Bayesian Analysis (2021) **TBA**, Number TBA, 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 difficulties, we propose fast beta linkage (`fabl`), an extension to the Beta Record Linkage (`BRL`) method of **?**. 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` can have markedly increased speed with minimal loss of accuracy when compared to `BRL`.

**Keywords:** data fusion, data cleaning, entity resolution, hashing, record linkage.

## 1 Introduction

Before conducting data analysis, it is often necessary 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 (e.g., **????**). 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 probabilistic record linkage methods rely on the seminal work of **?** and **?**. In their approach, the data analyst first creates comparison vectors for each pair of records in the data files. These vectors indicate how similar the records are on a set of variables measured in both files, known as the linkage variables. Using these comparison vectors, the analyst classifies each pair as a match or nonmatch using a likelihood ratio test. Crucially, these decisions are made independently for each pair. The **?** approach has been extended for a wide variety of applications (e.g., **??????**). An alternative paradigm is to model the linkage variables directly (e.g., **????**). In this article, we build on the contributions to the comparison vector approach.

The independent pairwise matching assumption from **?** is popular for its mathematical simplicity. However, in many situations, there are no duplications within a data file, meaning 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. Analysts typically use an additional post-processing step to turn the list of linked records into a bipartite matching (**?**), but this model misspecification can still lead to poor results (**?**).



*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

Alternatively, analysts can embed one-to-one matching requirements into the model specification, at an additional computational cost. **?** employed a Metropolis-Hastings algorithm to only allow sampling matches that respect 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 computationally intensive and, to our knowledge, has not been applied on data files with more than 100 records. **?** 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. `BRL` has been 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 article, we propose fast beta linkage (`fabl`), which extends the `BRL` model for increased efficiency and scalability. We use independent prior distributions for the matching statuses of each record, and modify the decision theoretic technique of **?** to ensure our linkage estimates are bipartite. This approach 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 similar levels 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 **?** and **?**. Section 3 proposes the `fabl` model, provides the Gibbs sampler for posterior inference, and describes 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 Fellegi-Sunter Approaches for Record Linkage

Consider two data files $A$ and $B$ comprising $n_A$ and $n_B$ records, respectively, and including $F$ linkage variables measured in both files. For $i = 1, \ldots, n_A$, let record $i$ be given by $A_i = (A_{i1}, \ldots, A_{iF})$, so that $A = (A_i : i = 1, \ldots, n_A)$. Similarly, for $j = 1, \ldots, n_B$, let record $j$ be given by $B_j = (B_{j1}, \ldots, B_{jF})$, so that $B = (B_j : j = 1, \ldots, n_B)$. Without loss of generality, denote files such that $n_A \geq n_B$.

Intuitively, matching records (those that refer to the same entity) should have similar values of the linking variables; records that are nonmatching should have dissimilar values. **?** proposed encoding this using a comparison vector $\gamma_{ij}$ computed for each record

pair $(i, j)$ in $A \times B$. Specifically, they define $\gamma_{ij} = (\gamma_{ij}^1, \ldots, \gamma_{ij}^F)$, where each $\gamma_{ij}^f$ is a value indicating the similarity of field $f$ for records $A_i$ and $B_j$. We define the comparison matrix $\gamma \in \mathbb{R}^{n_A n_B \times F}$ as the collection of all record pairs $\gamma_{ij}$.

When linking variable $f$ is categorical, a common way to define $\gamma_{ij}^f$ is an indicator for exact agreement. For example, if zip code is linking variable $f$, we can set $\gamma_{ij}^f = 1$ when the zip codes for records $A_i$ and $B_j$ agree exactly, and set $\gamma_{ij}^f = 2$ when they do not. For numerical linking variables, we can use the absolute difference of the two values. For example, if age is linking variable $f$, we can set $\gamma_{ij}^f = 1$ when the ages for records $A_i$ and $B_j$ match exactly, $\gamma_{ij}^f = 2$ when the ages are within one year but not equal, and $\gamma_{ij}^f = 3$ when the ages are two or more years apart. For text variables like names, we can use string distance metrics such as Levenstein or Jaro-Winkler distance (**?**). We then set thresholds that allow us to represent comparisons through discrete levels of disagreement (**??**).

More generally, let $\mathcal{S}_f(i, j)$ denote a similarity measure for linking variable $f$ of records $A_i$ and $B_j$. The range of $\mathcal{S}_f$ can be divided into $L_f$ intervals denoted by $I_{f1}, \ldots, I_{fL_f}$. Here, $I_{f1}$ represents the highest level of agreement (including 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 $A_i$ or $B_j$. As a result, the comparison vector $\gamma_{ij}$ will have missing values. We assume that this missingness occurs completely at random (MCAR, per **?**). To notate a missing value in any $\gamma_{ij}^f$, we use $I_{obs}(\gamma_{ij}^f) = 1$ when $\gamma_{ij}^f$ is observed and $I_{obs}(\gamma_{ij}^f) = 0$ otherwise. With the MCAR assumption, we can marginalize over the missing data, and do all computation simply using the observed data.

Having defined comparison vectors, we now turn to the **?** model for record linkage. We begin with notation for defining linkages. Under bipartite matching, the set of matches across $A$ and $B$ can be represented in two equivalent ways. First, we may use a matrix $\Delta \in \{0, 1\}^{n_A n_B}$, where

$$\Delta_{ij} = \begin{cases} 1, & \text{if records } A_i \text{ and } B_j \text{ refer to the same entity;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

This sparse matrix representation can become cumbersome for large linkage tasks. More compactly, bipartite matching also can be viewed as a labeling $\boldsymbol{Z} = (Z_1, \ldots, Z_{n_B})$ for the records in $B$ such that

$$Z_j = \begin{cases} i, & \text{if records } A_i \text{ and } B_j \text{ refer to the same entity;} \\ n_A + j, & \text{if record } B_j \text{ does not have a match in } A. \end{cases} \tag{2}$$

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. When presenting the models, we use both representations for convenience.

Let $\Gamma_{ij}$ represent a random variable for the comparison vector for arbitrary $A_i$ and $B_j$. Thus, $\gamma_{ij}$ is a realization of $\Gamma_{ij}$. For modeling the collection of $n_A n_B$ random variables $\Gamma_{ij}$, **?** employ two independence assumptions: first, that comparison vectors are independent given the matching status of the record pair, and second, that the matching status of each record pair is independent of the matching status of other pairs. Using these independence assumptions, one specifies a mixture model for each $\Gamma_{ij}$ (e.g., as in **????**). We have

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

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

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

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

**?** presents a Bayesian version of the model in (3a) through (3c). He uses uniform Dirichlet prior distributions for each $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$, and replaces (3c) with a prior distribution for $\boldsymbol{Z}$ that he calls the "beta distribution for bipartite matching." This assigns a Bernoulli distribution for the indicator that a record in $B$ has a match in $A$, that is, $I(Z_j \leq n_A) \sim \text{Bernoulli}(\pi)$. Additionally, $\pi \sim \text{Beta}(\alpha_\pi, \beta_\pi)$, where $\alpha_\pi$ and $\beta_\pi$ are known hyperparameters. It follows that the number of records in $B$ that have matches, denoted $n_{AB}(\boldsymbol{Z}) = \sum_{j=1}^{n_B} I(Z_j \leq n_A)$, is distributed according to a Beta-Binomial$(n_B, \alpha_\pi, \beta_\pi)$. Conditioning on the set of records in $B$ that have matches, formally denoted $\{I(Z_j \leq n_A)\}_{j=1}^{n_B}$, all $n_A!/(n_A - n_{AB}(\boldsymbol{Z}))!$ bipartite matchings are taken to be equally likely. Thus, the prior distribution used by **?** is given by

$$p(\boldsymbol{Z} | \alpha_\pi, \beta_\pi) = \frac{(n_A - n_{AB}(\boldsymbol{Z}))!}{n_A!} \frac{\text{B}(n_{AB}(\boldsymbol{Z}) + \alpha_\pi, n_B - n_{AB}(\boldsymbol{Z}) + \beta_\pi)}{\text{B}(\alpha_\pi, \beta_\pi)}, \tag{4}$$

where $\text{B}(\cdot, \cdot)$ 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 sequential, which can be slow when working on linkage tasks with more than a few thousand records.

# 3 Fast Beta Linkage

Instead of the prior over $\boldsymbol{Z}$ from **?**, we follow **?** and use independent priors for each component $Z_j$. However, unlike **?** who proposes a flat prior for $Z_j$, we use the fast beta linkage (`fabl`) prior below. For each $Z_j$, we have

$$p(Z_j = q | \pi) = \begin{cases} \frac{1}{n_A}\pi, & q \leq n_A; \\ 1 - \pi, & q = n_A + j; \end{cases} \tag{5}$$

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

We can interpret (5) as follows: record $B_j$ has some match in $A$ with probability $\pi$, and each record $A_i$ is equally likely to be that match. The hyperparameters $\alpha_\pi$ and $\beta_\pi$ encode prior beliefs about the proportion of records in $B$ that have matches in $A$.

In the **?** flat prior, each value $\{1, \ldots, n_A, n_A + j\}$ is a priori equally likely for $Z_j$. This amounts to a prior probability of $n_A/(n_A + 1)$ that record $B_j$ has a match in $A$. In our preliminary studies, we have found that this highly informative prior weighting on matching can result in overly high match rates. Hence, we prefer (5). We also note that the flat prior is equivalent to a special case of the fast beta prior with $\pi$ fixed at the mean of a Beta $(1, 1/n_A)$ random variable.

Linkage with `fabl` is conducted at the record level, rather than at the record pair level, as in the **?** model. That is, $\pi$ in (5) under `fabl` estimates the proportion of records in $B$ that have matches, whereas $\lambda$ in (3c) in the **?** model estimates the proportion of record pairs that are matches. In the bipartite case, we conjecture that some analysts will find $\pi$ to be a more interpretable parameter than $\lambda$. In this setting, there are at most $n_B$ matching pairs out of $n_A n_B$ total pairs, meaning that $\lambda$ is bounded above by $1/n_A$ and tends towards 0 as the size of the linkage task grows. Additionally, while the **?** model makes $n_A n_B$ independent matching decisions and `BRL` makes $n_B$ dependent matching decisions, `fabl` strikes a middle ground between the two, making $n_B$ 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.

For clarity, we present the full model for `fabl` below:

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

$$\boldsymbol{m}_f \sim \text{Dirichlet}(\alpha_{f1}, \ldots, \alpha_{fL_f}), \forall f = 1, \ldots, F, \tag{6b}$$

$$\boldsymbol{u}_f \sim \text{Dirichlet}(\beta_{f1}, \ldots, \beta_{fL_f}), \forall f = 1, \ldots, F, \tag{6c}$$

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

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

We estimate the posterior distribution of the parameters in (6a - 6e) using a Gibbs sampler. Supplement A presents derivations for the full conditional distributions. We initialize $\boldsymbol{m}$, $\boldsymbol{u}$ and $\pi$ from random draws from their prior distributions, and initialize $\boldsymbol{Z}$ to reflect no matches across data files; that is, $\boldsymbol{Z} = (n_A + 1, \ldots, n_A + n_B)$. Denote the current draw of the sampler by $(s)$ and the updated draw by $(s+1)$. To update $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$ for all $f = 1, \ldots, F$, we sample

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

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

$$\text{where } \alpha_{fl}(\boldsymbol{Z}^{(s)}) = \alpha_{fl} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(Z_j^{(s)} = i), \tag{7c}$$

$$\text{and } \beta_{fl}(\boldsymbol{Z}^{(s)}) = \beta_{fl} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(Z_j^{(s)} \neq i). \tag{7d}$$

Next, we sample $\pi$ from its full conditional, given by

$$\pi^{(s+1)} | \gamma, \boldsymbol{Z}^{(s)}, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)} \sim \text{Beta}(n_{AB}(\boldsymbol{Z}^{(s)}) + \alpha_\pi, n_B - n_{AB}(\boldsymbol{Z}^{(s)}) + \beta_\pi). \tag{8}$$

where $n_{AB}(\boldsymbol{Z}^{(s)}) = \sum_{j=1}^{n_B} I(Z_j^{(s)} \leq n_A)$.

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

$$p\left(Z_j^{(s+1)} = q | \gamma, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}\right) \propto \begin{cases} \frac{\pi^{(s+1)}}{n_A} w_{qj}^{(s+1)}, & q \leq n_A; \\ 1 - \pi^{(s+1)}, & q = n_A + j, \end{cases} \tag{9}$$

where, for all $i \in \{1, \ldots, n_A\}$ and $j \in \{1, \ldots, n_B\}$,

$$w_{ij}^{(s)} = \prod_{f=1}^{F}\prod_{l=1}^{L_f}\left(\frac{m_{fl}^{(s)}}{u_{fl}^{(s)}}\right)^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)}. \tag{10}$$

Finally, to obtain an estimate $\hat{\boldsymbol{Z}}$ of the linkage structure, we use the loss functions and Bayes estimate from **?**. Since (5) does not strictly enforce one-to-one matching, it is possible for this Bayes estimate to link multiple records in $B$ to one record in $A$. To obtain a Bayes estimate that fulfills the bipartite requirement, we minimize the expected loss subject to the constraint that $\hat{Z}_j \neq \hat{Z}_{j'}$ for all $j \neq j'$. See Supplement B for details redarding the initial Bayes estimate and this post-processing procedure.

# 4 Efficient and Scalable Implementation

The scale of linkage tasks possible through `BRL` is limited by the memory costs of storing $n_A n_B$ 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" (**?**). 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 often undesirable (**?**).

After computing all comparison vectors within a block, the modeler 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 last 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; **?** reviewed this issue in the context of the classical **?** 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 summary statistics that reduce the computational complexity of the Gibbs sampler and the memory requirements of storing the comparison vectors. Second, we introduce storage efficient indexing, which reduces the memory costs associated with unlikely matches.

## 4.1 Data Representation, Hashing, and Storage

Since each component $\gamma_{ij}^f$ is discrete, there are only finitely many possible realizations of the comparison vector $\gamma_{ij}$. Let $P$ be the number of patterns realized in $\gamma$. It is bounded above by $P^* = \prod_{f=1}^{F}(L_f + 1)$, where the addition of 1 to $L_f$ for each field accounts for the possibility of missing values. This quantity is determined by $F$ and $L_f$, and does not scale with $n_A$ or $n_B$.

To obtain a memory efficient representation, we map the agreement pattern of each record pair to a unique integer. **?** accomplished this through a hashing function, which we modify to explicitly handle missing values:

$$h^*(\gamma_{ij}) = \sum_{f=1}^{F} I_{obs}(\gamma_{ij}^f) 2^{\gamma_{ij}^f + I(f>1)\sum_{d=1}^{f-1}(L_d)}. \tag{11}$$

We then map the integers in (11) to sequential integers $\{1, \ldots, P\}$. Denote each unique agreement pattern as $h_p = (h_p^1, \ldots, h_p^F)$, and the set of unique agreement patterns as $\mathcal{P} = \{h_1, \ldots, h_P\}$. When the $(i, j)$ record pair exhibits agreement pattern $p$, we say $\gamma_{ij} = h_p$. In calculations, we will at times use the one-hot encoding of agreement pattern $h_p$, denoted $e(h_p)$. This is a vector of length $\sum_{f=1}^{F} L_f$ in which the $l + \sum_{k=1}^{f-1} L_k$ component is 1 when $\gamma_{ij}^f = l$, and 0 otherwise.

For example, consider five fields with binary agreements and possible missingness, denoted $NA$. The number of possible patterns is bounded above by $P^* = 3^5 = 243$. Suppose that records $A_5$ and $B_7$ exhibit agreement pattern $\gamma_{5,7} = (1, 1, 1, NA, 2)$, indicating exact agreement on the first three fields, missing information in the fourth field, and disagreement in the fifth field. The expression in (11) gives $h^*(\gamma_{5,7}) = 2^1 + 2^3 + 2^5 + 0 + 2^{10} = 1066$. All records with a hashed value of 1066 map to the integer 42. Thus, $\gamma_{5,7} = h_{42}$, and this agreement pattern has the one hot encoding $e(h_{42}) = (1, 0, 1, 0, 1, 0, 0, 0, 0, 1)$.

We then identify the records in $A$ with comparison vectors corresponding to each pattern $p$ for each record $B_j$. We denote this set $r_{p_j} = \{i \in 1, \ldots, n_A | \gamma_{ij} = h_p\}$, and collect all such sets in the nested list $\mathcal{R} = \{r_{p_j} | p \in \{1, \ldots, P\}, j \in \{1, \ldots, n_B\}\}$. We compute the number of records in $A$ that share agreement pattern $p$ with record $B_j$,

given by

$$N_{p_j} = |r_{p_j}| = \sum_{i=1}^{n_A} I(\gamma_{ij} = h_p). \tag{12}$$

We collect these counts in $\mathcal{N} = \{N_{p_j} | p \in 1, \ldots, P, j \in 1, \ldots, n_B\}$.

The set $\tilde{\gamma} = \{\mathcal{P}, \mathcal{R}, \mathcal{N}\}$ fully characterizes the comparison matrix $\gamma$ for writing the likelihood function for `fabl`. To see this, we employ the condensed notation

$$m_p = p(\Gamma_{ij} = h_p | Z_j = i) = \prod_{f=1}^{F} \prod_{l=1}^{L_f} m_{fl}^{I(h_p^f = l) I_{obs}(h_p^f)}, \tag{13a}$$

$$u_p = p(\Gamma_{ij} = h_p | Z_j \neq i) = \prod_{f=1}^{F} \prod_{l=1}^{L_f} u_{fl}^{I(h_p^f = l) I_{obs}(h_p^f)} \tag{13b}$$

to express the probability that records $A_i$ and $B_j$ form agreement pattern $p$ given that they are a match in (13a), and not a match in (13b). Viewed through the perspective of agreement patterns, the likelihood in (6a) is equivalent to

$$\mathcal{L}(\boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{u} \mid \tilde{\gamma}) = \prod_{j=1}^{n_B} \prod_{p=1}^{P} \prod_{i \in r_{p_j}} m_p^{I(Z_j = i)} u_p^{1 - I(Z_j = i)}. \tag{14}$$

The likelihood in (14) allows for more efficient posterior inference for $\boldsymbol{m}$, $\boldsymbol{u}$, and $\boldsymbol{Z}$, as we now describe.

## 4.2 Efficient Posterior Inference

The posterior updates for $\boldsymbol{m}$ and $\boldsymbol{u}$ depend on the data only through quantities that can be calculated through $\mathcal{N}$ and $\mathcal{P}$. Let $n_p(\boldsymbol{Z}) = \sum_{j=1}^{n_B} I\left(\gamma_{Z_j, j} = h_p\right)$ be the number of matching record pairs with agreement pattern $p$, and $N_p = \sum_{j=1}^{n_B} N_{p_j}$ be the total occurrence of pattern $p$ in the data across all record pairs. Then, conditional on $\boldsymbol{Z}$, we can express the contribution to the likelihood in the full conditional for $\boldsymbol{m}$ and $\boldsymbol{u}$ as

$$\mathcal{L}(\boldsymbol{m}, \boldsymbol{u} \mid \tilde{\gamma}, \boldsymbol{Z}, \pi) = \prod_{p=1}^{P} m_p^{n_p(\boldsymbol{Z})} u_p^{N_p - n_p(\boldsymbol{Z})}. \tag{15}$$

Additionally, let $\boldsymbol{\alpha_0} = (\alpha_{11}, \ldots, \alpha_{FL_F})$ and $\boldsymbol{\beta_0} = (\beta_{11}, \ldots, \beta_{FL_F})$ be concatenated vectors of prior parameters for the $\boldsymbol{m}$ and $\boldsymbol{u}$ distributions respectively. The terms needed for the posterior updates for the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters are given by the appropriate components of the vectors

$$\boldsymbol{\alpha}(\boldsymbol{Z}^{(s)}) = \boldsymbol{\alpha_0} + \sum_{p=1}^{P} n_p\left(\boldsymbol{Z}^{(s)}\right) \times e(h_p), \tag{16a}$$

$$\boldsymbol{\beta}(\boldsymbol{Z}^{(s)}) = \boldsymbol{\beta_0} + \sum_{p=1}^{P} \left(N_p - n_p\left(\boldsymbol{Z}^{(s)}\right)\right) \times e(h_p). \tag{16b}$$

Specifically, the $l + \sum_{k=1}^{f-1} L_k$ components of (16a) and (16b) provide the posterior updates for level $l$ and field $f$ in (7c) and (7d). However, the vectorized summations can be computed more efficiently than summing over $n_A n_B$ record pairs for each field and agreement level.

Similarly, the posterior updates for $\boldsymbol{Z}$ depend on the data only through quantities calculated through $\mathcal{N}, \mathcal{P}$, and $\mathcal{R}$. For each pattern $p$, let $w_p = m_p/u_p$. The contribution to the likelihood in the full conditional for $Z_j$ can be expressed as

$$\mathcal{L}(Z_j \mid \tilde{\gamma}, \boldsymbol{m}, \boldsymbol{u}, \pi) = \prod_{p=1}^{P} u_p^{N_{p_j}} \prod_{i \in r_{p_j}} w_p^{I(Z_j = i)}. \tag{17}$$

Sampling $Z_j$ from the full conditional provided in (9) can become computationally expensive when $n_A$ is large. This is because sampling a value from $n_A$ options with unequal weights requires normalizing the weights to probabilities, which has a computational cost that scales with $n_A$. To speed up computation, we use (17) to break this sampling step into two.

We first sample among $P + 1$ options for the agreement pattern between $B_j$ and its potential link, according to

$$p\left(\Gamma_{Z_j^{(s+1)},j} = h_p \mid \tilde{\gamma}, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}\right) \propto \begin{cases} \frac{\pi^{(s+1)} N_{p_j}}{n_A} w_p^{(s+1)}, & h_p \in \mathcal{P}; \\ 1 - \pi^{(s+1)}, & \text{otherwise.} \end{cases} \tag{18}$$

Since all records in $A$ sharing the same agreement pattern with $B_j$ are equally likely, we then sample among candidate records uniformly using

$$p\left(Z_j^{(s+1)} = q \mid \gamma_{Z_j^{(s+1)},j} = h_p, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}\right) = \begin{cases} \frac{1}{N_{p_j}}, & q \in r_{p_j}; \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

These changes can greatly improve the speed of the sampler, and each can be parallelized if desired for additional computational speed-ups. We emphasize the computational gains of this split sampler through Lemma 1.

**Lemma 1.** *Let $n_A$ and $n_B$ be the number of records in files $A$ and $B$, 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 $A \times B$. We assume $C$ cores available for parallelization and a Gibbs sampler with $T$ iterations. Then, the overall computational complexity of* `fabl` *with hashing is $O(Fn_An_B/C) + O(Tn_BP/C)$.*

*Proof.* We consider two steps: constructing the comparison vectors and the Gibbs sampler. The computational complexity of all pairwise comparisons across $A$ and $B$ is $O(Fn_An_B)$.

The hashing procedure for all pairwise comparisons is also $O(Fn_An_B)$. With $C$ processors available, we can split these computations across $C$ equally sized partitions and compute these comparisons in parallel, so the complexity becomes $O(Fn_An_B/C)$. There are then trivial computational costs associated with synthesizing summary statistics across these partitions. We note that this contribution to computational complexity applies for all versions of **?** algorithms, unless they use blocking to reduce the comparison space.

Without hashing, the computational complexity of updating the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters is $O(Fn_An_B)$. 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 as in **?** is $O(n_An_B)$. With hashing, we first sample the agreement pattern of the match with complexity $O(n_BP)$, and then we sample the record exhibiting that pattern with complexity $O(n_B)$. Thus, the complexity of sampling $\boldsymbol{Z}$ in a single iteration is $O(n_BP)$. Since $P << n_A$ in most applications, we have reduced the complexity of sampling $\boldsymbol{Z}$ from $O(Fn_An_B)$ under `BRL` to $O(n_BP)$ under `fabl`. With parallelization, this complexity is further reduced to $O(n_BP/C)$, and so the entire Gibbs sampler has complexity $O(Tn_BP/C)$ In summary, the total computational complexity for `fabl` is $O(Fn_An_B/C) + O(Tn_BP/C)$. □

## 4.3 Scaling to Large Linkage Tasks

For linkage tasks with large amounts of records, we can partition $A$ and $B$ into $t_A$ and $t_B$ smaller disjoint batches for more manageable computations. Let $\{A^1, \ldots, A^{t_A}\}$ be a partition of $A$ such that $\cup_{a=1}^{t_A} A^a = A$ and $A^a \cap A^{a'} = \emptyset$ for all $a \neq a'$. Likewise, let $\{B^1, \ldots, B^{t_b}\}$ be a partition of $B$ such that $\cup_{b=1}^{t_B} B^b = B$ and $B^b \cap B^{b'} = \emptyset$ for all $b \neq b'$. For each $a$ and $b$, we compute comparison vectors for all records in $A^a \times B^b$ to construct the comparison matrix $\gamma^{ab}$.

We then conduct hashing, obtain the compressed $\tilde{\gamma}^{ab}$, and delete the memory intensive matrix $\gamma^{ab}$ before continuing with the next batch of data. In detail, we calculate

$$r_{p_j}^{ab} = \{i \in 1, \ldots, n_A | \gamma_{ij} = h_p, B_j \in B^b\}, \tag{20a}$$

$$N_{p_j}^{ab} = |r_{p_j}^{ab}|. \tag{20b}$$

These can computed sequentially or in parallel. Summary statistics from each pairwise batch comparison can be synthesized to recover summary statistics for the full comparison matrix $\gamma$ through

$$r_{p_j} = (r_{p_j}^{11}, \ldots, r_{p_j}^{t_A t_B}) \text{ for } a = 1, \ldots, t_A \text{ and } b = 1, \ldots, t_B, \tag{21a}$$

$$N_{p_j} = \sum_{a=1}^{t_A} \sum_{b=1}^{t_B} N_{p_j}^{ab}. \tag{21b}$$

## 4.4 Storage Efficient Indexing

As discussed in Section 4.1, storing the indices, patterns, and counts in $\tilde{\gamma}$ uses less memory than storing the full comparison matrix $\gamma$. However, recording the indices for

all record pairs in $\mathcal{R}$ can become computationally burdensome for very large linkage tasks. We next introduce storage efficient indexing (SEI), which, when used with the methods in Section 4.3, allows us to compute $\mathcal{N}$ for all $n_A n_B$ record pairs while greatly reducing the memory costs of $\mathcal{R}$ associated with unlikely matches. This allows all-to-all comparisons for substantially larger linkage tasks.

All records $A_i$ that share agreement pattern $p$ with record $B_j$ have the same $w_p$. These records have the same probability to be identified as the link for record $B_j$. Thus, records $i \in r_{p_j}$ with large $N_{p_j}$ are unlikely to be sampled consistently enough to be deemed a match in the Bayes estimate. Rather than store all of these record labels, we store only a small number $S$. For each $r_{p_j}^{ab}$, we sample $S$ indices without replacement to form $\text{SEI}(r_{p_j}^{ab})$. We collect these memory reduced lists to form $\text{SEI}(r_{p_j})$ as in (21a), and collect these to form $\text{SEI}(\mathcal{R})$.

As shown in the full conditionals in (16a), (16b), (18), and (19), all original record pairs are still accounted for through $\mathcal{N}$, and thus we can proceed with posterior inference with the memory reduced $\text{SEI}(\tilde{\gamma}) = \{\mathcal{P}, \text{SEI}(\mathcal{R}), \mathcal{N}\}$. We provide guidance on choice of $S$ through a simulation in Section 5.3.

# 5 Simulation Studies

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

## 5.1 Speed

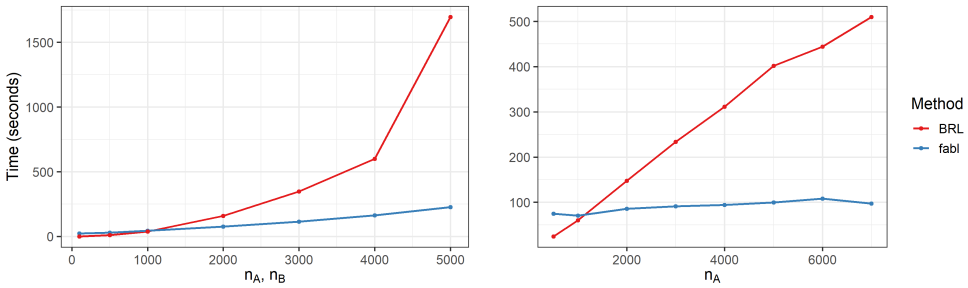
In our first simulation, we generate comparison vectors from pre-specified distributions so that we can easily increase the size of the linkage problem. We use $F = 5$ binary comparisons with probabilities for matching and nonmatching pairs shown in Table 1. For each record in $B$, we simulate $n_A$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

In Figure 1a, where we increase both $n_A$ and $n_B$, `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 $A$ and $B$) while run-time for `fabl` grows linearly (in the size of only $B$).

In Figure 1b, where we fix $n_B = 500$, we see near linear growth for the run-time under `BRL` as $n_A$ 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

| | $\boldsymbol{m}$ | | $\boldsymbol{u}$ | |
|---|---|---|---|---|
| | 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 1: Probabilities used for $\boldsymbol{m}$ and $\boldsymbol{u}$ in simulation study in Section 5.1.




(a) (b)

Figure 1: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations in simulations described in Section 5.1. In (1a), both $n_A$ and $n_B$ are increasing. We see quadratic growth in `BRL` and linear growth in `fabl`. In (1b), only $n_A$ only is increasing. We see linear growth in `BRL` and approximately constant run-time in `fabl`.

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 Supplement E.

Importantly, `BRL` implements a Gibbs sampler that is coded C (**?**), while `fabl` currently uses non-optimized code written only in `R`. 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 parallelization for the hashing step and sampling the matching status of the record pairs should lead to even more computational gains.

## 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 replicating a simulation study from **?**. 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 2: Construction of comparison vectors for accuracy study with simulated data files of Section 5.2.

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. These synthetic data files are available in the supplement to **?**.

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 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 use 100 pairs of simulated data files in order to obtain uncertainty quantification on our performance metrics. We use uniform priors for the $\boldsymbol{m}$, $\boldsymbol{u}$, and $\pi$ 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 Supplement B. Traceplots for parameters of interest for one example simulation are provided in Supplement C; 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; those results are in Supplement D.

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, $\sum_{j=1}^{n_B} I(\hat{Z}_j = Z_j, Z_j \leq n_A) / \sum_{j=1}^{n_B} I(Z_j \leq n_A)$. Precision is the proportion of links found by the model that are true matches, that is, $\sum_{j=1}^{n_B} I(\hat{Z}_j = Z_j, Z_j \leq n_A) / \sum_{j=1}^{n_B} I(\hat{Z}_j \leq n_A)$. The F-measure balances the two metrics to provide an overall measure of accuracy, and is defined as $2(\text{Recall} + \text{Precision})/(\text{Recall} \times \text{Precision})$. In Figure 2, we see that the two methods have comparable performance at all levels of error and overlap.

## 5.3 SEI Sensitivity

Finally, our last simulation demonstrates the robustness of `fabl` to different values of $S$ for the SEI memory reduction procedure. We perform record linkage on one set of the synthetic data files described in Section 5.2 with 500 records in each data file, 250 entities in common across data files, and 3 errors present across matching records. We perform SEI without batching the data, that is, $t_A = t_B = 1$. In practice, when it is computationally feasible to create and store $\gamma$ without batching, there is no need to

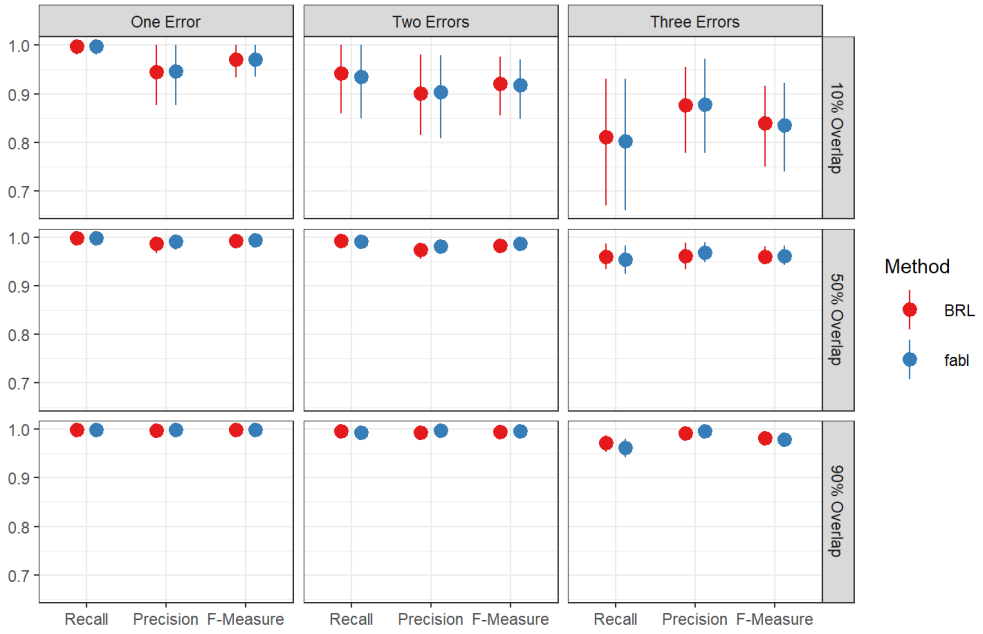



Figure 2: Posterior means and credible intervals for accuracy metrics under the replication of the simulation study from **?**. For each level of overlap and each level of error, we have 100 paired sets of 500 records. Thus this table summarizes results for 900 data files. We see comparable performance for all levels of error and overlap.

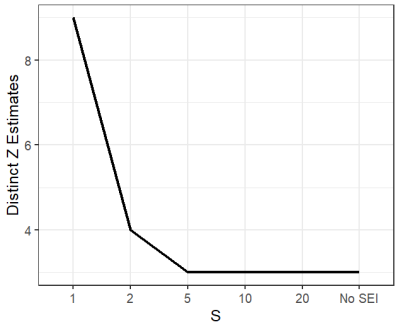



Figure 3: Distinct values of $\hat{Z}$ in the simulations of Section 5.3.

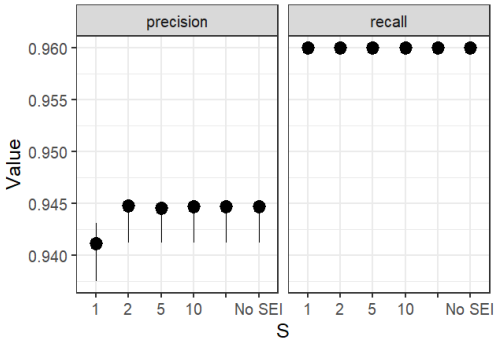



Figure 4: Means and 95% credible intervals for recall and precision in the simulations of Section 5.3.

reduce the memory of the hashed matrix through SEI. For illustration, however, it is easier to examine the effects of choices of $S$ in this setting.

We perform linkage using SEI with $S \in (1, 2, 5, 10, 20)$, and without using SEI, always with 500 iterations of the Gibbs sampler. Any particular SEI implementation may improve or worsen linkage performance; if the SEI procedure happens to only remove pairs that are not matches, recall and precision will improve. Therefore, we perform linkage under each setting 100 times, recording the linkage estimate $\hat{\boldsymbol{Z}}$, and recall and precision.

In Figure 3, the largest number of distinct linkage estimates occurs when $S = 1$. In this case, the SEI procedure arbitrarily removes large numbers of record labels from consideration, resulting in a noisier estimate of the linkage structure. The number of distinct linkage estimates decreases as $S$ increases, with larger values of $S$ providing results more similar to the linkage without SEI. In Figure 4, we see similar patterns in precision. Setting $S = 1$ can arbitrarily remove the index of a true match, leading the Gibbs sampler to concentrate probability on a false match, while larger values of $S$ produce results mirroring implementation with no SEI. We note, however, that even with $S = 1$, the loss in precision is small in these simulations.

Although the figures suggest that $S = 2$ is adequate for maintaining linkage performance, we suggest a more conservative value like $S = 10$. When evaluating the performance of a record linkage algorithm, researchers often examine posterior probabilities. By concentrating probability mass on arbitrary nonmatches, low values of $S$ may induce artificially high posterior probability for certain record pairs, providing a misleading perception of model performance.

# 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, 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. We are interested in estimating the total number of individuals killed in the war. We utilize lists of documented deaths 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 killings reported directly to the organization (**???**). The ERTL required additional investigation before recording denunciations as human rights abuses, and reports to the CDHES were made shortly after the events occurred; thus, both data files are thought to be fairly reliable. When estimating the total number of individuals killed, 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 (**?**).

There are several challenges with these data. First, the ERTL data file was automatically digitized, which inherently leads to some degree of typographical error. Second, it is common for villages in El Salvador to consist of only four to five extended families. This means there are a small number of last names in use, so many individuals have the same first and last name. Since the only fields recorded are given name, last name, date of death, and place of death, this leads to several instances in which distinct individuals have identical records. These individuals may be distant cousins, or are perhaps entirely unrelated, but would pose challenges for any record linkage method.

Following **?**, we utilize records that have non-missing entries for given and last name, which results in $n_A = 4420$ records in CDHES and $n_B = 1323$ 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 CDHES 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 $\boldsymbol{m}$, $\boldsymbol{u}$, and $\pi$ parameters.

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

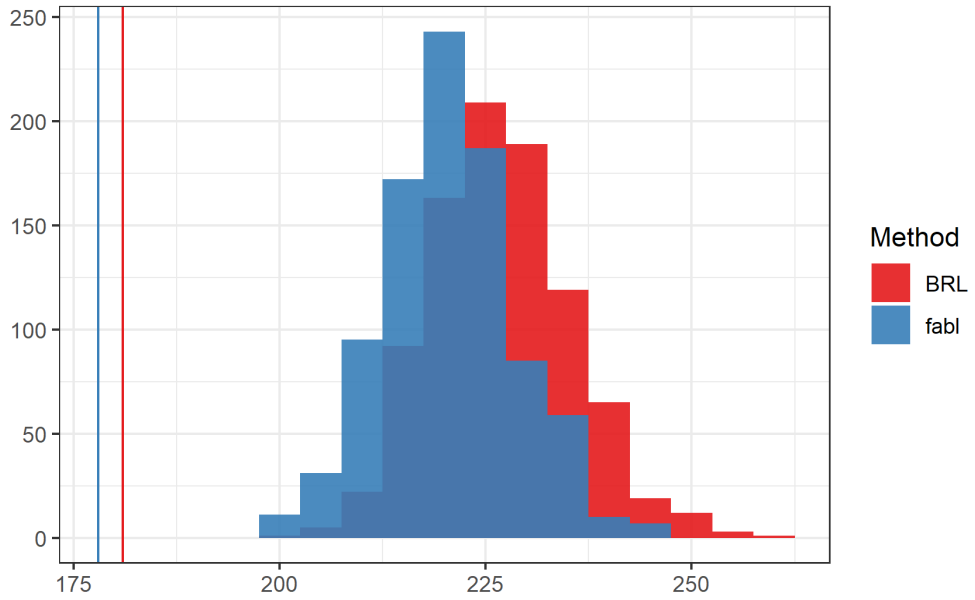



Figure 5: Posterior distribution and Bayes estimate of overlap across the two files in the El Salvador case study. We note they are quite similar under both methods.

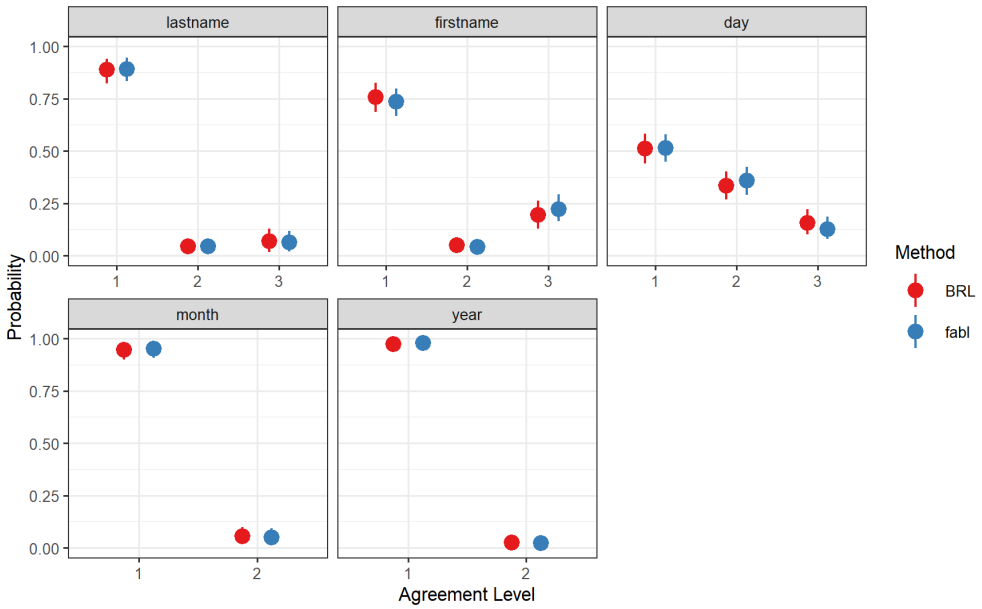



Figure 6: Posterior estimates of $\boldsymbol{m}$ parameters with 95% credible intervals for the El Salvador case study. They are quite similar across the two methods.

| | | 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 setup leads to 1875 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 data for increased speed under `fabl`. This setup leads to 432 possible agreement patterns in total.

We initially followed the comparison vector constructions set by **?**, using four levels of agreement for each field, according to the thresholds provided in Table 3. This results in 1875 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 4. Among the 432 possible agreement patterns, 159 are realized in the data. With this revised comparison specification, `fabl` runs in 109 seconds, approximately 3 times faster than the `BRL` run time of 313 seconds. The estimates of the $\boldsymbol{m}$ parameters under each method are similar, as shown in Figure 6. Estimates of the $\boldsymbol{u}$ parameters are indistinguishable, and thus omitted. Traceplots for parameters of interest are provided in Supplement F.

For completeness, we note that linkage with the more detailed comparison vectors requires 945 seconds for `BRL`, and 1093 seconds for `fabl`. Apparently, the number of 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 178 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 (205, 236), 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

| | | 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.

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 5 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 (**??**), 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 (**?**). 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 records over the $n_A = 20485$ individuals from 1982 to the $n_B = 17466$ 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 **?**, we break the longitudinal links and treat the data from 1982 and 1989 as stand-alone data files.

We link records using sex, location, and day, month, and year of birth using the criteria shown in Table 5. Storing $\gamma$ constructed through three comparison scores for each of $20485(17466) \approx 400$ million 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 method described in Section 4.3, we perform 30 smaller comparison tasks, using $t_A = 1$ and $t_B = 30$. We conduct linkage with all record indices recorded and also with SEI using $S = 10$, and obtain identical results. The hashed $\tilde{\gamma}$ without SEI is about 2.2 GB, and with SEI, it is about 760 MB. 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. Traceplots do not suggest convergence issues, and are similar to those seen in Supplement C and F. As shown in Figure 7, the Bayes estimate of the linkage structure has of 9634 matches, with a 95% credible interval of (9581, 9740). Since we have access to the true linkage

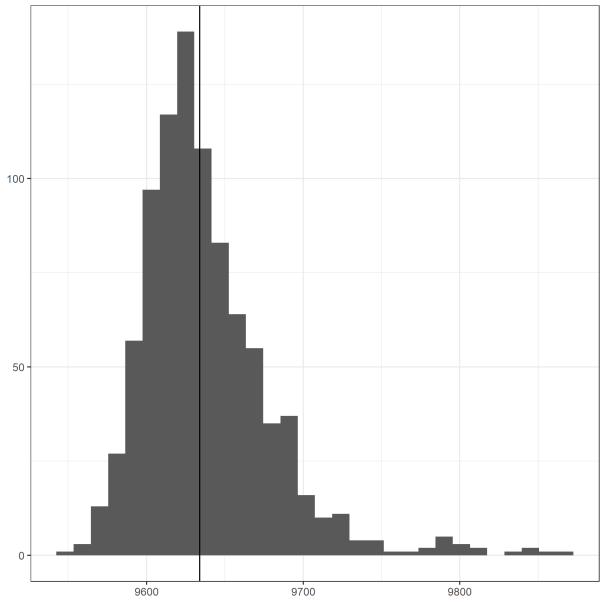



Figure 7: Posterior distribution and Bayes estimate of overlap across years 1982 and 1989 of NLTCS data.

structure, we can calculate recall to be 0.89 and precision to be 0.98, resulting in an F-measure of 0.94.

# 7 Conclusion

In this paper, we have proposed `fabl`, a Bayesian record linkage method that extends the work of **?** 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 **?**, 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 the comparisons for all $n_A n_B$ 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 even larger linkage tasks.

# Appendix A: Derivations of Full Conditionals

We provide detailed derivations of the full-conditionals provided in Section 3. The $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters are updated through standard multinomial-Dirichlet distributions. For a particular $m_{fl}$, we have

$$\mathcal{L}(m_{fl}|\gamma, \boldsymbol{u}, \boldsymbol{Z}, \pi) \propto \prod_{i=1}^{n_A}\prod_{j=1}^{n_B} m_{fl}^{I(Z_j=i)I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)} m_{fl}^{\alpha_{fl}-1} = m_{fl}^{\alpha_{fl}(\boldsymbol{Z})-1}, \tag{22}$$

where $\alpha_{fl}(\boldsymbol{Z}) = \alpha_{fl} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B} I_{obs}(\gamma_{ij}^f)I(\gamma_{ij}^f = l)I(Z_j = i)$. Analogous procedures lead to $\mathcal{L}(u_{fl}|\gamma, \boldsymbol{m}, \boldsymbol{Z}, \pi) \propto u_{fl}^{\beta_{fl}(\boldsymbol{Z})-1}$, where $\beta_{fl}(\boldsymbol{Z}) = \beta_{fl} + \sum_{i=1}^{n_A}\sum_{j=1}^{n_B} I_{obs}(\gamma_{ij}^f)I(\gamma_{ij}^f = l)I(Z_j \neq i)$. Thus, for the vectors of parameters $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$, we have

$$\boldsymbol{m}_f^{(s+1)} \mid \gamma, \boldsymbol{Z}^{(s)}, \boldsymbol{u}^{(s)}, \pi^{(s)} \sim \text{Dirichlet}(\alpha_{f1}(\boldsymbol{Z}^{(s)}), \ldots, \alpha_{fL_f}(\boldsymbol{Z}^{(s)})), \tag{23}$$

$$\boldsymbol{u}_f^{(s+1)} \mid \gamma, \boldsymbol{Z}^{(s)}, \boldsymbol{m}^{(s)}, \pi^{(s)} \sim \text{Dirichlet}(\beta_{f1}(\boldsymbol{Z}^{(s)}), \ldots, \beta_{fL_f}(\boldsymbol{Z}^{(s)})). \tag{24}$$

Since $\pi$ encodes the rate of matching across the two data files, the full conditional $p(\pi|\gamma, \boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{u}, \alpha_\pi, \beta_\pi)$ depends only on the number of links $n_{AB}(\boldsymbol{Z}) = \sum_{i=1}^{n_B} I(Z_j \leq n_A)$ encoded by $\boldsymbol{Z}$ and hyperparameters. We have the full conditional

$$p(\pi|\gamma, \boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{u}) \propto p(\boldsymbol{Z}|\pi)p(\pi) \tag{25}$$

$$\propto \pi^{n_{AB}(\boldsymbol{Z})}(1-\pi)^{n_B - n_{AB}(\boldsymbol{Z})}\pi^{\alpha_\pi - 1}(1-\pi)^{\beta_\pi - 1} \tag{26}$$

$$\propto \pi^{n_{AB}(\boldsymbol{Z})+\alpha_\pi - 1}(1-\pi)^{n_A - n_{AB}(\boldsymbol{Z})+\beta_\pi - 1}. \tag{27}$$

Thus, $\pi^{(s+1)}|\gamma, \boldsymbol{Z}^{(s)}, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}$ has a Beta$(n_{AB}(\boldsymbol{Z}^{(s)}) + \alpha_\pi, n_B - n_{AB}(\boldsymbol{Z}^{(s)}) + \beta_\pi)$ distribution.

Due to the independence in the fast beta prior in (5), we can obtain the full conditional for $\boldsymbol{Z}$ through the full conditionals for each individual $Z_j$. Let $\Gamma_{\cdot j}$ denote the random matrix of $n_A$ comparison vectors relating to an arbitrary record $B_j$, and let $\gamma_{\cdot j}$ be a realization of $\Gamma_{\cdot j}$. We have

$$p(\boldsymbol{Z}|\gamma, \boldsymbol{m}, \boldsymbol{u}, \pi) = \prod_{j=1}^{n_B} p(Z_j|\gamma_{\cdot j}, \boldsymbol{m}, \boldsymbol{u}, \pi). \tag{28}$$

Following the observation of **?**, when $B_j$ does not link to any record in $A$, the contribution to the likelihood is simply a product of $u$ parameters, which we will call $c_j$:

$$p(\Gamma_{\cdot j}|\boldsymbol{m}, \boldsymbol{u}, \pi, Z_j = n_A + j) = \prod_{i=1}^{n_A}\prod_{f=1}^{F}\prod_{l=1}^{L_f} u_{fl}^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)} = c_j. \tag{29}$$

When $Z_j = q$ for some $q \leq n_A$, we have

$$p(\Gamma_{\cdot j}|\boldsymbol{m}, \boldsymbol{u}, \pi, Z_j = q) = \prod_{f=1}^{F}\prod_{l=1}^{L_f} m_{fl}^{I(\gamma_{qj}^f=l)I_{obs}(\gamma_{qj}^f)} \prod_{i\neq q}\prod_{f=1}^{F}\prod_{l=1}^{L_f} u_{fl}^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)}. \tag{30}$$

We multiply and divide by the $u$ parameters for the matching record pair to obtain

$$p(\Gamma_{\cdot j}|\boldsymbol{m},\boldsymbol{u},\pi,Z_j=q)=\prod_{f=1}^{F}\prod_{l=1}^{L_f}\left(\frac{m_{fl}}{u_{fl}}\right)^{I(\gamma_{qj}^f=l)I_{obs}(\gamma_{qj}^f)}\prod_{i=1}^{n_A}\prod_{f=1}^{F}\prod_{l=1}^{L_f}u_{fl}^{I(\gamma_{ij}^f=l)I_{obs}(\gamma_{ij}^f)} \tag{31}$$

$$=w_{qj}c_j. \tag{32}$$

We can divide the result of each case by $c_j$ to get

$$p(\Gamma_{\cdot j}|\boldsymbol{m},\boldsymbol{u},\pi,Z_j)\propto\begin{cases}w_{qj}, & q\leq n_A;\\ 1, & q=n_A+j.\end{cases} \tag{33}$$

Lastly, we multiply the likelihood by the fast beta prior in (5) to obtain the full conditional

$$p\left(Z_j^{(s+1)}=q|\gamma,\boldsymbol{m}^{(s+1)},\boldsymbol{u}^{(s+1)},\pi^{(s+1)}\right)\propto\begin{cases}\frac{\pi^{(s+1)}}{n_A}w_{qj}^{(s+1)}, & q\leq n_A;\\ 1-\pi^{(s+1)}, & q=n_A+j.\end{cases} \tag{34}$$

# Appendix B: 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 $\hat{Z}_j\in\{1,\ldots,n_A,n_A+j,R\}$, with $R$ representing the option to leave the matching undetermined by the model. Specifically, we have

$$L(\hat{Z}_j,Z_j)=\begin{cases}0, & \text{if } Z_j=\hat{Z}_j;\\ \theta_R, & \text{if } \hat{Z}_j=R;\\ \theta_{10}, & \text{if } Z_j\leq 1,\hat{Z}_j=n_A+j;\\ \theta_{01}, & \text{if } Z_j=n_A+j,\hat{Z}_j\leq n_A;\\ \theta_{11}, & \text{if } Z_j\leq n_A,\hat{Z}_j\leq n_A,Z_j\neq\hat{Z}_j.\end{cases} \tag{35}$$

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

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

$$\hat{Z}_j=\begin{cases}i, & \text{if } p(Z_j=i|\gamma)>\frac{1}{2};\\ 0, & \text{otherwise.}\end{cases} \tag{36}$$

Since `fabl` does not strictly enforce one-to-one matching, it is possible for this Bayes estimate to link multiple records in $B$ to one record in $A$. In the event that we have two records $B_j$ and $B_{j'}$ such that both $p(\hat{Z}_j=i|\gamma)>\frac{1}{2}$ and $p(\hat{Z}_{j'}=i|\gamma)>\frac{1}{2}$, we accept the

match with the higher posterior probability, and declare the other to have no match. Since each $Z_j$ is independent, this is equivalent to minimizing the expected loss subject to the constraint that $\hat{Z}_j \neq \hat{Z}_{j'}$ for all $j \neq j'$. A similar approach appears in the most probable maximal matching sets used by **?** to match records to latent entities.

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

# Appendix C: Traceplots for Simulation Study

Figures 8, 9, and 10 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 $\boldsymbol{u}$ parameters show very little variation because the overwhelming majority of record pairs are nonmatching.

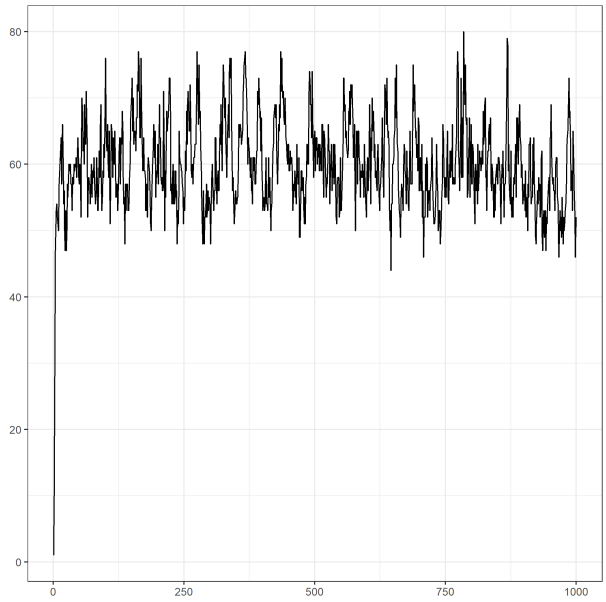



Figure 8: Representative traceplot of overlap between files from simulation study in Section 5.2.

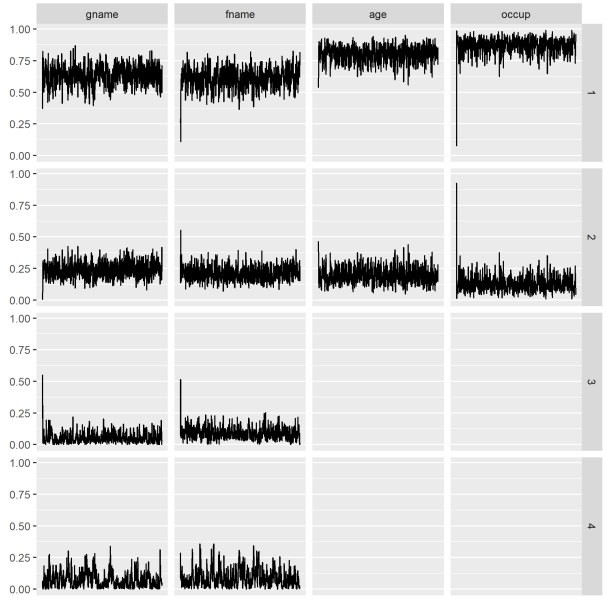



Figure 9: Representative traceplot of $\boldsymbol{m}$ parameter from simulation study in Section 5.2.

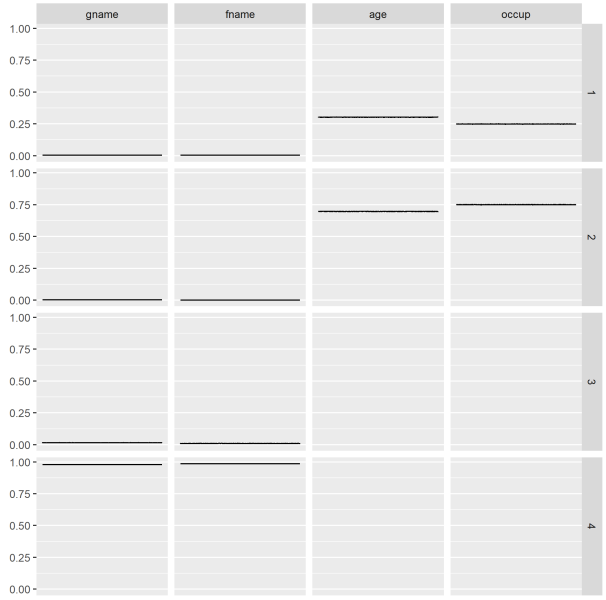



Figure 10: Representative traceplot of $\boldsymbol{u}$ parameters from simulation study in Section 5.2.

# Appendix D: Accuracy under Partial Estimates

In this section, we repeat the simulation study in Section 5.2, allowing for clerical review rather than forcing all records to have or not have links. Specifically, 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 nonmatches. Mathematically, NPV $= \sum_{j=1}^{n_B} I(\hat{Z}_j = Z_j = n_A + j) / \sum_{j=1}^{n_B} I(\hat{Z}_j = n_A + j)$. We continue to use the precision, which is renamed the positive predictive value (PPV) in this context. Lastly, we also examine the rejection rate (RR), or how often the model declines to make a linkage decision, defined as RR $= \sum_{j=1}^{n_B} I(\hat{Z}_j = R)/n_B$. To convey this information alongside NPV and PPV, for which values close to 1 indicate strong performance, we report the decision rate (DR), defined as DR $= 1 - RR$.

In Figure 11, 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 modeler to match by hand than would be left under `BRL`, but the decisions made by each method should have nearly equal accuracy.

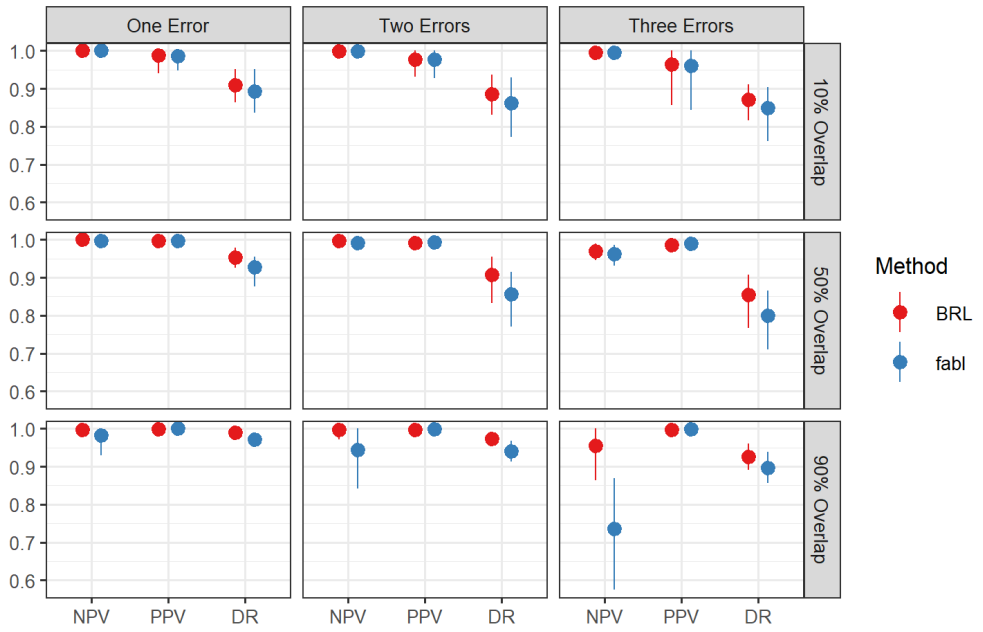


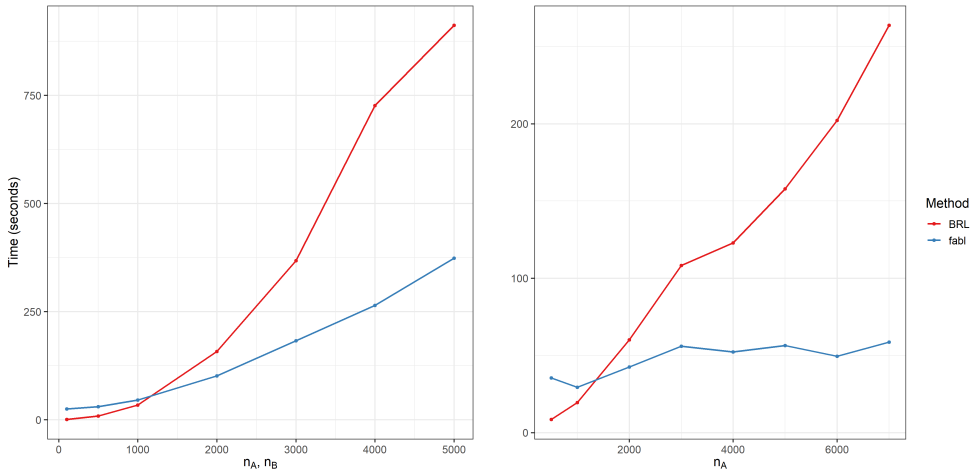

Figure 11: Negative predictive value (NPV), positive predictive value (PPV), and decision rate (DR) on data files in the simulation in Supplement D. We see poorer performance for `fabl` only in situations with high overlap.

# Appendix E: Additional Speed Simulation Study

Figures 12a and 12b illustrate that different constructions of the comparison vectors lead to similar speed gains. We replicate the speed study of Section 5.1 under different settings. Here, we use four fields of comparison, each with three possible levels of agreement, resulting in $3^4 = 81$ possible patterns. The $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters for this simulation are shown in Table 6.

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

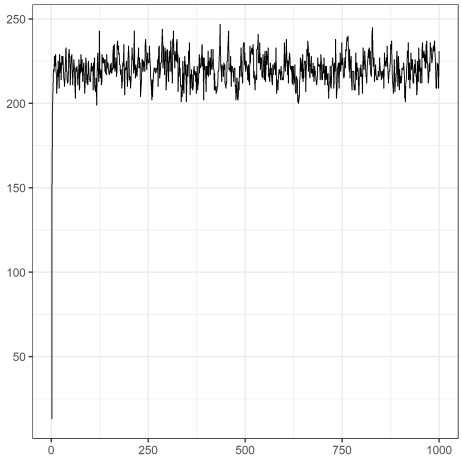
Table 6: Probabilities used for $\boldsymbol{m}$ and $\boldsymbol{u}$ distributions in simulation study in Supplement E.



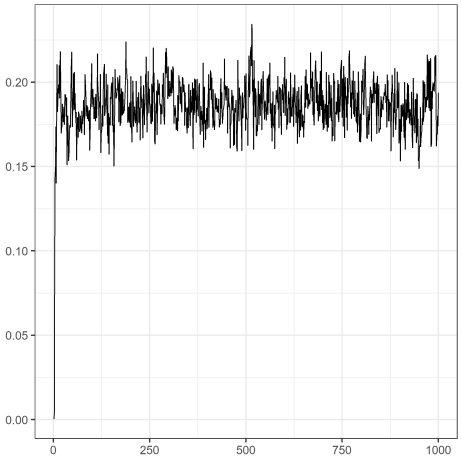

(a) (b)

Figure 12: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations in simulations described in Supplement E. In (12a), both $n_A$ and $n_B$ are increasing. We see quadratic growth in `BRL` and linear growth in `fabl`. In (12b), only $n_A$ only is increasing. We see linear growth in `BRL` and approximately constant run-time in `fabl`.

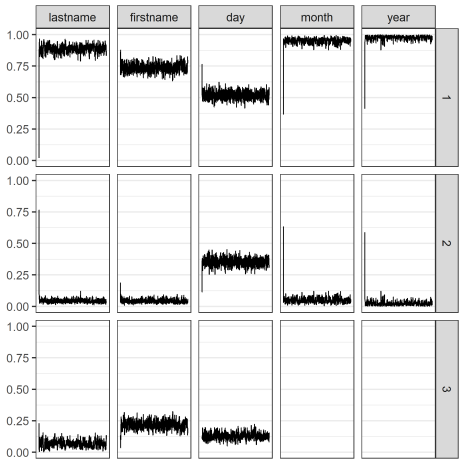
# Appendix F: Traceplots for El Salvador Case Study



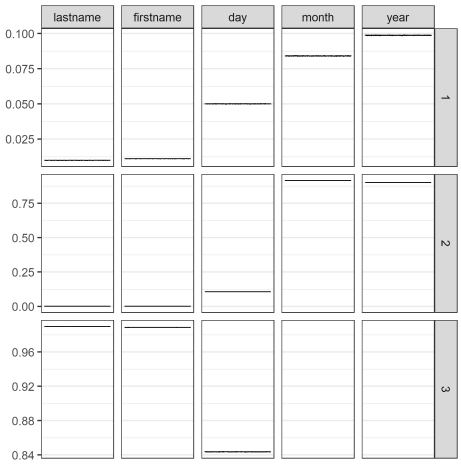

(a) Traceplot for overlap between files.




(b) Traceplot for $\pi$.




(c) Traceplot for $\boldsymbol{m}$ parameters.




(d) Traceplot for $\boldsymbol{u}$ parameters.

Figure 13: Traceplots for parameters of interest in El Salvador case study in Section 6.1.