure that each record in $\boldsymbol{X}_2$ can be matched to at most one record in $\boldsymbol{X}_1$, but allows for the possibility that multiple records in $\boldsymbol{X}_2$ match to the same record in $\boldsymbol{X}_1$. This behavior may be desirable; for example, we may be interested in matching birth records to marriage certificates, a situation in which such "many-to-one" matchings are reasonable (**?**). In situations when a bipartite estimate is desired, we can use the post processing procedure shown in section 3.2 to resolve many-to-one matchings.

With the `fabl` prior defined, we present the full model as follows:

$$\mathcal{L}(\boldsymbol{Z}, \Phi \mid \boldsymbol{\Gamma}) = \prod_{j=1}^{n_1}\prod_{i=1}^{n_2}\left[\prod_{f=1}^{F}\prod_{l=1}^{L_f} m_{fl}^{I(Z_j=i)} u_{fl}^{I(Z_j\neq i)}\right]^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)}$$
$$\boldsymbol{m}_f \sim \text{Dirichlet}(\alpha_{f0}, \ldots, \alpha_{fL_f})$$
$$\boldsymbol{u}_f \sim \text{Dirichlet}(\beta_{f0}, \ldots, \beta_{fL_f})$$
$$Z_j|\pi = \begin{cases} \frac{1}{n_1}\pi & z_j \leq n_1; \\ 1-\pi & z_j = n_1 + j \end{cases}$$

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

3.1 Gibbs Sampler

We work with the following factorization of the joint posterior distribution:

$$\begin{aligned}
p(\boldsymbol{Z}, \Phi, \pi|\Gamma) &\propto \mathcal{L}(\boldsymbol{Z}, \Phi \mid \Gamma^{obs}) p(\boldsymbol{Z}|\pi) p(\Phi) p(\pi) \\
&\propto \prod_{j=1}^{n_2} \prod_{i=1}^{n_1} \left[\prod_{f=1}^{F} \prod_{l=1}^{L_f} m_{fl}^{I(Z_j=i)} u_{fl}^{I(Z_j \neq i)} \right]^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)} \\
&\times \prod_{f=1}^{F} \prod_{l=1}^{L_f} m_{fl}^{\alpha_{fl}-1} \times \prod_{f=1}^{F} \prod_{l=1}^{L_f} u_{fl}^{\beta_{fl}-1} \\
&\times \prod_{j=1}^{n_2} \left[I(Z_j \leq n_1) \frac{1}{n_1} \pi + I(Z_j > n_1)(1-\pi) \right] \\
&\times \pi^{\alpha_\pi - 1} (1-\pi)^{\beta_\pi - 1}.
\end{aligned}$$

The $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters are updated through standard multinomial-Dirichlet distributions. Thus we have

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

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

As a function of π, $\boldsymbol{Z}$ is sequence of successes (when $z_j < n_A + 1$) and failures (when $z_j = n_A + 1$), and therefore $p(\boldsymbol{Z}|\pi) = \mathcal{L}(\pi|\boldsymbol{Z})$ is determined only by the number of links $n_{12}(\boldsymbol{Z}) = \sum_{i=1}^{n_2} I(z_j < n_1 + 1)$ encoded by $\boldsymbol{Z}$. We have

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

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

Because we sample each Z_j independently of all other $Z_{j'}$, we use only the full conditional for an individual Z_j. Let $\Gamma_{\cdot j}$ denote the set of n_1 comparison vectors with $j \in \boldsymbol{X_2}$. As a function of Z_j, $p(\Gamma_{\cdot j}|Z_j, \Phi) = \mathcal{L}(Z_j|\Gamma_{\cdot j}, \Phi)$ is a discrete distribution with

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

where

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

These w_{ij} are precisely the likelihood ratios used in the Fellegi-Sunter model to classify matches and non-matches, and we therefore call them the Fellegi Sunter weights.

With the likelihood in this form, we can derive the full conditional

$$
\begin{aligned}
p(Z_j|\Gamma_{\cdot j}, \Phi, \pi) &\propto p(\Gamma_{\cdot j}|Z_j, \Phi) P(Z_j|\pi) \\
&\propto \left(\sum_{i=1}^{n_1} w_{ij} \mathbf{1}_{z_j=i} + \mathbf{1}_{z_j=n_1+1} \right) \left(\pi \sum_{i=1}^{n_1} \frac{1}{n_1} \mathbf{1}_{z_j=i} + (1-\pi) \mathbf{1}_{z_j=n_1+1} \right) \\
&= \frac{\pi}{n_1} \sum_{i=1}^{n_1} w_{ij} \mathbf{1}_{z_j=i} + (1-\pi) \mathbf{1}_{z_j=n_1+1}.
\end{aligned}
$$

Thus, we have

$$
Z_j^{(s+1)} | \Phi, \Gamma, \pi \propto \begin{cases} \frac{\pi}{n_1} w_{ij} & z_j \leq n_1; \\ 1 - \pi & z_j = n_1 + 1. \end{cases}
$$

We integrate over π and rearrange terms to produce the final full conditional:

$$
p\left(Z_j^{(s+1)} = i | \Phi, \boldsymbol{Z^{(s)}} \right) \propto \begin{cases} w_{ij} & i \leq n_1; \\ n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\pi}{n_{12}(\boldsymbol{Z}) + \alpha_\pi} & i = n_1 + 1. \end{cases}
$$

3.2 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 **?**, in which $\hat{Z}_j \in \{1, \ldots, n_A + 1, R\}$, with R

representing the option to leave the matching undetermined by the model. Specifically, we have

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

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.

We seek to minimize the posterior expected loss, given by

$$\mathbb{E}[L(\hat{\boldsymbol{Z}}, \boldsymbol{Z})|\Gamma^{obs}] = \sum_{\boldsymbol{Z}} L(\hat{Z}_j, Z_j) P(\boldsymbol{Z}|\Gamma^{obs}).$$

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

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

For a more in-depth explanation of this function and the induced Bayes estimate, see **?**.

Since the Gibbs sampler 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 simply 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 $P(Z_j = i|\Gamma) > \frac{1}{2}$ and $P(Z_{j'} = i|\Gamma) > \frac{1}{2}$, 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 **?** to match records to latent entities.

4 Efficient and Scalable Implementation

To increase the speed of the Gibbs sampler and expand the scale of the linkage tasks we can undertake, we propose hashing methods similar to those used by **?** in the creation of `fastlink`, a fast and scalable implementation of the Fellegi Sunter method. The key insight is to recognize that record pairs contribute to posterior calculations only through the agreement pattern of the γ_{ij} vector. 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 orignial 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 Fellegi Sunter comparison vectors, to aid in vectorized computations. See Appendix 7.2 for details.

4.1 Data Representation, Hashing, and Storage

First, we hash record pairs of the same agreement pattern to unique integer values. **?** 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 $(i,j) \in h_p$.

With all record pairs converted to hashed values, we can create a more compact nested list

$$\mathcal{R} = \left\{ \{r_{j_p}\}_{p=1}^{P} \right\}_{j=1}^{n_2}$$

where $r_{j_p} = \{i \in \boldsymbol{X_1} | (i,j) \in h_p\}$. This representation is useful because it allows us to easily compute sufficient statistics

$$H_{j_p} = \sum_i I((i,j) \in h_p) = ||r_{j_p}||$$

and

$$H_p = \sum_{i,j} I((i,j) \in h_p) = \sum_j H_{j_p}.$$

For convenience, denote this set of sufficient statistics $\mathcal{H} = \left\{ \{H_{j_p}\}, \{H_p\} \right\}$. 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 proceed with $\tilde{\Gamma} = \{\mathcal{P}, \mathcal{R}, \mathcal{H}\}$ for posterior inference.

4.2 Scaling to Large Linkage Tasks

The hashing procedure described above considerably reduces the memory needed to store the comparison information. That is, instead of storing $n_1 \times n_2$ comparison vectors, we only store the P unique vectors, and then $n_1 \times n_2$ scalar quantities relating record pairs to those vectors. However, even storing these $n_1 \times n_2$ scalar labels can become

burdensome with large databases. Additionally, the overwhelming majority of these labels relate to record pairs that are clear non-matches.

To address this issue, we propose storage efficient indexing (SEI). In standard indexing, one decides a priori certain criteria that all true matching pairs must satisfy, and labels any record pairs that do not meet that criteria as non-matches. 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. While generally chosen to be quite loose, establishing these criteria requires knowledge of the problem and invites room for human error.

With SEI, however, we can reduce the comparison space, and its associated storage costs, while avoiding these drawbacks. Observe that all records $i \in \boldsymbol{X}_1$ that share the same agreement pattern with $j \in \boldsymbol{x}_2$ have the same Fellegi Sunter weight w_{ij}, and therefore the same probability when sampling Z_j. Thus, we know that records $i \in r_{j_p}$ such that H_{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 listed $\mathcal{R}^{\text{SEI}}$. Rather than 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. 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 $\tilde{\Gamma}^{\text{SEI}} = \{\mathcal{P}, \mathcal{R}^{\text{SEI}}, \mathcal{H}\}$. For simplicty, we suppress "SEI" in future notation, and use $\tilde{\Gamma}$ without loss of generality.

For large data, we can partition the two datasets $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ into smaller blocks $\{\boldsymbol{X}_{1m}\}$ and $\{\boldsymbol{X}_{2n}\}$ for more manageable computations. On a single machine, we can read-in data sequentially, conduct hashing, compress information through SEI, and delete the original data 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 then be easily synthesized to recover sufficient statistics for the larger linkage task. Thus, the combination of hashing, SEI, and partitioning allows us to conduct linkage tasks over much larger datasets.

4.3 Efficient Posterior Inference

Calculated at the level of the record pairs, updating $\alpha_{fl}(\boldsymbol{Z})$ and $\beta_{fl}(\boldsymbol{Z})$ for each field and level in the linkage task constitutes $2 \times \sum L_f$ many summations over $n_A \times n_B$ 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 $H_p^m = \sum_{j=1}^{n_B} \mathbf{1}_{(Z_j, j) \in h_p}$ to be the number of matching record pairs with agreement pattern p. 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 prior parameters for the $\boldsymbol{m}$ and $\boldsymbol{u}$ distributions respectively, the posterior update becomes

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

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

These constitute $\tilde{L}$ summations over P quantities, where $\tilde{L} = \sum_{f=1}^{F} L_f$ is the total number of levels used across all features in the linkage task.

Although sampling Z_j from a the full conditional provided in Section 3.1 conceptually straightforward, it becomes computationally burdensome when n_1 is large. This is because sampling a value from n_1 many 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. Second, we sample the record label uniformly among records associated with that agreement pattern for that particular $j \in B$. More concretely, define $h(Z_j)$ to be the agreement pattern between j and its potential match, and say $h(Z_j) = h_{P+1}$ when $Z_j = n_1 + 1$. Then,

$$P\left(h\left(Z_j^{(s+1)}\right) = p \mid \Phi, \boldsymbol{Z^{(s)}}\right) \propto \begin{cases} w_p \times H_{j_p} & p \leq P; \\ n_1 \frac{n_2 - n_{12}(\boldsymbol{Z}) + \beta_\lambda}{n_{12}(\boldsymbol{Z}) + \alpha_\lambda} & p = P + 1 \end{cases} \tag{7}$$

$$P\left(Z_j^{(s+1)} = i \mid h\left(Z_j^{(s+1)}\right) = p\right) = \begin{cases} \frac{1}{H_{j_p}} & (i,j) \in h_p \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Lastly, we recognize that all posterior updates are governed by the agreement patterns of the record pairs rather than the record labels themselves. Thus we complete the entire Gibbs sampler first at the level of the P agreement patterns with (7) above. After all iterations are complete, we can back-fill the records corresponding to the agreement patterns through (8).

We emphasize the computational gains of this split sampler: the first step is a sample from P 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 drastically improve the speed of the sampler, and each can be parallelized if desired for additional computational gains. We provide a summary of the `fabl` method through pseudocode in Appendix 7.1. We note here that this split sampler can be adapted for use with `BRL` as well, but would require updating $\{H_j\}$ to account for the previously matched records that no are longer candidate matches for the current record. `BRL` would see considerable speed up through the procedure, but computation time would still scale with the size of the larger dataset.

4.4 Computational Complexity

In this section, we prove the computational complexity of `fabl`.

Lemma 1. *The overall computational complexity of* `fabl` *is* $O(\frac{F}{B}n_1n_2) + O(n_2P)$.

Proof. To prove the computational complexity, we consider two steps — constructing the comparison vectors and the Gibbs sampler. The computational complexity of all pairwise comparisons across the two databases $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ is $O(Fn_1n_2)$. The hashing procedure for all pairwise comparisons is also of complexity $O(Fn_1n_2)$. 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 $O(\tilde{L}P)$. The complexity of updating 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 $\tilde{L} << n_1$ in all reasonable 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. The distributions used (see Table 1) are meant to emulate the rates of agreement across first name, last name, day of birth, and month of birth. For example, $u^{\text{month, agree}} = P(\text{Records have same birth-month} \mid \text{Nonmatch}) = \frac{1}{12}$. The parameters for year would vary largely by context, so we choose them here to adjust the desired difficulty of the linkage task. For simplicity, we consider only exact matching, so a vector (1, 0) corresponds to agreement and (0, 1) to disagreement. We simulate these data for different values of n_1 and n_2, and compare the run-time of `fabl` against `BRL`. Since we have 5 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 (**?**). In contrast, we use non-optimized code written in R for `fabl`. While this complicates

	m	u
fname	$\left(\frac{19}{20}, \frac{1}{20}\right)$	$\left(\frac{1}{100}, \frac{99}{100}\right)$
lname	$\left(\frac{19}{20}, \frac{1}{20}\right)$	$\left(\frac{1}{100}, \frac{99}{100}\right)$
day	$\left(\frac{19}{20}, \frac{1}{20}\right)$	$\left(\frac{1}{30}, \frac{29}{30}\right)$
month	$\left(\frac{19}{20}, \frac{1}{20}\right)$	$\left(\frac{1}{12}, \frac{11}{12}\right)$
year	$\left(\frac{19}{20}, \frac{1}{20}\right)$	$\left(\frac{1}{12}, \frac{11}{12}\right)$

Table 1: Distributions used for m and u probabilties in simulation studies

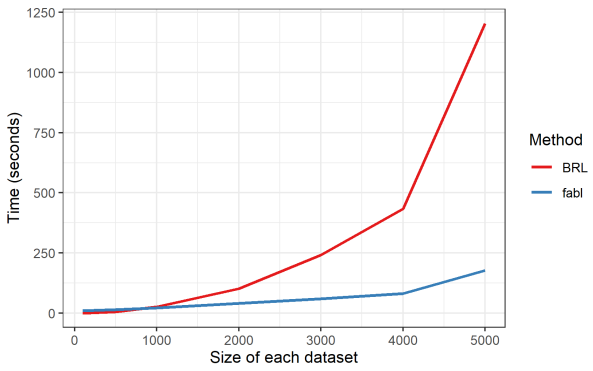

Figure 1: Run-time for BRL and fabl to run 1000 Gibbs iterations, including hashing step for fabl, for increasing values of both n_A and n_B. We see near quadratic growth in run-time for BRL, and near linear growth for fabl.

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 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, `BRL` is faster than `fabl` for low sample sizes, but that `fabl` is significantly faster at handling larger data. In particular, run-time for `BRL` seems to grow quadratically (or linearly with the size of both $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$) while run-time for `fabl` seems to grow linearly (in the size of $\boldsymbol{X}_2$).

The above discussion suggests that for fixed n_2, computation time for `fabl` should remain mostly constant with growing n_1. In the simulation results displayed in Figure 2 fixing $n_2 = 500$, we see linear growth for the run-time under `BRL` as n_1 increases, with 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.

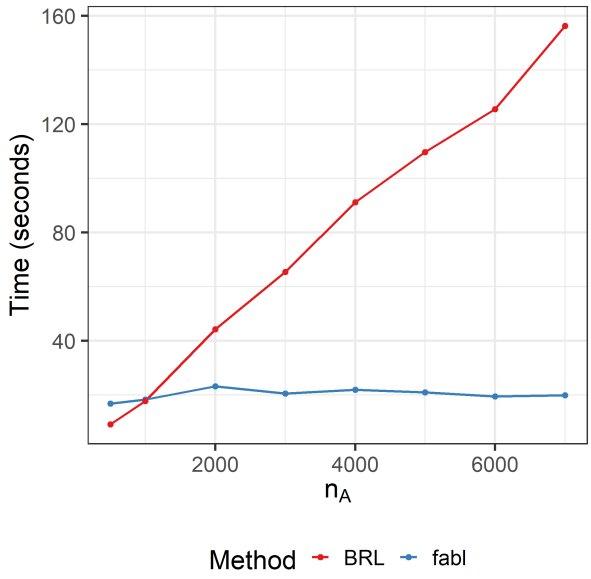

Figure 2: Run-time for BRL and fabl to run 1000 Gibbs iterations, including hashing step for fabl, with increasing n_A, and n_B fixed at 500. We see linear growth in run-time for BRL, and near constant run-time for fabl.

5.2 Accuracy under Full Estimates

We replicate a simulation study from **?** to compare `fabl` against `BRL` on several synthetic datasets with varying amounts of error and overlap (the number of records in common across files). Following methods proposed by **?** and **?**, 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 2. 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. 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 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 7.3; they show no obvious concern over MCMC convergence.

		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 2: Construction of comparison vectors for accuracy study with simulated datasets.

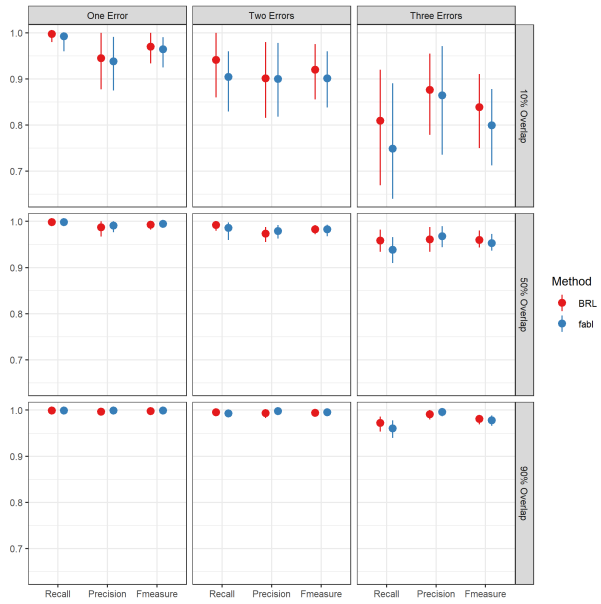

Figure 3: Posterior means and credible intervals for accuracy metrics under the replication of simulation study from **?**. For each level of overlap and each level of error, we have 100 paired sets of 500 records. We see comparable performance for all levels of error and overlap.

We compare `fabl` to `BRL` in terms of recall, precision and F-measure, as defined in **?**. 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 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.

5.3 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 report the rejection rate (RR), or how often the model declines to make a linkage decision, defined as RR $= \sum_{j=1}^{n_2} I(\hat{Z}_j = R)$.

In Figure 4, 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.

6 Case Studies

In our first case study, we revisit data from the El Salvadoran Civil War analyzed by **?**. 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. Throughout the time, several organizations attempted to document casualties of the conflict. 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. This is an important step in estimating the total number of deaths from a conflict using multiple systems estimation (**?**). 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 reported directly

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

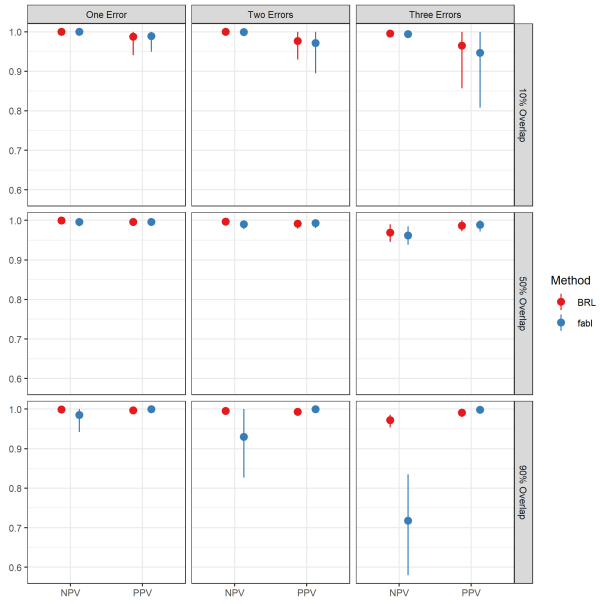

Figure 4: Negative predictive value (NPV) and positive predictive value (PPV) on simulated datasets. We see poorer performance for `fabl` only in situations with high overlap.

to the organization (**??**). 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.

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 **?**, 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 again use flat priors for the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters.

? constructs the comparison vectors using four levels of agreement for each field, according to the thresholds provided in Table 3. This results in $4^5 \times 2 = 2048$ possible

		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 3: Construction of comparison vectors for El Salvador data resembling original implementation from (**?**). 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 4: Construction of comparison vectors for El Salvador for increased speed under `fabl`. This set up leads to 216 possible agreement patterns in total.

agreement patterns, with 1173 patterns realized in the data. We initially used these comparisons to estimate fabl and BRL. However, after investigating the posterior distributions of the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters, we noticed that posterior distributions of several levels of the m and u parameters are nearly identical, suggesting that these levels are unnecessary for meaningfully modeling the distribution of the comparison vectors.

Therefore we re-ran our analysis with fewer agreement levels for each field according to Table 4. 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 BRL 239 second run time. The estimates of the $\boldsymbol{m}$ parameters under each method are very similar, as shown in Figure 6. Traceplots for parameters of interest are provided in Appendix 7.4.

For completeness, we note that the more detailed comparison vector produced a run time of 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 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

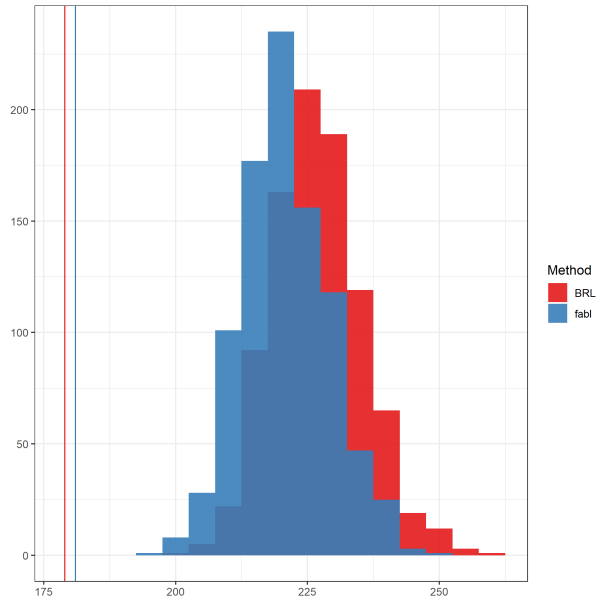

Figure 5: Posterior distribution and Bayes estimate of overlap across the two files. We note they are quite similar under both methods.

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 Section 5.2. 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. We note that Bayes estimates falling outside of posterior credible intervals has occurred previously in the record linkage literature (**??**), and remains a topic for future research.

We note that reconciling Bayes estimates that fall outside of posterior credible intervals is a common challenge in record linkage and an avenue of future work.

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 began in 1982, with follow-up surveys taken approximately every five years. For this study, we seek to link records over the

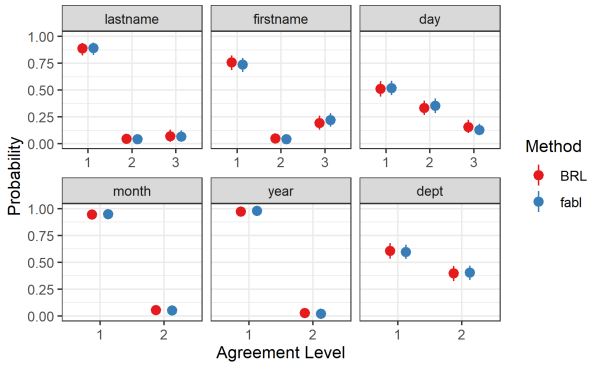

Figure 6: Posterior estimates of m parameters with 95% credible intervals for El Salvador case study. They are quite similar across the two methods.

$n_A = 20485$ records from 1982 to the $n_B = 17466$ records from 1989. The NLTCS data have longitudinal links, so that in reality one does not need to do a record linkage. However, following the strategy in **?** we break the longitudinal links and treat the data from 1982 and 1989 as stand-alone files.

We link records using sex, date of birth, and location using the thresholds shown in Table 5. Storing five 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. 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 indicate quick convergence, and are similar to those seen in Appendix 7.3 and 7.4

7 Conclusion

In this paper, we have proposed `fabl`, a Bayesian record linkage method that scales 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

		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 5: Construction of comparison vectors for NTLCS data

the other. We have demonstrated the speed and accuracy of `fabl` by replicating a simulation study and a case study in **?**, and through an additional case study that is computationally infeasible under `BRL`. Open source software to use `fabl` in `R` is available through Github.

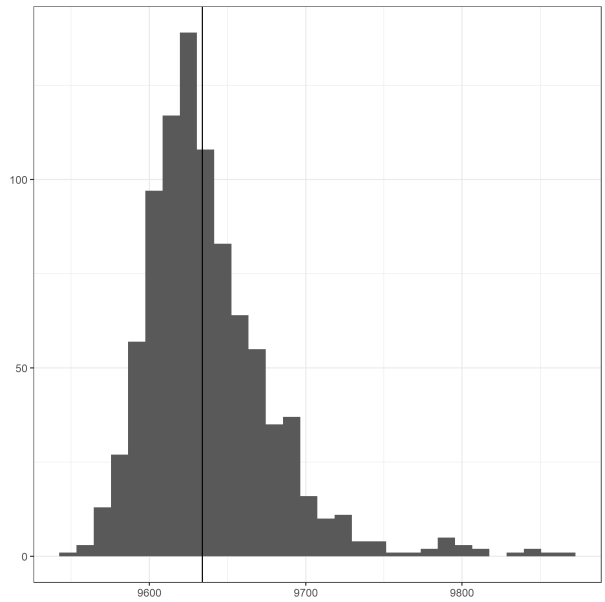

Figure 7: Posterior distribution and Bayes estimate of overlap across years 1982 and 1989 of NLTCS data.

Appendix

7.1 Summary of Fast Beta Linkage Method

procedure HASHING AND PREPROCESSING
  Construct and enumerate set of unique patterns $\mathcal{P}$ from F and $\{L_f\}$.
  Partition files $\boldsymbol{X_1}$ and $\boldsymbol{X_1}$ into chunks $\{\boldsymbol{X_{1n}}\}$, $\{\boldsymbol{X_{2m}}\}$.
  for each n, m **do**
    Create comparison vectors between $\boldsymbol{X_{1n}}$ and $\boldsymbol{X_{2m}}$.
    Hash records to $\mathcal{R}_{nm}$ and calculate summary statistics $\mathcal{H}_{nm}$.
    Use SEI to reduce memory usage; $\mathcal{R}_{nm} \rightarrow \mathcal{R}_{nm}^{\text{SEI}}$.
  end for
  Synthesize results across pairings to get $\tilde{\Gamma} = \{\mathcal{P}, \mathcal{R}, \mathcal{H}\}$.
end procedure
procedure GIBBS SAMPLING
  Initialize m, u, and Z parameters.
  for $t \in \{1, \ldots, T\}$ **do**
    Sample $\Phi^{t+1}|Z^t, \tilde{\Gamma}$.
    Sample $h\left(Z^{t+1}\right)|\Phi^{t+1}, \tilde{\Gamma}$. ▷ Sample agreement pattern, not record
  end for
  Sample $Z|h(Z), \tilde{\Gamma}$. ▷ Fills in record label based on agreement pattern
end procedure

7.2 One Hot Encoding Transformation

`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 $\gamma_{ij}^f = \ell$, 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 $f \in \{1, \ldots, F\}$, 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 6 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 6: Example records for one hot encoding.

7.3 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.

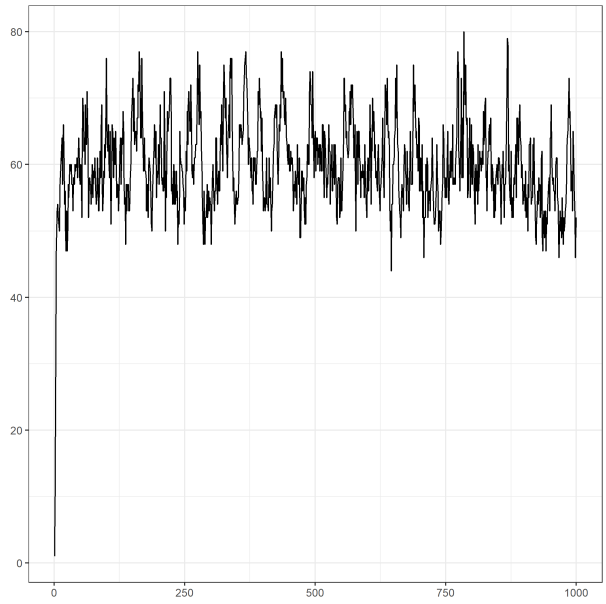

Figure 8: Representative traceplot of overlap between files from simulation study in Section 5.2

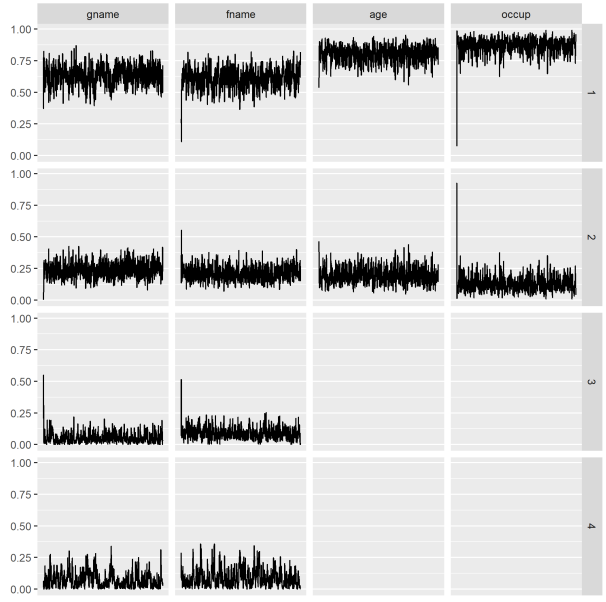

Figure 9: Representative traceplot of m parameter from simulation study in Section 5.2

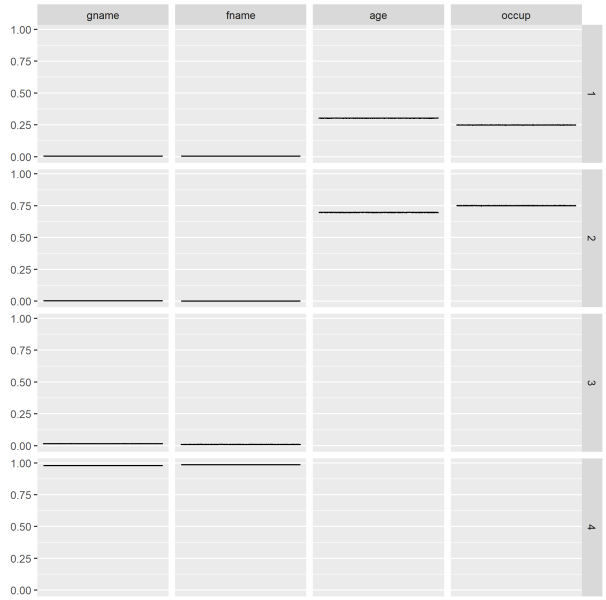

Figure 10: Representative traceplot of u parameters from simulation study in Section 5.2

7.4 Traceplots for El Salvador Case Study

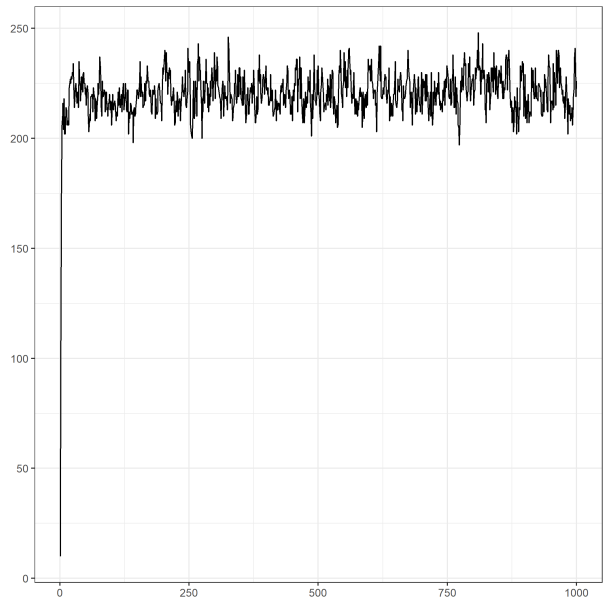

Figure 11: Traceplot for number of matches found across datasets in El Salvador case study

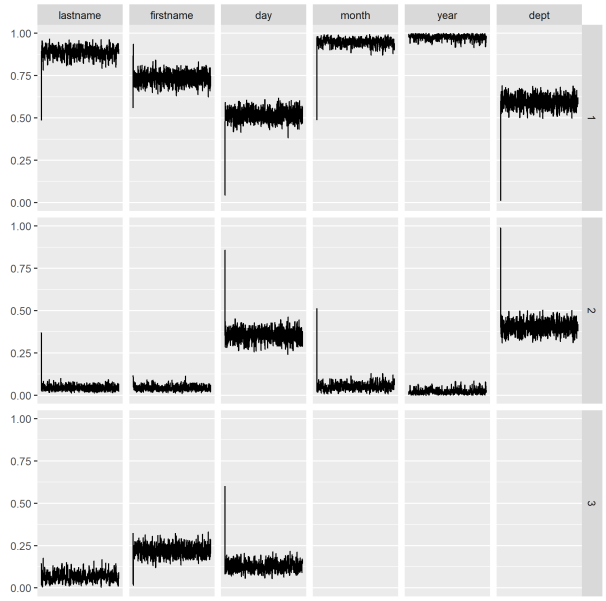

Figure 12: Traceplot for m parameter in El Salvador case study

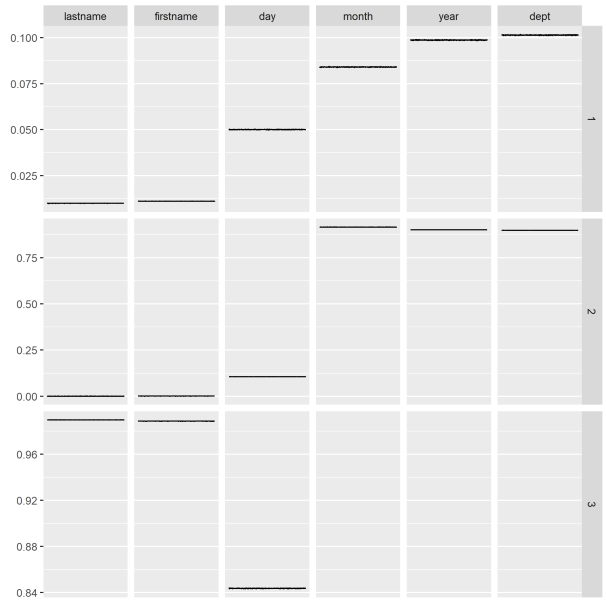

Figure 13: Traceplot for u parameter in El Salvador case study