Parlr: Frequently Asked Questions

Brian Kundinger

What are parlr's competitors?

Most record linkage techniques are derived from the seminal 1969 paper by Fellegi and Sunter: "A Theory for Record Linkage". The defining characteristic of their model was to transform the sets of records, which constitute text data that is difficult to model, into sets of comparison vectors governed by parameters that can be more easily estimated. Concretely, if files A and B have n_A and n_B records respectively, and if the files share F fields in common upon which to base the linkage, the Fellegi and Sunter approach generates a $n_A n_B \times F$ matrix Γ, which contains similarity scores between each pair of records across datasets. We say γ_{ij} is the comparison vector for record $i \in A$ and record $j \in B$, with γ_{ij}^f providing their similarity score on the f^{th} field. For ease of modeling and computation, we restrict these similarity scores to be discrete, ordinal variables, and the construction of these is left to the modeller. With notation $\gamma_{ij}^f \in \{0, \ldots, L_f - 1\}$ for the number of possible agreement levels for field f, we use binary 0-1 variables to indicate exact matching, and 0-1-2 variables to provide an option for partial matching. For text data, we calculate similarity based on Levenstein distance or some other text similarity score, and bin these scores to integers for use in the model.

Two modern adaptations of the Fellegi Sunter model are important for understanding `parlr` and its contribution. In 2019, Enamorado et al proposed `fastlink`, a method that closely followed the modelling assumptions of Fellegi and Sunter, but used innovating hashing techniques to greatly enhance its speed and scale. Importantly, they maintained Fellegi and Sunter's independent matching assumption, in which the matching status of any (i, j) pair is made independently of the matching of any other (i', j'). This leads to many matchings that violate "one-to-one" considerations that are often implicit in the data, which need to be resolved in a post-processing step. In contrast, Sadinle (2017) proposed "Beta Record Linkage for Bipartite Matching" and the accompanying `BRL` package which strictly enforces one-to-one matching. Specifically, in each iteration of his Gibbs sampler, he considers each record $j \in B$, removes from consideration the records $i \in A$ that have already been matched, then samples a potential link. However, accounting for these dependencies throughout the linkage process is computionally burdensome, leaving `BRL` only suitable for small to moderate linkage problems.

How is parlr different?

While `fastlink` makes $n_A \times n_B$ many indepedent binary linkage decisions and Sadinle makes n_B many *dependent* linkage decisions among n_A many options, `parlr` strikes middle ground by making n_B many *independent* linkage decisions among n_A many options. That is, in each iteration of the Gibbs sampler, we make one linkage decision per $j \in B$, but do not remove previously matched records from consideration. While this does mean

that a particular iteration of the Gibbs sampler may contain some matchings that violate one-to-one requirements, these violations happen infreqently and randomly across all record pairing, so that they do not seriously harm the final Bayes estimate of the linkage structure we make from all the Gibbs samples. In cases where our Bayes estimate does contain matches that violate one-to-one, we note that such cases far less frequently than under `fastlink`, and that these cases are easily resolved by simply taking the match pair with the highest posterior match probability.

Broadly speaking, we increase our computational efficiency by recognizing that record pairs contribute to posterior calculations only through the agreement pattern of the γ_{ij} vector. Let H be the set of unique agreement patterns in the data, let P denote the total number of unique agreement patterns. Note that P is bounded above by $\prod_{f=1}^{F} L_f$, and that this bound does not scale with n_A or n_B. We index these agreement patterns by $p \in \{1, \ldots, P\}$, and say $(i, j) \in h_p$ when the (i, j) pair exhibits the p^{th} agreement pattern. Wherever possible, we conduct calculations over these P agreement patterns rather than the $n_A \times n_B$ record pairs.

By making linkage decisions for each $j \in B$ independently, we do not need to concern ourselves with the precise record label for $j's$ potential pairing within the Gibbs sampler, only the agreement pattern of the γ vector. While the original Fellegi Sunter framework has computational complexity $O(n_A \times n_B)$, these modifications completely remove dependence on n_A throughout the Gibbs sampler, reducing the complexity to $O(n_B)$

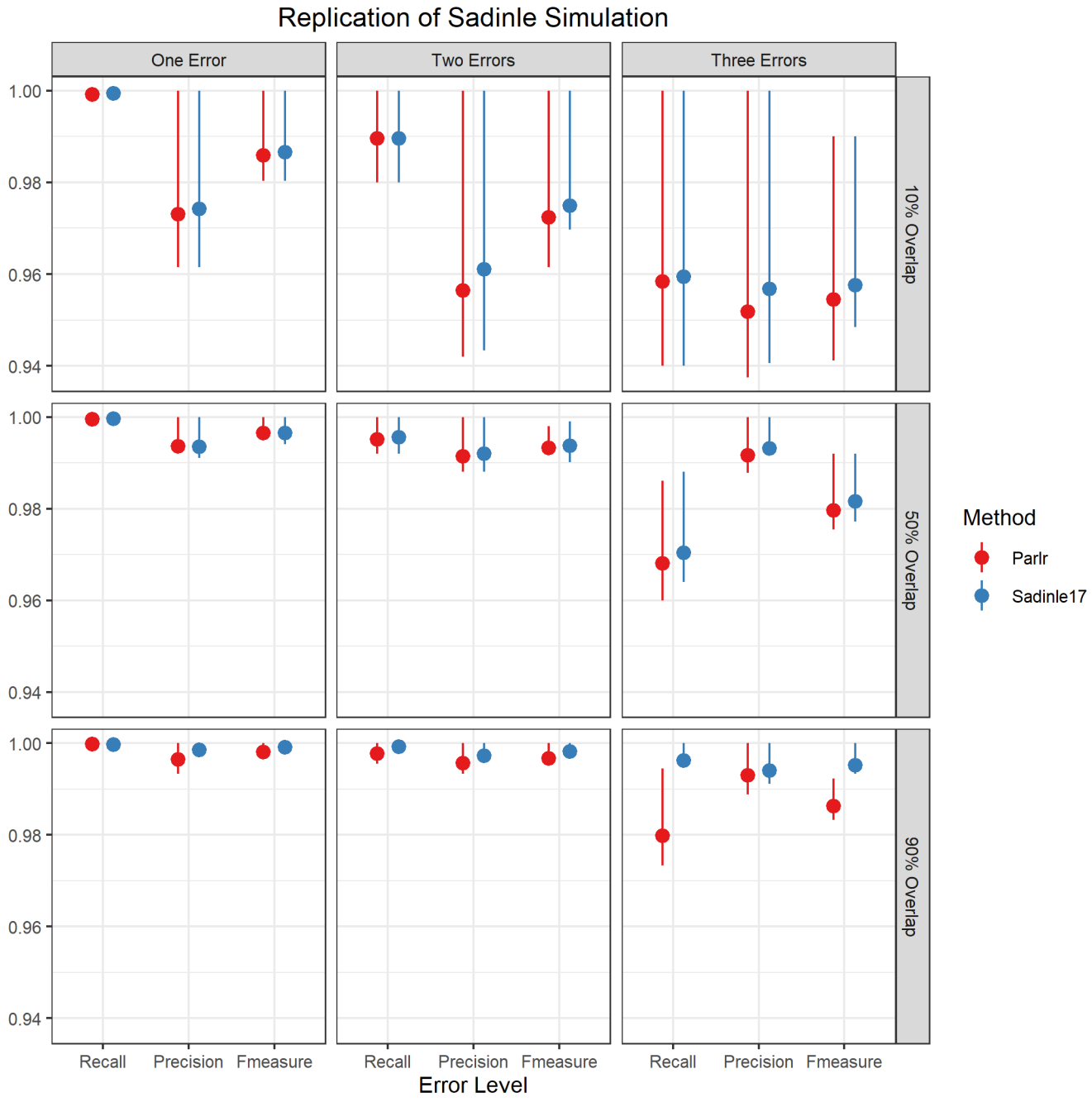
How accurate is it?

In his 2017 paper, Sadinle demonstrated the strength of his method by running `BRL` on simulated pairs of files with differing levels of errors and overlap. Comparing our method against `BRL` on these same simulated datasets, we find that our method only has weakend performance in the extreme scenario of very high errors and very high overlap across files.

$$\text{Recall} = \frac{\text{Matches Correctly Identified}}{\text{True Matches in Data}}$$

$$\text{Precision} = \frac{\text{Declared Matches}}{\text{True Matches in Data}}$$

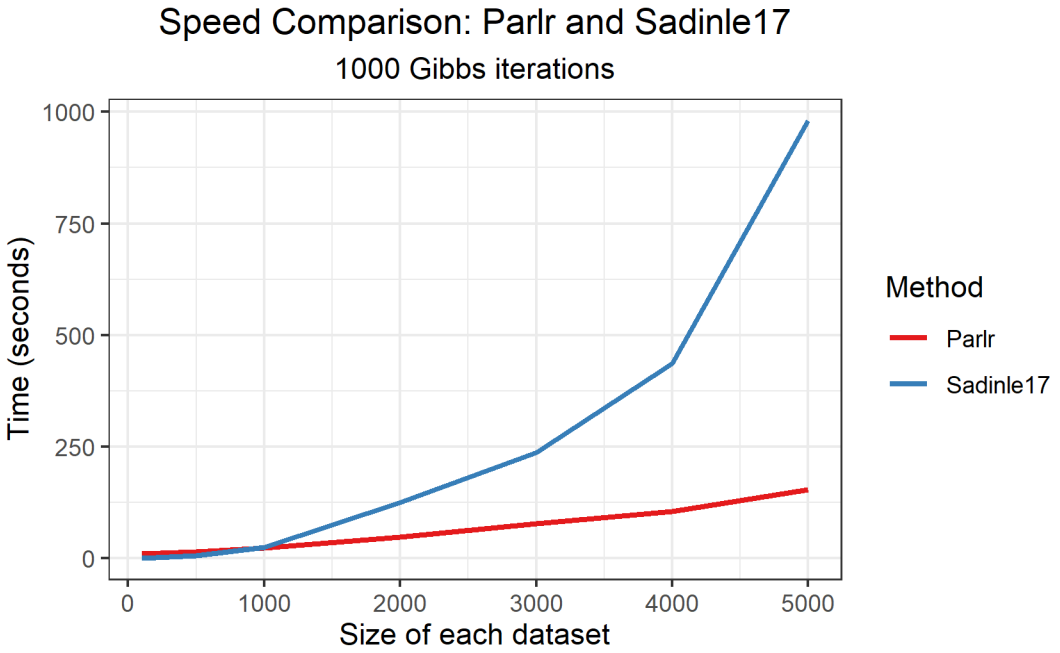
$$\text{F-Measure} = 2\left(\frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}\right)$$



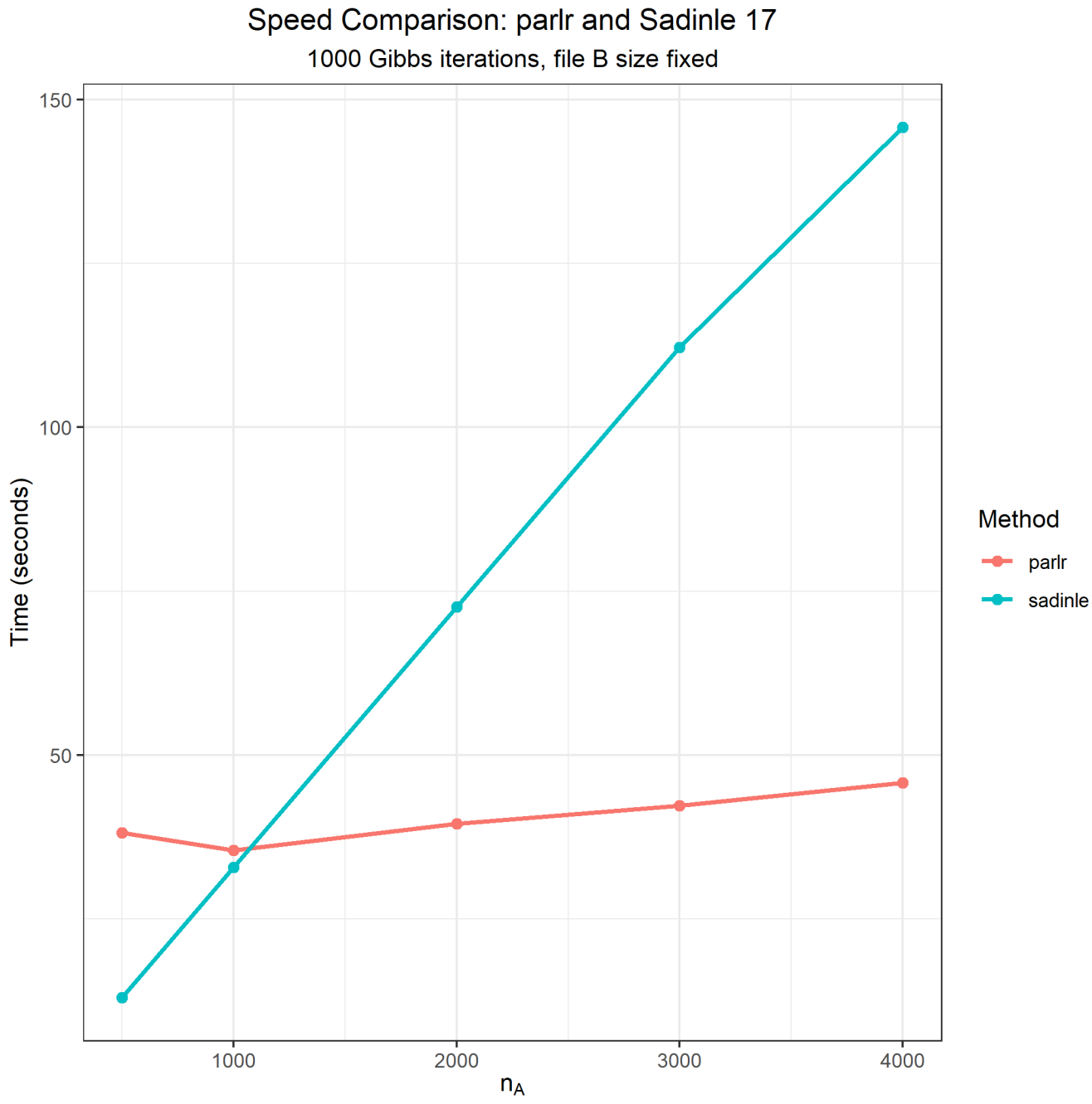
How fast is it?

We simulate Γ matrices of comparison vectors from multinomial distributions meant to emulate the behavior of similarity scores across first name, last name, and day, month, and and year of birth. Distributions provided below. We simulate these data for different values of $n_A and n_B$, and compare the runtime of `parlr` against `BRL`. We note here that `BRL` is coded in C, which makes for unfair comparison against `parlr`, currently only built in R. We see that at low data size, `BRL` outperforms, but that `parlr` is significantly faster at handling larger data. In particular, runtime for `BRL` seems to grow quadratically while

runtime for `parlr` seems to grow linearly. Additionally, although `parlr` is amenable to parallezation, this simulation was run on a single core. Running `parlr` in C or C++ with paralellization for the hashing step and sampling the matching status of the record pairs should lead to even more drastic results.



The above discussion suggests that for fixed n_B, computation time should remain mostly constant with growing n_A. Shockingly, this seems to be true. In the plot below, fixing $n_B = 500$, we see linear growth for the runtime under `BRL` as n_A increases, with much more static runtime under `parlr`. The slight increases in runtime that we do see are due primarily to the hashing step, which again can be run in parallel for large data.



What about deduplication of a single file

At the moment, `parlr` (and its `BRL` competitor) are only able to handle bipartite matching across two files, meaning there is duplication across, but not within, files. Though it seems similar to linkage across files, deduplication within one file is suprisingly much more difficult to model probabalistically. In particular, `parlr` assumes that the linkage decision for record $j \in B$ is independent from all other $j' \in B$, allowing us to propose a prior $\lambda \sim \text{Beta}(\alpha, \beta)$ for the probability of j having some match in A, and update that prior through standard beta-binomial posterior updates. With only one file however, if

the sampler decides that record 1 matches with record 5, there is no way to determine the linkage of record 5 independently; we do not even have n_B independent linkage decisions to work with! Constructing fast, scalable methods for deduplication is one avenue of future work.

Anything else?

Kundinger is particularly interested in linkage scenarios when the reliability of the data (the level of error in the fields), and the rates of matching, differ by subgroups in the data. The simplest case of this occurs when matching historical records for women; due to marriage conventions, the last name is a much less reliable identifier for women than for men. In an upcoming extention, we propose *linkage clusters* that account these difficulties, while still maintaining the computational advances from `parlr`.