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

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

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

Abstract. Within the field of record linkage, Bayesian methods have the crucial advantage of quantifying uncertainty from imperfect linkages. However, current implementations of Bayesian Fellegi-Sunter models are computationally intensive, making them challenging to use on larger-scale record linkage tasks. To address these computational considerations, we propose fast beta linkage (`fabl`), an extension to the Beta Record Linkage (`BRL`) method of Sadinle (2017). Specifically, we use independent prior distributions over the matching space, which allows for parallel updates to the Gibbs sampler, unlike the sequential updates of `BRL`. We ensure fabl results in a bipartite matching using decision theoretic post-processing techniques from the existing literature. To reduce computational overhead and memory costs, we use hashing and a new technique that we call 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: record linkage, entity resolution, data cleaning, parallelization, hashing.

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 (Christen, 2012; Gutman et al., 2013; Dalzell and Reiter, 2018; Tang et al., 2020). In this article, we consider bipartite record linkage, which merges two data files that contain duplications across, but not within, the respective data files.

Many probabilistic record linkage methods are rely on the seminal work of Fellegi and Sunter (1969) and Newcombe et al. (1959). 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 Fellegi and Sunter (1969) approach has been extended for a wide variety of applications (e.g., Winkler and Thibaudeau, 1990; Fair, 2004; Wagner et al., 2014; Gill and Goldacre, 2003; Enamorado et al., 2019; Aleshin-Guendel and Sadinle, 2023). An alternative paradigm is to model the linkage variables directly (e.g., Tancredi et al., 2011; Steorts et al., 2016; Marchant et al., 2021; Betancourt et al., 2022). In this article, we build on the contributions to the comparison vector approach.

The independent pairwise matching assumption from Fellegi and Sunter (1969) is popular mainly for its mathematical simplicity. However, in many situations, there are

*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

 DOI: 0000

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. To address this potential problem, data analysts typically use an additional post-processing step, for example, applying an optimization routine that turns the list of linked records into a bipartite matching (Jaro, 1989) or using decision theoretic procedures, such as those of Sadinle (2017) and Steorts et al. (2016), which both ensure bipartite (or rather transitive) matches.

Alternatively, analysts can embed one-to-one matching requirements into the model specification, at an additional computational cost. Larsen (2005) employed a Metropolis-Hastings algorithm to only allow sampling matches that respected one-to-one assumptions, but such algorithms exhibit slow mixing due to the combinatorial nature of the constrained matching space. Fortunato (2010) used simulated annealing to target the space of matches permitted under the one-to-one constraint, but the method is computationally intensive and, to our knowledge, has not been applied on data files with more than 100 records. Sadinle (2017) proposed the Beta Record Linkage model (`BRL`), using a prior distribution over the space of bipartite matchings to strictly enforce one-to-one requirements throughout a Gibbs sampler. Additionally, he introduced a class of loss functions that allows for a flexible estimation of the linkage structure, such that the modeler can weigh the relative importance of false positives and false negatives, and identify records pairings to be decided through clerical review. `BRL` 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 status of each record, an idea suggested in Wortman (2019). This independence assumption allows for a scalable update of the Gibbs sampler, unlike the sequential updates of Sadinle (2017). We employ the decision theoretic technique of Sadinle (2017) in order to ensure our method after record linkage is 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 Fellegi and Sunter (1969) and Sadinle (2017). Section 3 proposes the `fabl` model, provides the Gibbs sampler for posterior inference, and shows the loss function used to calculate the Bayes estimate for the bipartite matching. Section 4 introduces the hashing technique and storage efficient indexing used to increase the speed of calculations and the scale of linkage tasks amenable to `fabl`. Sections 5 and 6 demonstrate the speed and accuracy of `fabl` through simulation studies and case studies of homicides from the El Salvadoran Civil War and the National Long Term Care Study. Finally, Section 7 summarizes our contributions and highlights areas for further research.

2 Review of 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. Fellegi and Sunter (1969) proposed encoding this using a comparison vector γ_{ij} computed for each record pair (i, j) in $A \times B$. Specifically, they define a *comparison vector* $\gamma_{ij} = (\gamma_{ij}^1, \ldots, \gamma_{ij}^f, \ldots \gamma_{ij}^F)$, where each γ_{ij}^f is a value indicating the similarity of field f for records A_i and B_j. For notational convenience, we define the *comparison matrix* $\gamma \in \mathbb{R}^{n_A n_B F}$ as the collection of all record pairs' γ_{ij}.

When linking variable f is categorical, a common way to define γ_{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 (Cohen et al., 2003). We then set thresholds that allow us to represent comparisons through discrete levels of disagreement (Bilenko and Mooney, 2006; Elmagarmid et al., 2007).

More generally, let $\mathcal{S}_f(i, j)$ denote a similarity measure for 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.

Having defined comparison vectors, we now turn to the Fellegi and Sunter (1969) 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 Γ_{ij} represent a random variable for the comparison vector for arbitrary A_i and B_j. Thus, γ_{ij} is a realization of Γ_{ij}. For modeling the collection of $n_A n_B$ random variables Γ_{ij}, Fellegi and Sunter (1969) employ two independence assumptions: first, that comparison vectors are conditionally independent given their matching status, and second, that the matching status of the record pairs are independent. Using these independence assumptions, one specifies a mixture model for each Γ_{ij} (e.g., as in Winkler, 1999; Jaro, 1989; Larsen and Rubin, 2001; Enamorado et al., 2019). 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 λ is the marginal probability that a record pair is a match. When using comparison vectors with discrete agreement levels, $\mathcal{M}$ and $\mathcal{U}$ are collections of independent multinomial distributions for each linkage feature. Accordingly, $\boldsymbol{m} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_F)$, where $\boldsymbol{m}_f = (m_{f1}, \ldots, m_{fL_f})$ and $m_{fl} = p(\gamma_{ij}^f = l | \Delta_{ij} = 1)$ for all fields f and agreement levels l. The $\boldsymbol{u}$ parameters are defined similarly, with $u_{fl} = p(\gamma_{ij}^f = l | \Delta_{ij} = 0)$.

Sadinle (2017) presents a Bayesian version of the model in (3a) through (3c). He uses uniform Dirichlet prior distributions for each $\boldsymbol{m}_f$ and $\boldsymbol{u}_f$. However, he replaces (3c) with a prior distribution for $\mathbf{Z}$ that he calls the "beta distributions 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 α_π, β_π are known hyper-parameters. 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 Sadinle (2017) 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

For fabl, we also use (3a) and (3b), along with Dirichlet prior distributions for the $\mathbf{m}$ and $\mathbf{u}$ parameters. However, instead of the prior over $\boldsymbol{Z}$ from Sadinle (2017), we follow Wortman (2019) and use independent priors for each component Z_j. However, unlike Wortman (2019) who proposes a flat prior for Z_j, we propose an alternative, called the fast beta linkage (fabl) prior. For each Z_j, we have

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

$$\pi \sim \text{Beta}(\alpha_\pi, \beta_\pi). \tag{5b}$$

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

Remark 1. *In the Wortman (2019) 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, this highly informative prior weighting on matching can result in overly optimistic match rates. Hence, we prefer (5a). We also note that the flat prior is equivalent to a special case of the fast Beta prior with π fixed at the mean of a Beta$(1, 1/n_A)$ random variable.*

Remark 2. *Linkage with* `fabl` *is conducted at the record level, rather than at the record pair level, as in the Fellegi-Sunter model. That is, π under* `fabl` *estimates the proportion of records in B that have matches, whereas λ in the Fellegi and Sunter (1969) model estimates the proportion of record pairs that are matches. In the bipartite case, we conjecture that some analysts will find π in (5b) to be more a interpretable parameter than λ in (3c). In this setting, there are at most n_B matching pairs out of $n_A n_B$ total pairs, meaning that λ is bounded above by $1/n_A$ and tends towards 0 as the size of the linkage task grows. Additionally, while the Fellegi-Sunter 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.*

In many record linkage tasks, the records in A or B have missing fields, so that some γ_{ij}^f could be missing. To handle this, we follow the approach described in Sadinle (2017). Let $I_{obs}(\gamma_{ij}^f) = 1$ when γ_{ij}^f is observed and $I_{obs}(\gamma_{ij}^f) = 0$ otherwise. We assume that any missing comparisons are missing at random (MAR) or missing completely at random (MCAR). For likelihood based inference, both MAR and MCAR lead to ignorability, or rather, being able to consider the likelihood for the observed data (Little and Rubin, 2002, Section 6.2), so that we can marginalize over the missing data and perform all computation using the observed data (Sadinle, 2014, 2017).

We summarize the model for fabl as follows:

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

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

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

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

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

where recall that (α_π, β_π) are fixed hyper-parameters.

We now present a Gibbs sampler to approximate the joint posterior distribution of $\boldsymbol{Z}, \boldsymbol{m}$, $\boldsymbol{u}$, and π given the observed comparison data from the likelihood and priors assumed in (6a - 6e). Refer to Appendix A for derivations of the full conditional distributions. We initialize $\boldsymbol{m}$ and $\boldsymbol{u}$ from random draws from their prior distributions, and initialize $\boldsymbol{Z}$ to reflect no matches across data files; that is, $\boldsymbol{Z} = (n_A+1, \ldots, n_A+n_B)$. Brian: state how you initialize π. Denote the current draw of the sampler by s and the updated draw by $s+1$. To update $\mathbf{m}$ and $\mathbf{u}$, for all, $f = 1, \ldots, F$, sample

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

and

$$\boldsymbol{u}_f^{(s+1)} | \gamma, \boldsymbol{Z}^{(s)} \sim \text{Dirichlet}(\beta_{f1}(\boldsymbol{Z}^{(s)}), \ldots, \beta_{fL_f}(\boldsymbol{Z}^{(s)})), \tag{8}$$

where

$$\alpha_{fl}(\boldsymbol{Z}^{(s)}) = \alpha_{fl} + \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(Z_j^{(s)} = i), \quad \text{and} \tag{9}$$

$$\beta_{fl}(\boldsymbol{Z}^{(s)}) = \beta_{fl} + \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} I_{obs}(\gamma_{ij}^f) I(\gamma_{ij}^f = l) I(Z_j^{(s)} \neq i). \tag{10}$$

Next, we sample π from its full conditional, given by

$$\pi^{(s+1)} | \boldsymbol{Z}^{(s)}, \alpha_\pi, \beta_\pi, \pi \sim \text{Beta}(n_{AB}(\boldsymbol{Z}^{(s)}) + \alpha_\pi, n_B - n_{AB}(\boldsymbol{Z}^{(s)}) + \beta_\pi). \tag{11}$$

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:

$$p\left(Z_j^{(s+1)} = q | \gamma, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}\right) \propto \begin{cases} \frac{\pi^{(s+1)}}{n_A} w_{qj}^{(s+1)}, & q \leq n_A; \\ 1 - \pi^{(s+1)}, & q = n_A + j, \end{cases} \tag{12}$$

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

$$w_{ij}^{(s)} = \prod_{f=1}^{F} \prod_{l=1}^{L_f} \left(\frac{m_{fl}^{(s)}}{u_{fl}^{(s)}} \right)^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)} . \tag{13}$$

The full conditional in (12) does not depend on $\boldsymbol{Z}^{(s)}$, implying that

$$p(\boldsymbol{Z}^{(s+1)} \mid \gamma, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}) \propto \prod_{q} \left(\prod_{j} p\left(Z_j^{(s+1)} = q \mid \gamma, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)} \right) \right) . \tag{14}$$

Thus, we can make parallel updates to $\boldsymbol{Z}$ instead of sequential updates as in Sadinle (2017). In fact, to our knowledge, our approach and that of Wortman (2019) are the only two Bayesian approaches that propose parallel updates to the matching label (or linkage structure), in contrast to other Bayesian record linkage methods (Tancredi et al., 2011; Sadinle, 2014; Steorts et al., 2016; Marchant et al., 2021; Zanella et al., 2016; Betancourt et al., 2022; Aleshin-Guendel and Sadinle, 2023).

Finally, to obtain an estimate $\hat{\boldsymbol{Z}}$ of the linkage structure, we use the loss functions and Bayes estimate from Sadinle (2017). Since (5a) 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 Appendix B for details regarding 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" (Christen, 2019). In deterministic blocking, the modeler chooses fields thought to be highly reliable and only compares records that agree on those fields. The record linkage method is applied independently across all blocks, which can be done in parallel for additional speed gains. Of note, blocking on an unreliable field can lead to missed matches, making this form of blocking often undesirable (Steorts et al., 2014).

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; Murray (2016) reviewed this issue in the context of the classical Fellegi-Sunter model, leaving the effect of indexing on Bayesian record linkage models to future work.

With `fabl`, we introduce two techniques to 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 the 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 γ_{ij}^f is discrete, there are only finitely many possible realizations of the comparison vector γ_{ij}. Let P be the number of patterns realized in γ. The quantity P is bounded above by $P^* = \prod_{f=1}^{F}(L_f + 1)$, which clearly depends only on F and L_f. It does not depend (or scale with) 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. Enamorado et al. (2019) accomplished this through a hashing function, which we modify to explicitly handle missing values:

$$h^*(\gamma_{ij}) = \sum_{f=1}^{F} I_{obs}(\gamma_{ij}^f) 2^{\gamma_{ij}^f + I(f>1)\sum_{d=1}^{f-1}(L_d)}. \tag{15}$$

We then map the integers in (15) to sequential integers $\{1, \ldots, P\}$. Denote each unique agreement pattern as h_p, and the set of unique agreement patterns as $\mathcal{P} = \{h_1, \ldots, h_P\}$. When record pair, (i, j), exhibits agreement pattern p, let $\gamma_{ij} = h_p$.

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.

Example 1. *Consider five fields with binary agreements and possible missingness (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. Then (15) 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. Then,* $\gamma_{5,7} = h_{42}$. *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,

$$N_{p_j} = |r_{p_j}| = \sum_{i=1}^{n_A} I(\gamma_{ij} = h_p). \tag{16}$$

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 γ for purposes of writing the likelihood function for `fabl`. To see this, we employ the condensed notation

$$m_p = p(\Gamma_{ij} = h_p | Z_j = i) = \prod_{f=1}^{F} \prod_{l=1}^{L_f} m_{fl}^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)} \tag{17a}$$

$$u_p = p(\Gamma_{ij} = h_p | Z_j \neq i) = \prod_{f=1}^{F} \prod_{l=1}^{L_f} u_{fl}^{I(\gamma_{ij}^f = l) I_{obs}(\gamma_{ij}^f)} \tag{17b}$$

to express the probability that records A_i and B_j form agreement pattern p given that they are a match, in (17a), and not a match, in (17b). Viewed through the perspective of agreement patterns, the likelihood in (6a) can be expressed as

$$\mathcal{L}(\boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{u} \mid \tilde{\gamma}) = \prod_{j=1}^{n_B} \prod_{p=1}^{P} \prod_{i \in r_{p_j}} m_p^{I(Z_j = i)} u_p^{1 - I(Z_j = i)}. \tag{18}$$

The likelihood in (18) allows for more efficient approximate posterior inference of the Gibbs sampler in Section 3, as we describe in the next section.

4.2 Efficient Posterior Inference

In our derivations below, we assume the new update to the Gibbs sampler is $s + 1$ and the current state of the sampler is s. As to not overload notation, we suppress the state of the sampler until presenting the updates to each full conditional distribution based upon our computational proposals using hashing.

Turning first to the full conditional updates for $\boldsymbol{m}$ and $\boldsymbol{u}$, we observe that these parameters only depend on the observed data through quantities that are 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. Conditional on $\boldsymbol{Z}$, we can express the contribution to the likelihood in the full conditional distribution for $\boldsymbol{m}$ and $\boldsymbol{u}$ as

$$\mathcal{L}(\boldsymbol{m}, \boldsymbol{u} \mid \tilde{\gamma}, \boldsymbol{Z}, \pi) = \prod_{p=1}^{P} m_p^{n_p(\boldsymbol{Z})} u_p^{N_p - n_p(\boldsymbol{Z})}. \tag{19}$$

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

$$\boldsymbol{\alpha}(\boldsymbol{Z}^{(s)}) = \boldsymbol{\alpha_0} + \sum_{p=1}^{P} n_p\left(\boldsymbol{Z}^{(s)}\right) \times e(h_p), \tag{20a}$$

$$\boldsymbol{\beta}(\boldsymbol{Z}^{(s)}) = \boldsymbol{\beta_0} + \sum_{p=1}^{P} \left(N_p - n_p \left(\boldsymbol{Z}^{(s)} \right) \right) \times e(h_p). \tag{20b}$$

We can compute the vectorized summations in (20a) and (20b) more efficiently than the previous updates in (9) and (10), which require summing over $n_A n_B$ record pairs for each field and agreement level.

Turning to sampling for π, it remains the same as in (11), so we do not repeat it.

We also use (18) to sample $\boldsymbol{Z}$ efficiently. 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

$$\mathcal{L}(Z_j \mid \tilde{\gamma}, \boldsymbol{m}, \boldsymbol{u}, \pi) = \prod_{p=1}^{P} u_p^{N_{p_j}} \prod_{i \in r_{p_j}} w_p^{I(Z_j = i)}. \tag{21}$$

This likelihood lends itself to markedly faster posterior computations. Sampling Z_j from the full conditional provided in (12) 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. We use two steps in sampling each Z_j.

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

$$p\left(\Gamma_{Z_j^{(s+1)},j} = h_p \mid \tilde{\gamma}, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}\right) \propto \begin{cases} \frac{\pi^{(s+1)} N_{p_j}}{n_A} w_p^{(s+1)}, & h_p \in \mathcal{P}; \\ 1 - \pi^{(s+1)}, & \text{otherwise.} \end{cases} \tag{22}$$

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

$$P\left(Z_j^{(s+1)} = q \mid \Gamma_j = h_p, \boldsymbol{m}^{(s+1)}, \boldsymbol{u}^{(s+1)}, \pi^{(s+1)}\right) = \begin{cases} \frac{1}{N_{p_j}}, & q \in r_{p_j}; \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

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 $O(\frac{F}{C} n_A n_B) + O(\frac{T}{C} n_B P)$.*

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 $O(F n_A n_B)$.

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 $O(\frac{F}{C}n_An_B)$. 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 Fellegi and Sunter (1969) 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 Sadinle (2017) 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 $O(\frac{1}{C}n_BP)$, and so the entire Gibbs sampler has complexity $O(\frac{T}{C}n_BP)$ In summary, the total computational complexity for `fabl` is $O(\frac{F}{C}n_An_B) + O(\frac{T}{C}n_BP)$. □

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 chunks 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 example, if $n_A = n_B = 50000$ and $t_A = t_B = 100$, each A^a and B^b would contain disjoint sets of 500 records. For all a and b, we conduct all-to-all comparisons of the pairs of records in each (A^a, B^b) to construct the comparison matrix γ^{ab}.

We conduct hashing, obtain the compressed comparison matrix $\tilde{\gamma}^{ab}$, and remove the memory-intensive comparison matrix γ^{ab} before continuing with the next chunk of data. In detail, we calculate

$$r_{p_j}^{ab} = \{i \in 1, \ldots, n_A | \gamma_{ij} = h_p, j \in B^b\}, \tag{24a}$$

$$N_{p_j}^{ab} = |r_{p_j}^{ab}|. \tag{24b}$$

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

$$r_{p_j} = (r_{p_j}^{11}, \ldots, r_{p_j}^{t_A t_B}) \text{ for } a = 1, \ldots, t_A \text{ and } b = 1, \ldots, t_B, \tag{25a}$$

$$N_{p_j} = \sum_{a=1}^{t_A} \sum_{b=1}^{t_B} N_{p_j}^{ab}. \tag{25b}$$

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 γ. However, the memory requirements of each are still quadratic in nature. For very large linkage tasks, recording the indices for all record pairs in $\mathcal{R}$ can become computationally burdensome. We 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 (25a), and collect these to form $\text{SEI}(\mathcal{R})$.

Let n_{A^a} and n_{B^b} be the number of records in chunks A^a and B^b respectively. Instead of storing $n_A n_B$ record labels, with SEI we store at most $\sum_{a=1}^{t_A} \sum_{b=b}^{t_A} n_{A^a} n_{B^b} PS$ labels. As shown in the full conditionals in (20a), (20b), (22), and (23), 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 $\boldsymbol{u}$ distribution. For the other $n_B/2$ of these records, there is one match in A, so we simulate 1 comparison vector from the $\boldsymbol{m}$ distribution, and $n_A - 1$ comparison vectors from the $\boldsymbol{u}$ distribution. 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.

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

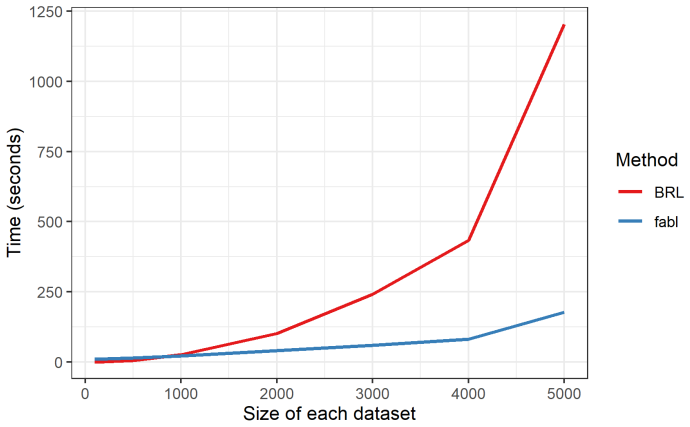

Figure 1: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including the hashing step for `fabl`, for increasing values of both n_A and n_B, as described in Section 5.1. We see near quadratic growth in run-time for `BRL`, and near linear growth for `fabl`.

In Figure 1, 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 2, 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 in parallel for large data. To illustrate that these trends are generalizeable to other specifications of the comparison vectors, we have included the run-time results for an additional simulation study, under different comparison vector settings, in Appendix E.

Importantly, `BRL` implements a Gibbs sampler that is coded C (Sadinle, 2017), 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.

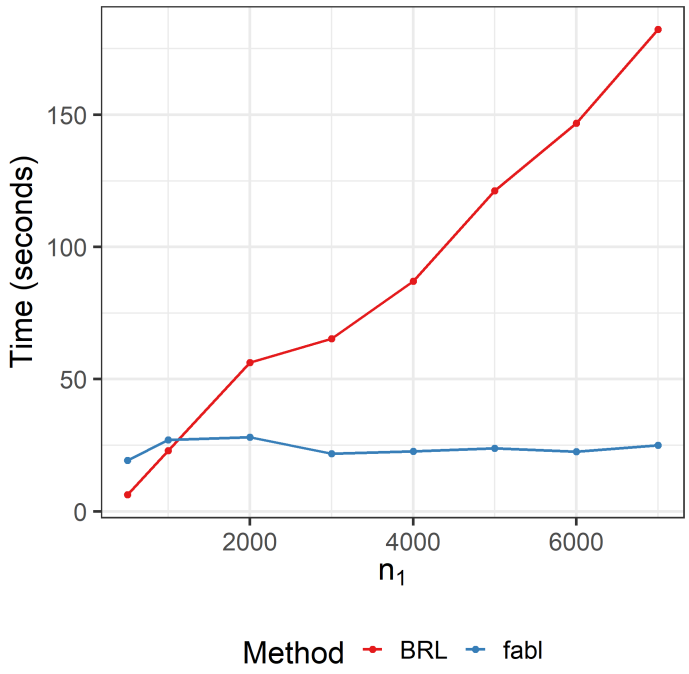

Figure 2: Run-time for `BRL` and `fabl` to run 1000 Gibbs iterations, including hashing step for `fabl`, with n_B fixed at 500, as described in Section 5.1. We see near linear growth in run-time for `BRL`, and near constant run-time for `fabl`.

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 Sadinle (2017). The simulations employ a collection of synthetic data files with varying amounts of error and overlap (the number of records in common across files). Following methods proposed by Christen and Pudjijono (2009) and Christen and Vatsalan (2013), clean records are first simulated from frequency tables for first name, last name, age, and occupation in Australia. Fields are then chosen for distortion uniformly at random. Names are subject to string insertions, deletions and substitutions, as well as common keyboard, phonetic, and optical recognition errors. Age and occupation are distorted through keyboard errors and missingness. These synthetic data files are available in the supplement to Sadinle (2017).

We create comparison vectors according to the default settings of the `compareRecords`

		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.

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 all $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters, with $\alpha_{fl} = \beta_{fl} = 1$ for all f and l. We run the Gibbs sampler for 1000 iterations, and discard the first 100 as burn-in. We calculate Bayes estimates $\hat{\boldsymbol{Z}}$ of the linkage structure using the loss function and post-processing procedure described in Appendix B. Traceplots for parameters of interest for one example simulation are provided in Appendix D; 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 Appendix C.

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

5.3 SEI Sensitivity

Finally, our last simulation demonstrates our method's robustness to different values of S for the SEI memory reduction procedure. We perform record linkage on one set of synthetic datafile described in Section 5.2 with 500 records in each datafile, 250 entities in common across datafiles, and 3 errors present across matching records. To achieve more drastic results, we perform SEI without chunking the data, that is, $t_A = t_B = 1$. In practice, if it is possible to create and store the comparison matrix for all record pairs at one time, there is no need to reduce the memory of the hashed matrix through SEI. For illustration however, it is easier to illustrate the effects of choices of S in this setting.

We preform linkage using SEI with $S = (1, 2, 5, 10, 20)$, and without using SEI, always with 500 iterations of the Gibbs sampler. As any particular SEI implementation may

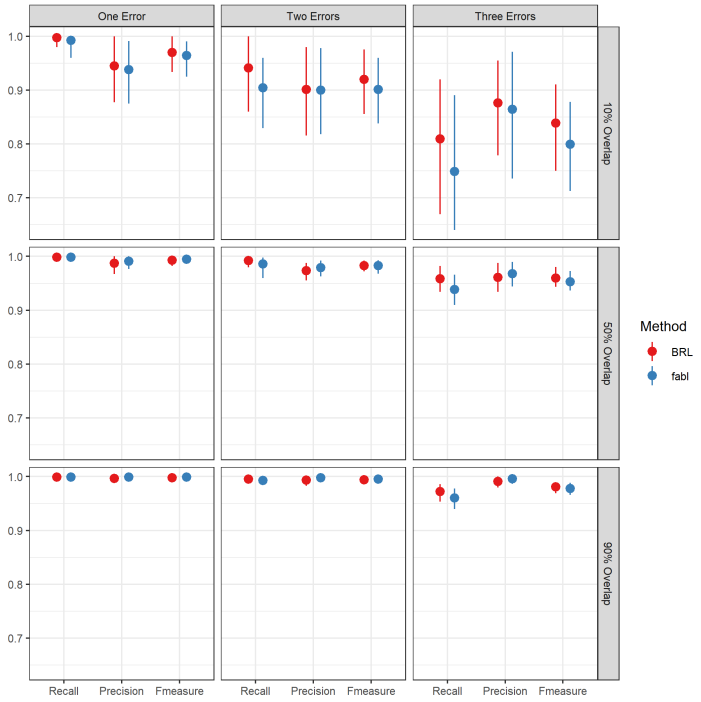

Figure 3: Posterior means and credible intervals for accuracy metrics under the replication of simulation study from Sadinle (2017). For each level of overlap and each level of error, we have 100 paired sets of 500 records. Thus this table summarizes results for 900 data files. We see comparable performance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

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 4, the largest number of distinct linkage estimates occurs when $S = 1$. This makes sense, because 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 5, 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.

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

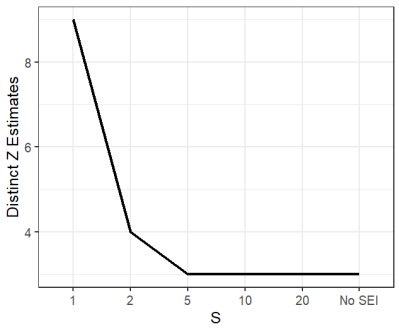

Figure 4: Distinct values of $\hat{Z}$ in Section 5.3 simulation.

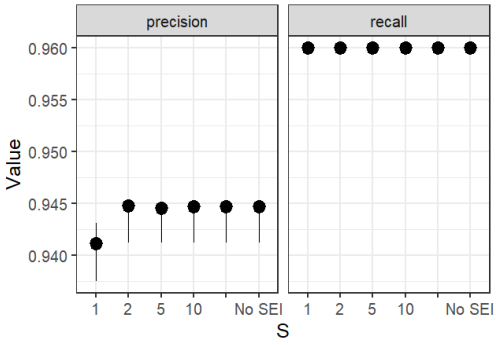

Figure 5: Means and 95% credible intervals for recall and precision in Section 5.3 simulation.

performance of a record linkage algorithm, researchers often examine posterior probabilities. By concentrating probability mass on arbitrary nonmatches, low values of S may induce suspiciously 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 Sadinle (2017) and others (Lum et al., 2013; Jewell et al., 2018). 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 casualties from the war. We utilize lists of casualties from the war, one collected by El Rescate - Tutela Regal (ERTL)(Howland, 2008) and another from the Salvadoran Human Rights Commission (CDHES, by its acronym in Spanish)(Ball, 2000).[1] The ERTL dataset comprises digitized denunciations published throughout the conflict, and the CDHES dataset comprises casualties reported directly to the organization. The ERTL required additional investigation before recording denunciations as human rights abuses, and reports to the CHDES were made shortly

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

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

Table 3: Construction of comparison vectors for El Salvador data resembling original implementation from Sadinle (2017). This setup leads to 2048 possible agreement patterns in total.

after the events occurred; thus, both data files are thought to be fairly reliable. When estimating the total number of casualties, one cannot simply sum the numbers recorded by each organization, as it is likely that the same individuals are recorded in multiple casualty lists. Instead, record linkage techniques must be used to merge data files before analyzing the data (Lum et al., 2013).

There are several challenges with these data. First, both data files have been automatically digitized, which inherently leads to some degree of typographical error. Second, the only fields recorded are given name, last name, date of death, and place of death. While it may be common in other data sets for a parent and child to share the same given name, in this data set, it is more common for cousins to share this information, especially in villages, where there may be only four to five extended families. In Latin America, an individual receives a surname from their father and their mother, leading to a limited number of surnames in small villages.

Following Sadinle (2017), we utilize records that have non-missing entries for given and last name, which results in $n_A = 4420$ records in CHDES 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 CHDES and 80% of records in ERTL, we exclude department from our analysis. Thus, we conduct record linkage using given name, last name, municipality, and day, month, and year of death. We use uniform priors for the $\boldsymbol{m}$ and $\boldsymbol{u}$ parameters.

We initially followed the comparison vector constructions set by Sadinle (2017), using four levels of agreement for each field, according to the thresholds provided in Table 3. This results in $5^5 \times 3 = 6025$ 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 216 possible agreement patterns, 159 are realized in the data. With this revised comparison specification, `fabl` runs in 61 seconds, approximately 4 times faster than the `BRL` run time of 239 seconds. The estimates of the $\boldsymbol{m}$ parameters

		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 216 possible agreement patterns in total.

		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.

under each method are similar, as shown in Figure 7. Estimates of $\boldsymbol{u}$ are indistinguishable, and thus omitted. Traceplots for parameters of interest are provided in Appendix F.

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

Through `fabl`, we arrive at a Bayes estimate of 179 individuals recorded in both data files. We calculate posterior samples of the size of the overlap across files by finding the number of links in each iteration of the Gibbs sampler, and subtracting the number of matches that violate one-to-one matching. The posterior 95% credible interval for the overlap across files is (206, 238), indicating that the Bayes estimate identifies fewer matches than the Gibbs sampler identifies on average. This is because a large number of records in ERTL have multiple plausible matches in CDHES; `fabl` recognizes that a match exists among the several options, but is unable to definitely declare a specific pair as a match in the Bayes estimate. We see similar results under `BRL`, with a Bayes estimate of 181 individuals recorded in both data files, and a posterior 95% credible interval of (211, 244). See Figure 6 for a visual comparison of the Bayes estimates and posterior credible intervals for the two methods. We note that Bayes estimates falling outside of posterior credible intervals has been observed previously in the record linkage literature (Sadinle, 2017; Steorts et al., 2016), and remains a topic for future research.

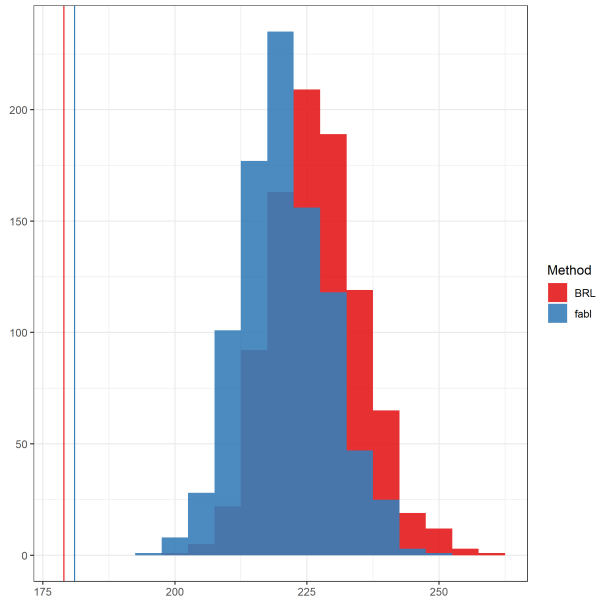

Figure 6: Posterior distribution and Bayes estimate of overlap across the two files. We note they are quite similar under both methods.

6.2 National Long Term Care Study

The National Long Term Care Study (NLTCS) is a longitudinal study tracking the health outcomes of Medicare recipients (Steorts et al., 2016). The initial survey began in 1982, with follow-up surveys taken approximately every five years. As such, patients are surveyed at most once in a given year, and many patients are surveyed across multiple years. In addition, patients can either drop out of the study, pass away, or enter as new patients. Hence, the assumptions of our model hold for this study. We seek to link 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 Guha et al. (2022), we break the longitudinal links and treat the data from 1982 and 1989 as stand-alone data files.

We link records using sex, date of birth, and location using the thresholds shown in Table 5. Storing γ constructed through three comparison scores for each of $20485(17466) \approx 400,000,000$ 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 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

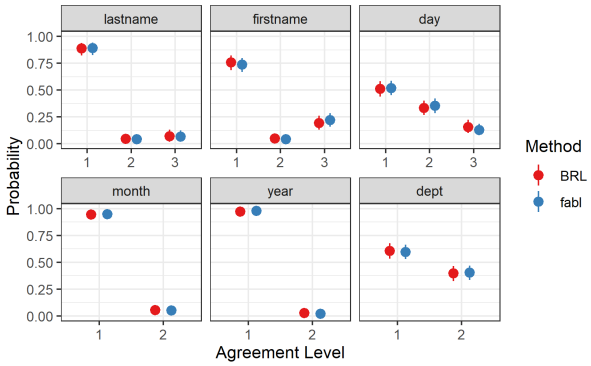

Figure 7: Posterior estimates of $\boldsymbol{m}$ parameters with 95% credible intervals for the El Salvador case study. They are quite similar across the two methods.

40 minutes, which could be reduced considerably through parallel computing.

We run a Gibbs sampler for 1000 iterations, taking about 235 seconds. As shown in Figure 8, the Bayes estimate of the linkage structure consists of 9634 matches, with a 95% credible interval of (9581, 9740). Since we have access to the true linkage structure, we can calculate recall to be 0.89 and precision to be 0.98, resulting in an F-measure of 0.94. Traceplots do not suggest convergence issues, and are similar to those seen in Appendix D and F

7 Conclusion

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

Although the `fabl` method greatly reduces the memory costs for all-to-all comparisons, computing all $n_A n_B$ 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

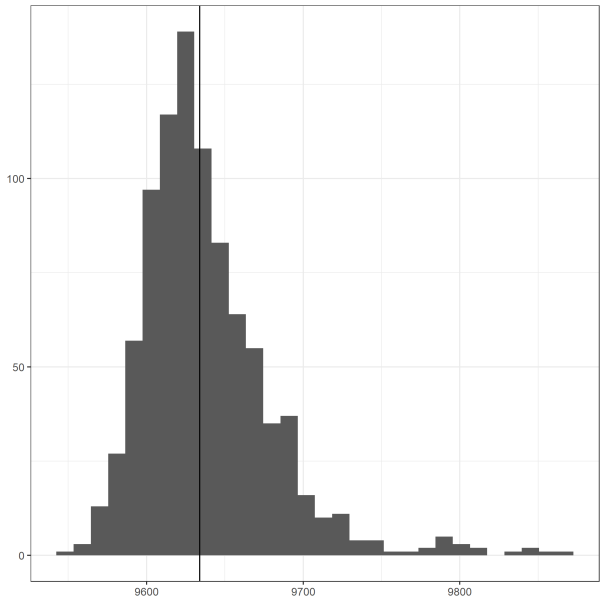

Figure 8: Posterior distribution and Bayes estimate of overlap across years 1982 and 1989 of NLTCS data.

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.

Supplementary Material

Supplementary Material of "Efficient and Scalable Bipartite Matching with Fast Beta Linkage (fabl)." See the .pdf document.

References

Aleshin-Guendel, S. and Sadinle, M. (2023), "Multifile Partitioning for Record Linkage and Duplicate Detection," *Journal of the American Statistical Association*, 0, 1–10. 1, 7

Ball, P. (2000), "The Salvadoran Human Rights Commission: Data Processing, Data Representation, and Generating Analytical Reports," in *Making the Case: Investigating Large Scale Human Rights Violations Using Information Systems and Data Analysis*, eds. P. Ball, H. F. Spirer, and L. Spirer, pp. 15–24, American Association for the Advancement of Science. 17

Betancourt, B., Sosa, J., and Rodríguez, A. (2022), "A prior for record linkage based on allelic partitions," *Computational Statistics & Data Analysis*, 172, 107 – 474. 1, 7

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

Christen, P. (2012), "A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication," *IEEE Transactions on Knowledge and Data Engineering*, 24, 1537–1555. 1, 15

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

Christen, P. and Pudjijono, A. (2009), "Accurate Synthetic Generation of Realistic Personal Information," in *Advances in Knowledge Discovery and Data Mining*, eds. T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, pp. 507–514, Berlin, Heidelberg, Springer Berlin Heidelberg. 14

Christen, P. and Vatsalan, D. (2013), "Flexible and Extensible Generation and Corruption of Personal Data," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, p. 1165–1168, New York, NY, USA, Association for Computing Machinery. 14

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

Dalzell, N. M. and Reiter, J. P. (2018), "Regression Modeling and File Matching Using Possibly Erroneous Matching Variables," *Journal of Computational and Graphical Statistics*, 27, 728–738. 1

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

Enamorado, T., Fifield, B., and Imai, K. (2019), "Using a Probabilistic Model to Assist Merging of Large-Scale Administrative Records," *American Political Science Review*, 113, 353–371. 1, 4, 8

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

Fellegi, I. P. and Sunter, A. B. (1969), "A Theory for Record Linkage," *Journal of the American Statistical Association*, 64, 1183–1210. 1, 2, 3, 4, 5, 11

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

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

Guha, S., Reiter, J., and Mercatanti, A. (2022), "Bayesian Causal Inference with Bipartite Record Linkage," *Bayesian Analysis*, -1. 20

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

Howland, T. (2008), "How El Rescate, a Small Nongovernmental Organization, Contributed to the Transformation of the Human Rights Situation in El Salvador," *Human Rights Quarterly*, 30, 703–757. 17

Jaro, M. A. (1989), "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," *Journal of the American Statistical Association*, 84, 414–420. 2, 4

Jewell, N. P., Spagat, M., and Jewell, B. L. (2018), "Accounting for civilian casualties: From the past to the future," *Social Science History*, 42, 379–410. 17

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

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

Little, R. and Rubin, D. (2002), *Statistical Analysis with Missing Data*, Wiley, Hoboken, New Jersey. 5

Lum, K., Price, M. E., and Banks, D. (2013), "Applications of Multiple Systems Estimation in Human Rights Research," *The American Statistician*, 67, 191–200. 17, 18

Marchant, N. G., Kaplan, A., Elazar, D. N., Rubinstein, B. I. P., and Steorts, R. C. (2021), "d-blink: Distributed End-to-End Bayesian Entity Resolution," *Journal of Computational and Graphical Statistics*, 30, 406–421. 1, 7

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

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

606 Sadinle, M. (2014), "Detecting Duplicates in a Homicide Registry Using a Bayesian
607 Partitioning Approach," *The Annals of Applied Statistics*, 8, 2404–2434. 5, 7

608 Sadinle, M. (2017), "Bayesian Estimation of Bipartite Matchings for Record Linkage,"
609 *Journal of the American Statistical Association*, 112, 600–612. 1, 2, 4, 5, 7, 11, 13, 14,
610 16, 17, 18, 19, 21

611 Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014), "A Comparison
612 of Blocking Methods for Record Linkage," in *Privacy in Statistical Databases*, ed.
613 J. Domingo-Ferrer, pp. 253–268, Cham, Springer International Publishing. 7

614 Steorts, R. C., Hall, R., and Fienberg, S. E. (2016), "A Bayesian Approach to Graphical
615 Record Linkage and Deduplication," *Journal of the American Statistical Association*,
616 111, 1660–1672. 1, 2, 7, 19, 20

617 Tancredi, A., Liseo, B., et al. (2011), "A Hierarchical Bayesian Approach to Record
618 Linkage and Population Size Problems," *The Annals of Applied Statistics*, 5, 1553–1585.
619 1, 7

620 Tang, J., Reiter, J. P., and Steorts, R. C. (2020), "Bayesian Modeling for Simultaneous
621 Regression and Record Linkage," in *Privacy in Statistical Databases*, eds. J. Domingo-
622 Ferrer and K. Muralidhar, pp. 209–223, Cham, Springer International Publishing.
623 1

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

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

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

633 Wortman, J. P. H. (2019), "Record Linkage Methods with Applications to Causal
634 Inference and Election Voting Data," Ph.D. thesis, Duke University. 2, 5, 7

635 Zanella, G., Betancourt, B., Wallach, H., Miller, J., Zaidi, A., and Steorts, R. C. (2016),
636 "Flexible Models for Microclustering with Application to Entity Resolution," in *Proceed-*
637 *ings of the 30th International Conference on Neural Information Processing Systems*,
638 NIPS'16, pp. 1425–1433, NY, USA, Curran Associates Inc. 7