Multifile Partitioning for Record Linkage and Duplicate Detection

Serge Aleshin-Guendel and Mauricio Sadinle
Department of Biostatistics, University of Washington, Seattle, WA

ABSTRACT
Merging datafiles containing information on overlapping sets of entities is a challenging task in the absence of unique identifiers, and is further complicated when some entities are duplicated in the datafiles. Most approaches to this problem have focused on linking two files assumed to be free of duplicates, or on detecting which records in a single file are duplicates. However, it is common in practice to encounter scenarios that fit somewhere in between or beyond these two settings. We propose a Bayesian approach for the general setting of multifile record linkage and duplicate detection. We use a novel partition representation to propose a structured prior for partitions that can incorporate prior information about the data collection processes of the datafiles in a flexible manner, and extend previous models for comparison data to accommodate the multifile setting. We also introduce a family of loss functions to derive Bayes estimates of partitions that allow uncertain portions of the partitions to be left unresolved. The performance of our proposed methodology is explored through extensive simulations. Supplementary materials for this article are available online.

1. Introduction

When information on individuals is collected across multiple datafiles, it is natural to merge these datafiles to harness all available information. This merging requires identifying coreferent records, that is, records that refer to the same entity, which is not trivial in the absence of unique identifiers. This problem arises in many fields, including public health (Hof, Ravelli, and Zwinderman 2017), official statistics (Jaro 1989), political science (Enamorado, Fifield, and Imai 2019), and human rights (Sadinle 2014, 2017; Ball and Price 2019).

Most approaches in this area have thus far focused on one of two settings. Record linkage has traditionally referred to the setting where the goal is to find coreferent records across two datafiles, where the files are assumed to be free of duplicates. Duplicate detection has traditionally referred to the setting where the goal is to find coreferent records within a single file. In practice, however, it is common to encounter problems that fit somewhere in between or beyond these two settings. For example, we could have multiple datafiles that are all assumed to be free of duplicates, or we might have duplicates in some files but not in others. In these general settings, the data collection processes for the different datafiles possibly introduce different patterns of duplication, measurement error, and missingness into the records. Further, dependencies among these data collection processes determine which specific subsets of files contain records of the same entity. We refer to this general setting as multifile record linkage and duplicate detection.

Traditional approaches to record linkage and duplicate detection have mainly followed the seminal work of Fellegi and Sunter (1969), by modeling comparisons of fields between pairs of records in a mixture model framework (Winkler 1994; Jaro 1989; Lar森 and Rubin 2001). These approaches work under, and take advantage of, the intuitive assumption that coreferent records will look similar, and noncoreferent records will look dissimilar. However, these approaches output independent decisions for the coreference status of each pair of records, necessitating the use of ad hoc post-processing steps to reconcile incompatible decisions that ignore the logical constraints of the problem.

Our approach to multifile record linkage and duplicate detection builds on the previous Bayesian approaches where the parameter of interest is defined as a partition of the records. These Bayesian approaches have been carried out in two frameworks. In the direct-modeling framework, one directly models the fields of information contained in the records (Matsakis 2010; Tancredi and Liseo 2011; Liseo and Tancredi 2011; Steorts 2015; Steorts, Hall, and Fienberg 2016; Enamorado and Steorts 2020; Tancredi, Steorts, and Liseo 2020; Marchant et al. 2021), which requires a custom model for each type of field. While this framework can provide a plausible generative model for the records, it can be difficult to develop custom models for complicated fields like strings, so most approaches are limited to modeling categorical data, with some exceptions (Liseo and Tancredi 2011; Steorts 2015). In the comparison-based framework, following the traditional approaches, one models comparisons of fields between pairs of records (Fortini et al. 2001; Larsen 2005; Sadinle 2014, 2017). By modeling comparisons of fields, instead of the fields directly, a generic modeling approach can be taken for any field type, as long as there is a meaningful measure of similarity for that field type.
Sadinle and Fienberg (2013) generalized Fellegi and Sunter (1969) by linking \( K > 2 \) files with no duplicates. However, in addition to inheriting the issues of traditional approaches, their approach does not scale well in the number of files or the file sizes encountered in practice. Steorts, Hall, and Fienberg (2016) presented a Bayesian approach in the direct-modeling framework for the general setting of multifile record linkage and duplicate detection, which has been extended by Steorts (2015) and Marchant et al. (2021). This approach uses a flat prior on arbitrary labels of partitions, which incorporates unintended prior information.

In light of the shortcomings of existing approaches, we propose an extension of Bayesian comparison-based models that explicitly handles the setting of multifile record linkage and duplicate detection. We first present in Section 2, a parameterization of partitions specific to the context of multifile record linkage and duplicate detection. Building on this parameterization, in Section 3, we construct a structured prior for partitions that can incorporate prior information about the data collection processes of the files in a flexible manner. As a by-product, a family of priors for \( K \)-partite matchings is constructed. In Section 4 we construct a likelihood function for comparisons of fields between pairs of records that accommodates possible differences in the datafile collection processes. In Section 5, we present a family of loss functions that we use to derive Bayes estimates of partitions. These loss functions have an abstain option which allow portions of the partition with large amounts of uncertainty to be left unresolved. Finally, we explore the performance of our proposed methodology through simulation studies in Section 6. In the appendix, we present an application of our proposed approach to link three Colombian homicide record systems.

2. Multifile Partitioning

Consider \( K \) files \( X_1, \ldots, X_K \), each containing information on possibly overlapping subsets of a population of entities. The goal of multifile record linkage and duplicate detection is to identify the sets of records in \( X_1, \ldots, X_K \) that are coreferent, as illustrated in Figure 1. Identifying coreferent records across datafiles represents the goal of record linkage, and identifying coreferent records within each file represents the goal of duplicate detection.

We denote the number of records contained in datafile \( X_k \) as \( r_k \), and the total number of records across all files as \( r = \sum_{k=1}^{K} r_k \). We label the records in all datafiles in a consecutive order, that is, those in \( X_1 \) as \( R_1 = (1, \ldots, r_1) \), those in \( X_2 \) as \( R_2 = (r_1 + 1, \ldots, r_1 + r_2) \), and so on, finally labeling the records in \( X_K \) as \( R_K = (\sum_{k=1}^{K-1} r_k + 1, \ldots, r) \). We denote \([r] = (1, \ldots, r)\), where it is clear that \([r] = (R_1, \ldots, R_K)\), which represents all the records coming from all datafiles.

Formally, multifile record linkage and duplicate detection is a partitioning problem. A partition of a set is a collection of disjoint subsets, called clusters, whose union is the original set. In this context, the term coreference partition refers to a partition \( C \) of all the records in \( X_1, \ldots, X_K \), or equivalently a partition \( C \) of \([r]\), such that each cluster \( c \in C \) is exclusively composed of all the records generated by a single entity (Matsakis 2010; Sadinle 2014). This implies that there is a one-to-one correspondence between the clusters in \( C \) and the entities represented in at least one of the datafiles. Estimating \( C \) is the goal of multifile record linkage and duplicate detection.

2.1. Multifile Coreference Partitions

In the setting of multifile record linkage and duplicate detection, the datafiles are the product of \( K \) data collection processes, which possibly introduce different patterns of duplication, measurement error, and missingness. This indicates that records coming from different datafiles should be treated differently. To take this into account, we introduce the concept of a multifile coreference partition by endowing a coreference partition \( C \) with additional structure to preserve the information on where records come from. Each cluster \( c \in C \) can be decomposed as \( c = c_1 \cup \cdots \cup c_k \cup \cdots \cup c_K \), where \( c_k \) is the subset of records in cluster \( c \) that belong to datafile \( X_k \), which leads us to the following definition.

Definition 1. The multifile coreference partition of datafiles \( X_1, \ldots, X_K \) is obtained from the coreference partition \( C \) by expressing each cluster \( c \in C \) as a \( K \)-tuple \((c_1, \ldots, c_K)\), where \( c_k \) represents the records of \( c \) that come from datafile \( X_k \).

For simplicity, we will continue using the notation \( C \) to denote a multifile coreference partition, although technically this new structure is richer and therefore different from a coreference partition that does not preserve the datafile membership.
of the records. The multifile representation of partitions is useful for decoupling the features that are important for within-file duplicate detection or for across-files record linkage.

For duplicate detection, the goal is to identify coreferent records within each datafile. This can be phrased as estimating the within-file coreference partition \( C_k \) of each datafile \( X_k \). Clearly, these \( C_k \) can be obtained from the multifile partition \( C \) by extracting the \( k \)th entry of each cluster \( c = (c_1, \ldots, c_K) \in C \). Two useful summaries of a given within-file partition \( C_k \) are the number of within-file clusters \( n_k = |C_k| \), which is the number of unique entities represented in datafile \( X_k \), and the within-file cluster sizes \( d_k = |\{c_k : c_k \in C_k\}| \), which represent the number of records associated with each entity in datafile \( X_k \).

On the other hand, in record linkage the goal is to identify coreferent records across datafiles. Given the within-file partitions, \( C_1, \ldots, C_K \), the goal can be phrased as identifying which clusters across these partitions represent the same entities. This across-datafiles structure can be formally represented by a \( K \)-partite matching. Given \( K \) sets \( V_1, \ldots, V_K \), a \( K \)-partite matching \( M \) is a collection of subsets from \( \bigcup_{k=1}^K V_k \) such that each \( m \in M \) contains maximum one element from each \( V_k \). If we think of each \( V_k \) as the set of clusters \( C_k \) representing the entities in datafile \( X_k \), then it is clear that the goal is to identify the \( K \)-partite matching \( M \) that puts together the clusters that refer to the same entities across datafiles. This structure can be extracted from a multifile coreference partition \( C \), given that each element \( c = (c_1, \ldots, c_K) \in C \) contains the coreferent clusters across all within-file partitions. Indeed, a multifile coreference partition can be thought of as a \( K \)-partite matching of within-file coreference partitions.

A useful summary of the across-datafile structure is the amount of entity-overlap between datafiles, represented by the number of clusters \( c = (c_1, \ldots, c_K) \in C \) with records in specific subsets of the files. We can concisely summarize the entity-overlap of the datafiles through a contingency table. In particular, consider a \( 2^K \) contingency table with cells indexed by \( h \in \{0, 1\}^K \) and corresponding cell counts \( n_h \). Here, \( h \) represents a pattern of inclusion of an entity in the datafiles, where a 1 indicates inclusion and a 0 exclusion. For instance, if \( K = 3 \), \( n_{011} \) is the number of clusters \( c = (c_1, c_2, c_3) \in C \) representing entities with records in datafiles 2 and 3 but without records in datafile 1. We let \( \mathcal{H} = \{0, 1\}^K \setminus \{0\}^K \) and denote the (incomplete) contingency table of counts as \( n = (n_{h})_{h \in \mathcal{H}} \), which we refer to as the overlap table. We ignore the cell \( \{0\}^K \) which would represent entities that are not recorded in any of the \( K \) files. This cell is not of interest in this article, although it is the parameter of interest in population size estimation (see, e.g., Bird and King 2018).

Example. To illustrate the concept of a multifile partition, consider two files with five and seven records respectively, so that \( X_1 \) contains records 1 – 5 and \( X_2 \) contains records 6 – 12. Suppose the coreference partition is \( \{\{1, 9\}, \{2\}, \{3, 8, 10, 11\}, \{4, 5, 7\}, \{6\}, \{12\}\} \). The corresponding multifile partition is \( C = \{\{1, 9\}, \{2\}, \{3, 8, 10, 11\}, \{4, 5, 7\}, \{6\}, \{12\}\} \). As illustrated in Figure 2, the within-file partitions can be extracted as \( C_1 = \{\{1\}, \{2\}, \{3, 4, 5\}\} \) and \( C_2 = \{\{6\}, \{7\}, \{9\}, \{8, 10, 11\}, \{12\}\} \), and the within-file cluster sizes are \( d_1 = (1, 1, 1, 2) \) and \( d_2 = (1, 1, 1, 3, 1) \). The overlap table in this case is \( (n_{11}, n_{10}, n_{01}) \), indicating that \( n_{11} = 3 \) entities are represented in both datafiles, \( n_{10} = 1 \) entity is represented only in the first datafile, and \( n_{01} = 2 \) entities are represented only in the second datafile. In total, there are \( n_1 = |C_1| = n_{11} + n_{10} = 4 \) unique entities represented in \( X_1 \), \( n_2 = |C_2| = n_{11} + n_{01} = 5 \) unique entities represented in \( X_2 \), and \( n = |C| = n_{11} + n_{10} + n_{01} = 6 \) entities among both datafiles.

### 3. A Structured Prior for Multifile Partitions

Bayesian approaches to multifile record linkage and duplicate detection require prior distributions on multifile coreference partitions. We present a generative process for multifile partitions, building on our representation introduced in Section 2.1. The idea is to generate a multifile partition by first generating summaries that characterize it, as follows:

| Files | Within File Partitions | Multifile Partition | Multifile Partition Summaries |
|-------|------------------------|---------------------|-------------------------------|
| \( X_1 \): 1 2 3 4 5 | \( C_1 \): 1 2 3 4, 5 | \( C \): 1 2 3 4, 5 | # of Clusters: \( n = 6 \) |
| \( X_2 \): 6 7 8 9 | 6 | 6 | Overlap Table: |
| | 2 | 9 | In \( X_1 \): |
| | 3 | 11 | Out \( X_1 \): |
| | 4, 5 | 8, 10, 11 | \( n_{11} = 3 \), \( n_{01} = 2 \) |
| | 11 | 12 | Out \( X_2 \): |
| | 10 | - | \( n_{10} = 1 \) |
| | 12 | | # of Within File Clusters: |
| | | | \( n_1 = 4 \), \( n_2 = 5 \) |
| | | | Within File Cluster Sizes: |
| | | | \( d_1 = (1, 1, 1, 2) \), \( d_2 = (1, 1, 1, 3, 1) \) |

**Figure 2.** An illustration of a multifile partition of \( [12] \), where \( X_1 \) contains records 1 – 5 and \( X_2 \) contains records 6 – 12.
1. Generate the number of unique entities \( n \) represented in the datafiles, which also corresponds to the number of clusters of the multifile partition.

2. Given \( n \), generate an overlap table \( n = \{n_k\}_{k=1}^H \) so that \( n = \sum_{k=1}^H n_k \), where \( H = \{0, 1\}^K \setminus \{0\}^K \). From \( n \) we can derive the number of entities in datafile \( X_k \) as \( n_k = \sum_{h=\emptyset} h_k n_h \), where \( n_h \) is the \( k \)th entry of \( h \).

3. For each \( k = 1, \ldots, K \), given \( n_k \), independently generate a set of counts \( d_k = \{d_{ki}\}_{i=1}^n \), representing the number of records associated with each entity in file \( X_k \). From \( d_k \) we can derive the number of records in file \( X_k \) as \( r'_k = \sum_{i=1}^n d_{ki} \). Index the \( r'_k \) records as \( R'_k = (\sum_{i=1}^{k-1} r'_i + 1, \ldots, \sum_{i=1}^k r'_i) \).

4. For each \( k = 1, \ldots, K \), given \( d_k \), induce a within-file partition \( C_k \) by randomly allocating \( R'_k \) into \( n_k \) clusters of sizes \( d_{k1}, \ldots, d_{kn_k} \).

5. Given the overlap table \( n \) and within-file partitions \( \{C_1, \ldots, C_K\} \), generate a \( K \)-partite matching of the within-file partitions by selecting uniformly at random from the set of all \( K \)-partite matchings with overlap table \( n \). By definition, the result is a multifile coreference partition.

By parameterizing each step of this generative process, we can construct a prior distribution for multifile partitions, as we now show.

### 3.1. Parameterizing the Generative Process

**Prior for the Number of Entities or Clusters.** In the absence of substantial prior information, we follow a simple choice for the prior on the number of clusters, by taking a uniform distribution over the integers less than some upper bound, \( U \), i.e., \( \mathbb{P}(n) = U^{-1}(n \in \{1, \ldots, U\}) \). In practice, we set \( U \) to be the actual number of records across all datafiles \( r \), which is observed. More informative specifications are discussed in **Appendix A**.

**Prior for the Overlap Table.** Conditional on \( n \), we use a Dirichlet-multinomial distribution as our prior on the overlap table \( n = \{n_k\}_{k=1}^H \). Given a collection of positive hyperparameters for each cell of the overlap table, \( \alpha = \{\alpha_k\}_{k=1}^H \), and letting \( \alpha_0 = \sum_{k=\emptyset}^H \alpha_k \), the prior for the overlap table under this choice is \( \mathbb{P}(n|\alpha) = \frac{\Gamma(n + \alpha_0)!}{\prod_{k=\emptyset}^H \Gamma(n_k + \alpha_k)!} \prod_{k=\emptyset}^H \frac{n_k!}{\Gamma(n_k + 1)!} \frac{\alpha_k^{n_k}}{\alpha_0^{n_k + 1}} \). Due to conjugacy, \( \alpha \) can be interpreted as prior cell counts, which can be used to incorporate prior information about the overlap between datafiles. In the absence of substantial prior information, when the number of files is not too large and the overlap table is not expected to be sparse, we recommend setting \( \alpha = (1, \ldots, 1) \). In **Appendix A** we discuss alternative specifications when the overlap table is expected to be sparse.

**Prior for the Within-File Cluster Sizes.** Given the number of entities in datafile \( X_k \), \( n_k = \sum_{h=\emptyset}^H h_k n_h \), we generate the within-file cluster sizes \( d_k = \{d_{ki}\}_{i=1}^n \) assuming that \( d_{k1}, \ldots, d_{kn_k} | n_k \sim \text{Poisson}(\lambda_k) \). Here, \( \lambda_k \) represents the probability mass function of a distribution on the positive integers, so that \( \mathbb{P}(d_k | n_k) = \prod_{i=1}^{n_k} \lambda_k(d_{ki}) \). We do not expect a priori many duplicates per entity, and therefore we expect the counts in \( d_k \) to be mostly ones or to be very small (Miller et al. 2015; Zanella et al. 2016).

We therefore use a similar approach to Klami and Jitta (2016), and use distributions truncated to the range \( \{1, \ldots, U_k\} \), where \( U_k \) is a file-specific upper bound on cluster sizes. We further use distributions where prior mass is concentrated at small values. A default specification is to use a Poisson distribution with mean 1 truncated to \( \{1, \ldots, U_k\} \), that is, \( p_k(d_{ki}) \propto (\lambda_k)^{-1}I(d_{ki} \in \{1, \ldots, U_k\}) \). More informative options could be used for \( p_k(\cdot) \) by using any distribution on \( \{1, \ldots, U_k\} \), where this could vary from file to file if some files were known to have more or less duplication. See **Appendix A** for more discussion.

**Prior for the Within-File Partitions.** Given the within-file cluster sizes \( d_k \), the number of ways of assigning \( d_{k1}, \ldots, d_{kn_k} \) records to clusters \( 1, \ldots, n_k \), respectively, is given by the multinomial coefficient \( r'_k! / \prod_{i=1}^{n_k} d_{ki}! \), with \( r'_k = \sum_{i=1}^{n_k} d_{ki} \). However, the ordering of the clusters is irrelevant for constructing the within-file partition \( C_k \) of \( R'_k \). There are \( n_k! \) ways of ordering the \( n_k \) clusters of \( C_k \), which leads to \( r'_k! / (n_k! \prod_{i=1}^{n_k} d_{ki}!) \) partitions of \( R'_k \) into clusters of sizes \( d_{k1}, \ldots, d_{kn_k} \). We then have \( \mathbb{P}(C_k|d_k) = \frac{( n_k! / r'_k! ) \prod_{i=1}^{n_k} d_{ki}!}{n_k!} \).

**Prior for the K-Partite Matching.** Given the overlap table \( n \) and the within-file partitions \( \{C_1, \ldots, C_K\} \), our prior over \( K \)-partite matchings of the within-file partitions is uniform. Thus, we just need to count the number of \( K \)-partite matchings with overlap table \( n \). This is taken care of by **Proposition 1**, proven in **Appendix A**.

**Proposition 1.** The number of \( K \)-partite matchings that have the same overlap table, \( n = \{n_k\}_{k=1}^H \), is \( \prod_{k=1}^K n_k! / \prod_{k=\emptyset}^H n_k! \). Thus, \( \mathbb{P}(C|\{C_k\}_{k=1}^K, n) = \prod_{k=\emptyset}^H n_k! / \prod_{k=1}^K n_k! \).

The **Structured Prior for Multifile Partitions.** Letting quantities followed by \( (C) \) mean they are computable from \( C \), the density of our structured prior for multifile partitions is

\[
\mathbb{P}(C) = \mathbb{P}(n|\alpha) \prod_{k=1}^K \mathbb{P}(d_k|n_k) \mathbb{P}(C_k|d_k) \mathbb{P}(C|\{C_k\}_{k=1}^K, n)
\]

\[
= \mathbb{P}(n(C)) \frac{n(C)! \Gamma(\alpha_0)}{\Gamma(n(C) + \alpha_0) \prod_{k=\emptyset}^H \Gamma(n_k)} \prod_{k=1}^K \left[ \frac{1}{r'_k(C_k)} \prod_{c_k \in C_k} \frac{\Gamma(n_k c_k + \alpha_k)}{\Gamma(\alpha_k)} \right].
\] (1)

### 3.2. Comments and Related Literature

The structured prior for multifile partitions allows us to incorporate prior information about the total number of clusters, the overlap between files, and the amount of duplication in each file. If we restrict the prior for the within-file cluster sizes to \( p_k(d_{ki}) = I(d_{ki} = 1) \) for a given datafile \( X_k \), then we enforce the assumption that there are no duplicates in that file. Imposing this restriction for all datafiles leads to the special case of a prior for \( K \)-partite matchings, which is of independent interest, as we are not aware of any such constructions outside of the bipartite case (Fortini et al. 2001; Larsen 2005; Sadinle 2017).

Our prior construction, where priors are first placed on interpretable summaries of a partition and then a uniform prior is placed on partitions which have those summaries, mimics the construction of the priors on bipartite matchings of Fortini et al. (2001); Larsen (2005), and Sadinle (2017), and the Kolchin and
allelic partition priors of Zanella et al. (2016) and Betancourt, Sosa, and Rodriguez (2020). While the Kolchin and allelic partition priors could both be used as priors for multifile partitions, these do not incorporate the datfile membership of records. Using these priors in the multifile setting would imply that the sizes of clusters containing records from only one file have the same prior distribution as the sizes of clusters containing records from two files, which should not be true in general.

Miller et al. (2015) and Zanella et al. (2016) proposed the microclustering property as a desirable requirement for partition priors in the context of duplicate detection: denoting the size of the largest cluster in a partition of \( r \) by \( M_r \), a prior satisfies the microclustering property if \( M_r/r \to 0 \) in probability as \( r \to \infty \).

A downside of priors with this property is that they still allow the size of the largest cluster to go to \( \infty \) as \( r \) increases. For this reason, Betancourt, Sosa, and Rodriguez (2020) introduced the stronger bounded microclustering property, which we believe is more practically important: for any \( r \), \( M_r \) is finite with probability 1. Our prior satisfies the bounded microclustering property as \( M_r \leq \sum_{k=1}^{K} U_k \).

While our parameter of interest is a partition \( C \) of \( r \) records, the prior developed in this section is a prior for a partition of a random number of records. In practice, we condition on the file sizes, \( \{r_k\}_{k=1}^{K} \), and use the prior \( \Pr(C \mid \{r_k\}_{k=1}^{K}) \propto \Pr(C)I(r_k(C) = r_k(C) \text{ for all } k) \), which alters the interpretation of the prior. A similar problem occurs for the Kolchin partition priors of Betancourt, Zanella, and Storts (2020), which are similar to Kolchin partition priors, but sequences of clusters priors of Betancourt, Zanella, and Steorts (2020). This motivated the exchangeable microclustering property if

\[
x_k \in \mathbb{R}^t, r_k \in \mathbb{N}, \text{for } k \in \mathbb{N},
\]

reduced the stronger microclustering property if \( r_k \to 0 \) in probability as \( k \to \infty \).

For this reason, Betancourt, Sosa, and Rodríguez (2020) introduced a comparison-based modeling approach to the allelic partition priors of Zanella et al. (2016) and Betancourt, 1790

we now introduce a comparison-based modeling approach to the allelic partition priors of Zanella et al. (2016) and Betancourt, Sosa, and Rodríguez (2020), which are similar to Kolchin partition priors, but sequences of clusters priors of Betancourt, Zanella, and Steorts (2020), which are similar to Kolchin partition priors, but lead to a directly interpretable prior specification. It would be interesting in future work to see if an analogous prior could be developed for our structured prior for multifile partitions. Despite this limitation, we demonstrate in simulations in Section 6 that incorporating strong prior information into our structured prior for multifile partitions can lead to improved frequentist performance over a default specification.

4. A Model for Comparison Data

We now introduce a comparison-based modeling approach to multifile record linkage and duplicate detection, building on the work of Fellegi and Sunter (1969), Jaro (1989), Winkler (1994), Larsen and Rubin (2001), Fortini et al. (2001), Larsen (2005), and Sadinle (2014, 2017). Working under the intuitive assumption that coreferent records will look similar, and noncoreferent records will look dissimilar, these approaches construct statistical models for comparisons computed between each pair of records.

There are two limitations of multifile setting described in Section 2 that are important to consider when constructing a model for the comparison data. First, models for the comparison data should account for the fact that the distribution of the comparisons between record pairs might potentially change across different pairs of files. For example, if files \( X_k \) and \( X_k' \) are not accurate, whereas files \( X_q \) and \( X_q' \) are, then the distribution of comparisons between \( X_k \) and \( X_k' \) will look very different compared with the distribution of comparisons between \( X_q \) and \( X_q' \). Second, the fields available for comparison will vary across pairs of files. For example, files \( X_k \) and \( X_k' \) may have collected information on a field that file \( X_q \) did not. In this scenario, we would like a model that is able to utilize this extra field when linking \( X_k \) and \( X_k' \), even though it is not available in \( X_q \).

In this section we introduce a Bayesian comparison-based model that explicitly handles the multifile setting by constructing a likelihood function that models comparisons of fields between different pairs of files separately. The separate models are able to adapt to the level of noise of each file pair, and the maximal number of fields are able to be compared for each file pair.

4.1. Comparison Data

We construct comparison vectors for pairs of records to provide evidence for whether they correspond to the same entity. For \( k \leq k' \), let \( P_{kk'} = \{(i,j) : i < j, i \in X_k, j \in X_{k'}\} \) denote the set of all record pairs between files \( X_k \) and \( X_{k'} \), and let \( F \) be the total number of different fields available from the \( K \) files. For each file pair \( (k,k') \), \( k \leq k' \), and record pair \( (i,j) \in P_{kk'} \), we compare each field \( f = 1, \ldots , F \) using a similarity measure \( S_f(i,j) \), which will depend on the data type of field \( f \). For unstructured categorical fields such as race, \( S_f \) can be a binary comparison which checks for agreement. For more structured fields containing strings or numbers, \( S_f \) should be able to capture partial agreements. For example, string fields can be compared using a string metric like the Levenshtein edit distance (see, e.g., Bilenko et al. 2003), and numeric fields can be compared using absolute differences. Comparison \( S_f(i,j) \) will be missing if field \( f \) is not recorded in record \( i \) or record \( j \), which includes the case where field \( f \) is not recorded in datafiles \( X_k \) or \( X_{k'} \).

While we could directly model the similarity measures \( S_f(i,j) \), this would require a custom model for each type of comparison, which inherits similar problems to the direct modeling of the fields themselves. Instead, we follow Winkler (1990) and Sadinle (2014, 2017) in dividing the range of \( S_f \) into \( I_f \) intervals \( I_{f0}, I_{f1}, \ldots , I_{fF} \) that represent varying levels of agreement, with \( I_{f0} \) representing the highest level of agreement, and \( I_{fF} \) representing the lowest level of agreement. We then let \( \gamma_f^{ij} = l \) if \( S_f(i,j) \in I_l \), where larger values of \( \gamma_f^{ij} \) represent larger disagreements between records \( i \) and \( j \) in field \( f \). Finally, we form the comparison vector \( \gamma_f = (\gamma_1^{ij}, \ldots , \gamma_F^{ij}) \). Constructing the comparison data this way allows us to build a generic modeling approach. In particular, extending Fortini et al. (2001), Larsen (2005), and Sadinle (2014, 2017), our model for the comparison data is

\[
\gamma_f | C(i) = C(j) \equiv (i,j) \in P_{kk} \quad \text{iid} \quad \text{M}_{kk}(m_{kk}),
\]

\[
\gamma_f | C(i) \neq C(j) \equiv (i,j) \in P_{kk'} \quad \text{iid} \quad \text{U}_{kk'}(u_{kk'}),
\]

where \( C \) is a multifile partition, \( C(i) \) denotes record \( i \)'s cluster in \( C \), \( C(i) = C(j) \) indicates that records \( i \) and \( j \) are coreferent, \( \text{M}_{kk}(m_{kk}) \) is a model for the comparison data among coreferent record pairs from the file pair \( X_k \) and \( X_{k'} \), \( \text{U}_{kk'}(u_{kk'}) \) is a model for the comparison data among noncoreferent record pairs from datafile pair \( X_k \) and \( X_{k'} \), and \( m_{kk'} \) and \( u_{kk'} \) are vectors of parameters.

In the next section we make two further assumptions that simplify the model parameterization. Before doing so, we note a
few limitations of our comparison-based model. First, computing comparison vectors scales quadratically in the number of records. Second, comparison vectors for different record pairs are not actually independent conditional on the partition (see Section 2 of Tancredi and Liseo 2011). Third, modeling discretized comparisons of record fields represents a loss of information. While the first limitation is computational and unavoidable in the absence of blocking (see Appendix B), the other two limitations are inferential. Despite these limitations, we find in Section 6 that the combination of our structured prior for multifile partitions and our comparison-based model can produce linkage estimates with satisfactory frequentist performance.

4.2. Conditional Independence and Missing Data

Under the assumptions that the fields in the comparison vectors are conditionally independent given the multifile partition of the records and that missing comparisons are ignorable (Sadinle 2014, 2017), the likelihood of the observed comparison data, $y_{\text{obs}}$, becomes

$$L(C, \Phi|y_{\text{obs}}) = \prod_{k \leq k'} \prod_{f=1}^{F} \prod_{l=0}^{L_f} \left( m_{kk'}^{(f)} \right) \left( u_{kk'}^{(f)} \right)^{L_f} \left( v_{kk'}^{(f)} \right). \quad (2)$$

Here $m_{kk'}^{(f)} = \mathbb{P}(y_{ij}^f = 1|C(i) = C(j), (i,j) \in P_{kk'})$, $u_{kk'}^{(f)} = \mathbb{P}(y_{ij}^f = 1|C(i) \neq C(j), (i,j) \in P_{kk'})$, $d_{kk'}^{(f)}(C) = \sum_{(i,j) \in P_{kk'}} I_{obs}(y_{ij}^f) = 1 \mathbb{P}(C(i) = C(j))$, $b_{kk'}^{(f)}(C) = \sum_{(i,j) \in P_{kk'}} I_{obs}(y_{ij}^f) = 1 \mathbb{P}(C(i) \neq C(j))$, $I_{obs}()$ is an indicator of whether its argument was observed, and $\Phi = (m, u)$ where $m$ collects all of the $m_{kk'}^{(f)} = (m_{00}^{(f)}, \ldots, m_{kk'}^{(f)})$ and $u$ collects all of the $u_{kk'}^{(f)} = (u_{00}^{(f)}, \ldots, u_{kk'}^{(f)})$. For a given multifile partition $C$, $d_{kk'}^{(f)}(C)$ represents the number of record pairs in $P_{kk'}$ that belong to the same cluster with observed agreement at the level $l$ in field $f$, and $b_{kk'}^{(f)}(C)$ represents the number of record pairs in $P_{kk'}$ that do not belong to the same cluster with observed agreement at the level $l$ in field $f$.

5. Bayesian Estimation of Multifile Partitions

Bayesian estimation of the multifile coreference partition $C$ is based on the posterior distribution $p(C, \Phi|y_{\text{obs}}) \propto p(C)p(\Phi)L(C, \Phi|y_{\text{obs}})$, where $p(C)$ is our structured prior for multifile partitions (1), $L(C, \Phi|y_{\text{obs}})$ is the likelihood from our model for comparison data (2), and $p(\Phi)$ represents a prior distribution for the $\Phi = (m, u)$ model parameters. We now specify this prior $p(\Phi)$, outline a Gibbs sampler to sample from $p(C, \Phi|y_{\text{obs}})$, and present a strategy to obtain point estimates of the multifile partition $C$.

5.1. Priors for $m$ and $u$

We will use independent, conditionally conjugate priors for $m_{kk'}^{(f)}$ and $u_{kk'}^{(f)}$, namely $m_{kk'}^{(f)} \sim \text{Dirichlet}(\mu_{kk'}^{(f)}, \ldots, \mu_{kk'}^{(f)})$ and $u_{kk'}^{(f)} \sim \text{Dirichlet}(\nu_{kk'}^{(f)}, \ldots, \nu_{kk'}^{(f)})$. In this article, we will use a default specification of $(\mu_{kk'}^{(f)}, \ldots, \mu_{kk'}^{(f)}) = (\nu_{kk'}^{(f)}, \ldots, \nu_{kk'}^{(f)}) = (1, \ldots, 1)$. We believe this prior specification is sensible for the $u$ parameters, following the discussion in (Sadinle 2014, sec. 3.2), as comparisons amongst noncorreferent records are likely to be highly variable and it is more likely than not that eliciting meaningful priors for them is too difficult. For the $m$ parameters, it might be desirable in certain applications to introduce more information into the prior. For example, one could set $\mu_{kk'}^{(f)} > \cdots > \mu_{kk'}^{(f)}$ to incorporate the prior belief that higher levels of agreement should have larger prior probability than lower levels of agreement. Another route would be to use the sequential parameterization of the $m$ parameters, and the associated prior recommendations, described in Sadinle (2014).

5.2. Posterior Sampling

In Appendix B, we outline a Gibbs sampler that produces a sequence of samples $(C[i], \Phi[i])_{i=1}^T$ from the posterior distribution $p(C, \Phi|y_{\text{obs}})$, which we will use to obtain Monte Carlo approximations of posterior expectations involved in the derivation of point estimates $\hat{C}$, as presented in the next section. In Appendix B, we discuss the computational complexity of the Gibbs sampler, how computational performance can be improved through the usage of indexing techniques, and the initialization of the Gibbs sampler.

5.3. Point Estimation

In a Bayesian setting, one can obtain a point estimate $\hat{C}$ of the multifile partition using the posterior $\mathbb{P}(C|y_{\text{obs}}) = \int p(C, \Phi|y_{\text{obs}})d\Phi$ and a loss function $L(C, \hat{C})$. The Bayes estimate is the multifile partition $\hat{C}$ that minimizes the expected posterior loss $\mathbb{E}[L(C, \hat{C})|y_{\text{obs}}] = \sum C \cdot L(C, \hat{C})p(C|y_{\text{obs}})$, although in practice such expectations are approximated using posterior samples. Previous examples of loss functions for partitions included Binder’s loss (Binder 1978) and the variation of information (Meilă 2007), both recently surveyed in Wade and Ghahramani (2018). The quadratic and absolute losses presented in Tancredi and Liseo (2011) are special cases of Binder’s loss, and Steorts, Hall, and Fienberg (2016) drew connections between their proposed maximal matching sets and the losses of Tancredi and Liseo (2011).

In many applications, there may be much uncertainty on the linkage decision for some records in the datafiles. For example, in Figure 1 it is unclear which of the records with last name “Smith” are coreferent. It is thus desirable to leave decisions for some records unresolved, so that the records can be hand-checked during a clerical review, which is common in practice (see, e.g., Ball and Price 2019). In the classification literature, leaving some decisions unresolved is done through a reject option (see, e.g. Herbei and Wegkamp 2006), which here we will refer to as an abstain option. We will refer to point estimates with and without an abstain option as partial estimates and full estimates, respectively. We now present a family of loss functions for multifile partitions which incorporate an abstain option, building upon the family of loss functions for bipartite matchings presented in Sadinle (2017).
5.3.1. A Family of Loss Functions with an Abstain Option

For the purpose of this section, we will represent a multifile partition $C$ as a vector $Z = (Z_1, \ldots, Z_r)$ of labels, where $Z_i \in \{1, \ldots, r\}$, such that $Z_i = Z_j$ if $C(i) = C(j)$. We represent a Bayes estimate here as a vector $\hat{Z} = (\hat{Z}_1, \ldots, \hat{Z}_r)$, where $\hat{Z}_i \in \{1, \ldots, r, A\}$, with $A$ representing an abstain option intended for records whose linkage decisions are not clear and need further review. We assign different losses to using the abstain option and to different types of matching errors. We propose to compute the overall loss additively, as $L(Z, \hat{Z}) = \sum_{i=1}^{r} L_i(Z, \hat{Z})$. To introduce the expression for the ith-record-specific loss $L_i(Z, \hat{Z})$, we use the notation $\Delta_y = I(Z_i = Z_j)$, and likewise $\hat{\Delta}_y = I(\hat{Z}_i = \hat{Z}_j)$.

The proposed individual loss for record $i$ is

$$L_i(Z, \hat{Z}) = \begin{cases} 
\lambda_A, & \text{if } \hat{Z}_i = A, \\
0, & \text{if } \Delta_y = \hat{\Delta}_y \text{ for all } j \text{ where } \hat{Z}_j \neq A, \\
\lambda_{\text{FNMI}}, & \text{if } \hat{Z}_i \neq A, \sum_{j \neq i} \Delta_j = 0, \sum_{j \neq i} \hat{\Delta}_j > 0, \\
\lambda_{\text{FM1}}, & \text{if } \hat{Z}_i \neq A, \sum_{j \neq i} \Delta_j > 0, \sum_{j \neq i} \hat{\Delta}_j = 0, \\
\lambda_{\text{FM2}}, & \text{if } \hat{Z}_i \neq A, \sum_{j \neq i} \hat{\Delta}_j > 0, \sum_{j \neq i} (1 - \hat{\Delta}_j) \Delta_j > 0.
\end{cases}$$

That is, $\lambda_A$ represents the loss from abstaining from making a decision; $\lambda_{\text{FNMI}}$ is the loss from a false non-match (FNMI) decision, that is, deciding that record $i$ does not match any other record ($\sum_{j \neq i} \Delta_j = 0$) when in fact it does ($\sum_{j \neq i} \Delta_j > 0$); $\lambda_{\text{FM1}}$ is the loss from a type 1 false match (FM1) decision, that is, deciding that record $i$ matches other records ($\sum_{j \neq i} \hat{\Delta}_j > 0$) when it does not actually match any other record ($\sum_{j \neq i} \Delta_j = 0$); and $\lambda_{\text{FM2}}$ is the loss from a type 2 false match (FM2), that is, a false match decision when record $i$ is matched to other records ($\sum_{j \neq i} \hat{\Delta}_j > 0$ but it does not match all of the records it should be matching ($\sum_{j \neq i} (1 - \hat{\Delta}_j) \Delta_j > 0$).

The posterior expected loss is $E(L(Z, \hat{Z})|p^{\text{obs}}) = \sum_{i=1}^{r} \mathbb{E}[L_i(Z, \hat{Z})|p^{\text{obs}}]$, where

$$E[L_i(Z, \hat{Z})|p^{\text{obs}}] = \begin{cases} 
\lambda_A, & \text{if } \hat{Z}_i = A, \\
\lambda_{\text{FNMI}} \mathbb{P}(\sum_{j \neq i} \Delta_j = 0|p^{\text{obs}}), & \text{if } \hat{Z}_i \neq A, \sum_{j \neq i} \hat{\Delta}_j = 0, \\
\lambda_{\text{FM1}} \mathbb{P}(\sum_{j \neq i} \Delta_j = 0|p^{\text{obs}}) + \lambda_{\text{FM2}} \mathbb{P}(\sum_{j \neq i} (1 - \hat{\Delta}_j) \Delta_j > 0|p^{\text{obs}}), & \text{if } \hat{Z}_i \neq A, \sum_{j \neq i} \hat{\Delta}_j > 0.
\end{cases}$$

and quantities computed with respect to the posterior distribution, $\mathbb{P}(Z|p^{\text{obs}})$, can all be approximated using posterior samples. While this presentation is for general positive losses $\lambda_{\text{FNMI}}, \lambda_{\text{FM1}}, \lambda_{\text{FM2}},$ and $\lambda_A$, these only have to be specified up to a proportionality constant (Sadinle 2017). If we do not want to allow the abstain option, then we can set $\lambda_A = \infty$ and the derived full estimate $\hat{Z}$ will have a linkage decision for all records. Although we have been using partition labelings $Z$, the expressions in (3) and (4) are invariant to different labelings of the same partition. In the two-file case, Sadinle (2017) provided guidance on how to specify the individual losses $\lambda_{\text{FNMI}}, \lambda_{\text{FM1}}, \lambda_{\text{FM2}},$ and $\lambda_A$ in cases where there is a notion of false matches being worse than false non-matches or vice versa. Sadinle (2017) also gave recommendations for default values of these losses that lead to good frequentist performance in terms not over- or under-matching across repeated samples. In Appendix C, we discuss how our proposed loss function differs from the loss function of Sadinle (2017) and propose a strategy for approximating the Bayes estimate.

6. Simulation Studies

To explore the performance of our proposed approach for linking three duplicate-free files, as in the application to the Colombian homicide record systems of Appendix E, we present two simulation studies under varying scenarios of measurement error and datafile overlap. The two studies correspond to scenarios with equal and unequal measurement error across files, respectively. Both studies present results based on full estimates. In Appendix D, we further explore the performance of our proposed approach for linking three files with duplicates, with results based on full and partial estimates.

6.1. General Setup

We start by describing the general characteristics of the simulations. For each of the simulation scenarios we conduct 100 replications, for each of which we generate three files as follows. For each of $n = 500$ entities, $h \in \mathcal{H}$ is drawn from a categorical distribution with probabilities $\{p_h|h \in \mathcal{H}\}$, where $h$ represents the subset of files the entity appears in, and so we change the values of $\{p_h|h \in \mathcal{H}\}$ across simulation scenarios to represent varying amounts of file overlap. Files are then created by generating the implied number of records for each entity. In the additional simulations considered in Appendix D, the generated number of records for each entity depends not only on $h$, but also on the duplication mechanism.

All records are generated using a synthetic data generator developed in Tran, Vatsalan, and Christen (2013), which allows for the incorporation of different forms of measurement error in individual fields, along with dependencies between fields we would expect in applications. The data generator first generates clean records before distorting them to create the observed records. In particular, each observed record will have a fixed number of erroneous fields, where errors selected uniformly at random from a set of field dependent errors displayed in (Sadinle 2014, tab. 3) (reproduced in Appendix D), with a maximum of two errors per field. We generate records with seven fields of information: sex, given name, family name, age, occupation, postal code, and phone number.

For each simulation replicate, we construct comparison vectors as given in (Sadinle 2014, tab. 4) (reproduced in Appendix D). We use the model for comparison data proposed in Section 4 with flat priors on $m$ and $n$ as discussed in Section 5.1, and the structured prior proposed in Section 3 with a uniform prior on the number of clusters and $\alpha = (1, \ldots, 1)$ as described in Section 3.1. Using the Gibbs sampler presented in Appendix B, we obtain 1000 samples from the posterior distribution of multifile partitions, and discard the first 100 as burn-in. In Appendix D, we discuss convergence of the Gibbs sampler, present running times of the proposed approach, and present an extra simulation exploring the running time of the approach with a larger number of records. We then approximate the Bayes estimate $\hat{Z}$ for multifile partitions using the loss function described in Section 5.3.1 as described in Appendix C. For full
estimates, we use the default values of $\lambda_{\text{FN}} = \lambda_{\text{FM}} = 1$ and $\lambda_{\text{FM2}} = 2$ recommended by Sadinle (2017). In Appendix D, we explore alternative specifications of the loss function.

We will assess the performance of the Bayes estimate using precision and recall with respect to the true coreference partition $\mathbf{Z}$. Let $P$ be the set of all record pairs. Using notation from Section 5.3.1, let $TM(\mathbf{Z}, \mathbf{Z}) = \sum_{(i,j) \in P} \Delta_{ij} \Delta_{ij}$ be the number of true matches (record pairs correctly declared coreferent), $FM(\mathbf{Z}, \mathbf{Z}) = \sum_{(i,j) \in P} (1 - \Delta_{ij} \Delta_{ij})$ be the number of false matches (record pairs incorrectly declared coreferent), and $\text{FNM}(\mathbf{Z}, \mathbf{Z}) = \sum_{(i,j) \in P} \Delta_{ij}(1 - \Delta_{ij})$ the number of false non-matches (record pairs incorrectly declared noncoreferent). Then precision is $TM(\mathbf{Z}, \mathbf{Z})/[TM(\mathbf{Z}, \mathbf{Z}) + FM(\mathbf{Z}, \mathbf{Z})]$, the proportion of record pairs declared as coreferent that were truly coreferent, and recall is $TM(\mathbf{Z}, \mathbf{Z})/[TM(\mathbf{Z}, \mathbf{Z}) + \text{FNM}(\mathbf{Z}, \mathbf{Z})]$, the proportion of record pairs that were truly coreferent that were correctly declared as coreferent. Perfect performance corresponds to precision and recall both being 1. In the simulations, we computed the median, 2nd, and 98th percentiles of these measures over the 100 replicate datasets. Additionally, in Appendix D, we assess the performance of the Bayes estimate when estimating the number of entities, $n$.

6.2. Duplicate-Free Files, Equal Errors Across Files

In this simulation study we explore the performance of our methodology by varying the number of erroneous fields per record over $\{1, 2, 3, 5\}$, and also varying $\{p_{k}\}_{k \in \mathcal{H}}$, which determines the amount of overlap, over four scenarios:

- **High Overlap**: $p_{001} = p_{010} = p_{100} = 0.4/3, p_{011} = p_{101} = p_{110} = 0.15, p_{111} = 0.15$, $\alpha = 0.05$, $\lambda_{\text{FM}} = 2$.
- **Medium Overlap**: $p_{001} = p_{010} = p_{100} = 0.7/3, p_{011} = p_{101} = p_{110} = 0.05, p_{111} = 0.15$.
- **Low Overlap**: $p_{001} = p_{010} = p_{100} = 0.8/3, p_{011} = p_{101} = p_{110} = 0.05/3, p_{111} = 0.15$.
- **No-Three-File Overlap**: $p_{001} = p_{010} = p_{100} = 0.55/3, p_{011} = p_{101} = p_{110} = 0.15, p_{111} = 0.0$.

These are intended to represent a range of scenarios that could occur in practice. In the high overlap scenario 60% of the entities are expected to be in more than one datafile, in the low overlap scenario 80% of the entities are expected to be represented in a single datafile, and in the no-three-file overlap scenario no entities are represented in all datafiles.

To implement our methodology, in addition to the general set-up described in Section 6.1, we restrict the prior for the within-file cluster sizes so that they have size one with probability one, incorporating the assumption of no-duplication within files (see Section 3.2). Imposing this restriction for all datafiles leads to a prior for tripartite matchings. To illustrate the impact of using our structured prior, we compare with the results obtained using our model for comparison data with a flat prior on tripartite matchings.

The results of the simulation are seen in Figure 3. We see that our proposed approach performs consistently well across different settings, with the exception of the no-three-file overlap setting under high measurement error, where the precision decreases dramatically. The approach using a flat prior on tripartite matchings has poor precision in comparison, and it is particularly low when the amount of overlap is low. This suggests that our structured prior improves upon the flat prior by protecting against over-matching (declaring noncoreferent record pairs as coreferent). In Appendix D we demonstrate how the performance in the no-three-file overlap setting can be improved through the incorporation of an informative prior for the overlap table through $\alpha$.

6.3. Duplicate-Free Files, Unequal Errors Across Files

In this simulation study, we have different patterns of measurement error across the three files. Rather than each field in each

![Figure 3](image-url). Performance comparison for simulation with equal measurement error across files. Black lines refer to results under our structured prior, gray lines to results under the flat prior, solid lines show medians, and dashed lines show 2nd and 98th percentiles.
Figure 4. Performance comparison for simulation with unequal measurement error across files. “Proposed” refers to our proposed approach, “Single Model” refers to the approach using a single model for all file-pairs and our structured prior for partitions, and “Flat Prior” refers to the approach using our model for comparison data with a flat prior on tripartite matchings. Dots show medians, and bars show 2nd and 98th percentiles.

record having a chance of being erroneous according to Table 3 of Sadinle (2014), we will use the following measurement error mechanism to generate the data. For each record in the first file, age is missing, given name has up to seven errors, and all other fields are error free. For each record in the second file, sex and occupation are missing, last name has up to seven errors, and all other fields are error free. For each record in the third file, phone number and postal code have up to seven errors and all other fields are error free. Under this measurement error mechanism, there is enough information in the error free fields to inform pairwise linkage of the files. We further vary \( \{ p_{ih} \}_{i \in H} \) over the no-three-file and high overlap settings from Section 6.2.

Our goal in this study is to demonstrate that having both the structured prior for partitions and the separate models for comparison data from each file-pair can lead to better performance than not having these components. We will compare our model as described in Section 6.2 to both our model for comparison data with a flat prior on tripartite matchings (as in Section 6.2) and a simplification of our model for comparison data using a single model for all file-pairs but with our structured prior for partitions.

The results of the simulation are given in Figure 4. We see that our proposed approach outperforms both alternative approaches in both precision and recall in both overlap settings. This suggests that both the structured prior for tripartite matchings and the separate models for comparison data from each file-pair can help improve performance over alternative approaches. We note that in the no-three-file (high) overlap setting the precision of the proposed approach is greater than or equal to the precision of the approach using a single model for all file-pairs in 98 (100) of the 100 replications.

7. Discussion and Future Work

The methodology proposed in this article makes three contributions. First, the multifile partition parameterization, specific to the context of multifile record linkage and duplicate detection, allows for the construction of our structured prior for partitions, which provides a flexible mechanism for incorporating prior information about the data collection processes of the files. This prior is applicable to any Bayesian approach which requires a prior on partitions, including direct-modeling approaches such as Steorts, Hall, and Fienberg (2016). We are not aware of any priors for \( K \)-partite matchings when \( K > 2 \), so we hope our construction will lead to more development in this area. The second contribution is an extension of previous comparison-based models that explicitly handles the multifile setting. Allowing separate models for comparison data from each file pair leads to higher quality linkage. The third is a novel loss function for multifile partitions which can be used to derive Bayes estimates with good frequentist properties. Importantly, the loss function allows for linkage decisions to be left unresolved for records with large matching uncertainty. As with our structured prior on partitions, the loss function is applicable to any Bayesian approach which requires point estimates of partitions, including direct-modeling approaches.

There are a number of directions for future work. One direction is the modeling of dependencies between the comparison fields (see, e.g., Larsen and Rubin 2001), which should further improve the quality of the linkage. Another direction is the development of approaches to jointly link records and perform a downstream analysis, thereby propagating the uncertainty from the linkage. See Binette and Steorts (2020, sec. 7.2) for a recent review of such joint models. In this direction, a natural task to consider next is population size estimation, where the linkage of the data files plays a central role (Tancredi and Liseo 2011; Tancredi, Steorts, and Liseo 2020).

Supplementary Material

Appendices: Supplementary appendices A–E. (.pdf file)
R-package multilink: Code implementing the proposed methodology. (.tar.gz file)
Simulation code and data: Code and data used to run simulations. (.zip file)

Acknowledgments

The authors thank Jorge A. Restrepo for providing the Colombian homicide data.

Funding

This research was supported by the NSF under grant SES-1852841 and by the NIH–NIDA under award R21DA051756. The authors thank Jorge A. Restrepo for providing the Colombian homicide data.
References

Ball, P., and Price, M. (2019), "Using Statistics to Assess Lethal Violence in Civil and Inter-State War," Annual Review of Statistics and Its Application, 6, 63–84. [1786,1791]

Betancourt, B., Sosa, J., and Rodriguez, A. (2020), "A Prior for Record Linkage Based on Allelic Partitions," arXiv:2008.10118. [1790]

Betancourt, B., Zanella, G., and Steorts, R. C. (2020), "Random Partition Models for Microclustering Tasks," Journal of the American Statistical Association, pp. 1–13, DOI: 10.1080/01621459.2020.1841647. [1790]

Bilenko, M., Mooney, R. J., Cohen, W. W., Ravikumar, P. and Fienberg, S. E. (2003), "Adaptive Name Matching in Information Integration," IEEE Intelligent Systems, 18, 16–23. [1790]

Binder, D. A. (1978), "Bayesian Cluster Analysis," Biometrika, 65, 31–38. [1791]

Binette, O., and Steorts, R. C. (2020), "(Almost) All of Entity Resolution," arXiv:2008.04443. [1794]

Bird, S. M., and King, R. (2018), "Multiple Systems Estimation (or Capture–Recapture Estimation) to Inform Public Policy," Annual Review of Statistics and Its Application, 5, 95–118. [1788]

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. [1786]

Enamorado, T., and Steorts, R. C. (2020), "Probabilistic Blocking and Distributed Bayesian Entity Resolution," in International Conference on Privacy in Statistical Databases, Cham: Springer, pp. 224–239. [1786]

Fellegrì, I. P., and Sunter, A. B. (1969), "A Theory for Record Linkage," Journal of the American Statistical Association, 64, 1183–1210. [1786,1787,1790]

Fortini, M., Liseo, B., Nuccitelli, A. and Scanu, M. (2001), "On Bayesian Record Linkage," Research in Official Statistics, 4, 185–198. [1786,1787,1790]

Herbei, R., and Wegkamp, M. H. (2006), "Classification With Reject Option," Canadian Journal of Statistics, 34, 709–721. [1791]

Hof, M. H., Ravelli, A. C., and Zwinderman, A. H. (2017), "A Probabilistic Record Linkage Model for Survival Data," Journal of the American Statistical Association, 112, 1504–1515. [1786]

Jarolí, 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. [1786,1789,1790]

Klami, A., and Jitta, A. (2016), "Probabilistic Size-Constrained Microclustering," in Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, Arlington, VA: AUAI Press, pp. 329–338. [1789]

Larsen, M. D. (2005), "Advances in Record Linkage Theory: Hierarchical Bayesian Record Linkage Theory," in Proceedings of the Section on Survey Research Methods, American Statistical Association, pp. 3277–3284. [1786,1787,1790]

Larsen, M. D., and Rubin, D. B. (2001), 'Iterative Automated Record Linkage Using Mixture Models', Journal of the American Statistical Association 96, 32–41. [1786,1790,1794]

Liseo, B. and Tancredi, A. (2011), "Bayesian Estimation of Population Size via Linkage of Multivariate Normal Data Sets," Journal of Official Statistics 27, 491–505. [1786]

Marchant, N. G., Kaplan, A., Elazar, D. N., Rubinstein, B. I. and Steorts, R. C. (2021), "d-Blink: Distributed End-to-End Bayesian Entity Resolution," Journal of Computational and Graphical Statistics, 30, 406–421. [1786,1787]

Matsakis, N. E. (2010), "Active Duplicate Detection with Bayesian Nonparametric Models," PhD thesis, Massachusetts Institute of Technology. [1786,1787]

Meili, M. (2007), "Comparing Clustering as Information Based Distance," Journal of Multivariate Analysis, 98, 873–895. [1791]

Miller, J., Betancourt, B., Zaidi, A., Wallach, H., and Steorts, R. C. (2015), "Microclustering: When the Cluster Sizes Grow Sublinearly With the Size of the Data Set," arXiv:1512.00792. [1789,1790]

Sadine, M. (2014), "Detecting Duplicates in a Homicide Registry Using a Bayesian Partitioning Approach," The Annals of Applied Statistics, 8, 2404–2434. [1786,1787,1790,1791,1792,1794]

——— (2017), "Bayesian Estimation of Bipartite Matchings for Record Linkage," Journal of the American Statistical Association, 112, 600–612. [1786,1787,1790,1791,1792,1793]

Sadine, M., and Fienberg, S. E. (2013), "A Generalized Fellegi–Sunter Framework for Multiple Record Linkage With Application to Homicide Record Systems," Journal of the American Statistical Association, 108, 385–397. [1787]

Steorts, R. C. (2015), "Entity Resolution With Empirically Motivated Priors," Bayesian Analysis, 10, 849–875. [1786,1787]

Steorts, R. C., Hall, R., and Fienberg, S. E. (2016), "A Bayesian Approach to Graphical Record Linkage and Deduplication," Journal of the American Statistical Association, 111, 1660–1672. [1786,1787,1791,1794]

Tancredi, A., and Liseo, B. (2011), "A Hierarchical Bayesian Approach to Record Linkage and Size Population Problems," Annals of Applied Statistics, 5, 1553–1585. [1786,1791,1794]

Tancredi, A., Steorts, R., and Liseo, B. (2020), "A Unified Framework for Deduplication and Population Size Estimation" (with discussion), Bayesian Analysis, 15, 633–682. [1786,1794]

Tran, K.-N., Vatsalan, D., and Christen, P. (2013), "GeoCo: An Online Personal Data Generator and Corruptor," in Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 2473–2476. [1792]

Wade, S., and Ghahramani, Z. (2018), "Bayesian Cluster Analysis: Point Estimation and Credible Balls" (with discussion), Bayesian Analysis, 13, 559–626. [1791]

Winkler, W. E. (1990), "String Comparator Metrics and Enhanced Decision Rules in the Fellegi–Sunter Model of Record Linkage," in Proceedings of the Section on Survey Research Methods, American Statistical Association, pp. 354–359. [1790]

——— (1994), "Advanced Methods for Record Linkage," in Proceedings of the Section on Survey Research Methods, Alexandria, VA: American Statistical Association, pp. 467–472. [1786,1790]

Zanella, G., Betancourt, B., Miller, J. W., Wallach, H., Zaidi, A. and Steorts, R. C. (2016), "Flexible Models for Microclustering With Application to Entity Resolution," in Advances in Neural Information Processing Systems, NY, USA: Curran Associates Inc., pp. 1417–1425. [1789,1790]