Title
Linking Administrative Data: Strategies and Methods

Authors
Augustine, Elsa
Reddy, Vikash
Rothstein, Jesse

Publication Date
2018-12-01

Data Availability
The data associated with this publication are available at:
https://github.com/californiapolicylab/data-linking

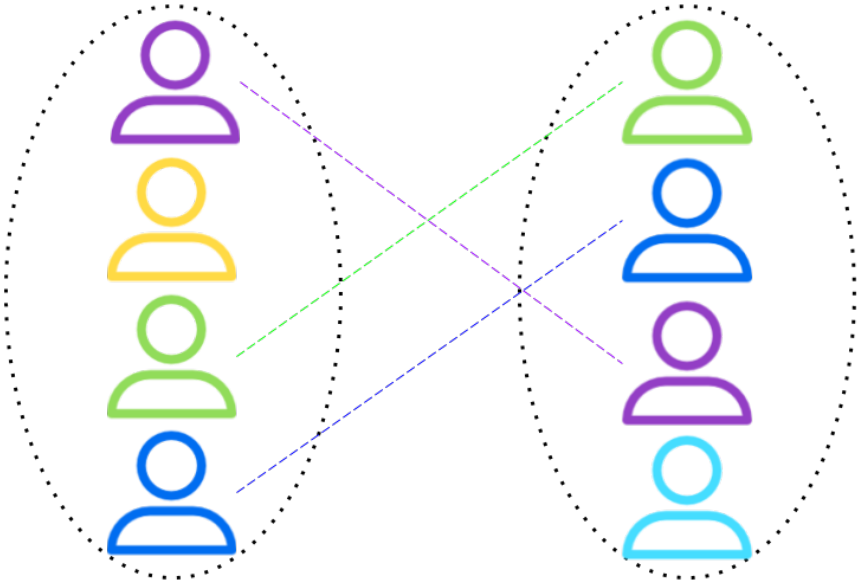
CALIFORNIA POLICY LAB

Linking Administrative Data: Strategies and Methods

A California Policy Lab White Paper

ELSA AUGUSTINE
VIKASH REDDY
JESSE ROTHSTEIN

DECEMBER 2018



The California Policy Lab builds better lives through data-driven policy. We are a project of the University of California, with sites at the Berkeley and Los Angeles campuses.

Executive Summary

We review the linking of datasets that contain identifying information (e.g., names, birthdates) but not unique common identifiers for each individual. We discuss strategies for identifying matches in three families: rules-based matching, supervised machine learning, and unsupervised machine learning. These vary in the ways that they combine human knowledge with computing power. We define different measures of accuracy and explore the performance of common algorithms in test data.

Our goal is to de-mystify data linking for non-technical readers. We attempt to explain the criteria that should inform the choice of linking methods, and the decisions that need to be made to implement them.

Acknowledgements

We thank Evan White, Charles Davis, Nathan Hess, Jared Murray, Sarah Tahamont, Aaron Chalfin, and Zubin Jelveh for helpful conversations and comments on earlier drafts. We are grateful to Caleb Siu for excellent research assistance and to the Laura and John Arnold Foundation for financial support. Cover image created using an icon made by Lucy G from www.flaticon.com.

Contents

Introduction

Let us begin with an example. In cities and counties across California, homelessness is a burgeoning problem. More than 100,000 of the state's residents are homeless, and while it is true that a low supply of affordable housing is one driver of homelessness in the state, over time it has become clear that this crisis is not due to the dearth of low-income housing alone. Homelessness is a complex issue tied to many other inter-related factors, such as mental and physical health, substance use, employment, and interactions with the criminal justice system. To fully grasp the complexity of this issue or begin to address it therefore requires drawing connections and insights from multiple sectors.

Much of the data needed to study the homelessness crisis already exists thanks to the recordkeeping of state and local public agencies in their daily operations. For example, a local police department will know an individual's arrest history, while the state's health agency maintains records of any inpatient psychiatric care received. This information together may begin to explain the root causes of an individual's housing instability and suggest interventions that could help him or her stay housed.

Increasingly, policymakers are recognizing the promise that administrative records hold for unearthing insights about the populations they serve and the impacts of various government programs. A valuable characteristic of administrative data is that it is often possible to link together data from multiple programs, agencies, or databases, allowing researchers to begin to study multi-dimensional problems like the homelessness crisis in California.[1] Research drawing on these linked data is finding relationships across sectors that were previously unappreciated, like the impact of dental care on educational attainment, or the impact of school discipline on criminal justice involvement. However, the initial linking of administrative data is not trivial; in fact it is often quite difficult.

A few decades ago, data linkage had to be done by hand and was enormously labor intensive, often prohibitively so. Later, computers could link datasets much more quickly, but only when they shared a common unique identifier (e.g., a Social Security Number, or SSN). This was great progress, but limits on the practice remained. Often data systems to be linked do not have common unique identifiers; moreover, real-world databases are plagued by missing fields, errors and transpositions, true changes in values (e.g., when an individual moves to a new address), and differences in the way that information is recorded. Until recently, even rudimentary approaches to this problem were so computationally intensive as to be limited to very small databases.

This is no longer the case. Advanced computers, improved database design, and machine learning methods can greatly facilitate improved data matching, even when databases lack common unique identifiers. Data matching remains a mysterious art, however, and is not well understood, even by most users of administrative data.

---

[1] Many terms are used to describe this process of identifying observations that reference the same individual across data systems – a sampling includes "data matching," "data linking" or "linkage," "record linking," "entity resolution," "object identification," "de-duplication," and "field matching" (Christen, 2012). In this paper, we use "record linkage," "data matching," and "data linking" interchangeably to describe the task of matching records across datasets.

This white paper aims to demystify data matching. We discuss the various methods available, at a level intended to be accessible to a non-technical reader. We emphasize those methods that are appropriate for linking across administrative datasets maintained by public agencies, with some personally identifiable information (PII) but no common, unique identifier like an SSN. Our goal is to reveal the advantages and disadvantages of different approaches for this common problem.

In the sections that follow, we illustrate the steps involved in data linking and discuss the ways that computing power can be brought to bear to facilitate this. We then review some of the software packages that implement the various approaches, and present evidence from a particular setting of their accuracy and performance.

Protecting Privacy While Linking Data

When working with administrative data, protecting the privacy of individuals represented in the data is always of utmost importance. Analysis of data that has been linked can often be done without having access to any personally identifiable information (PII); however, the linking process itself relies on such information. As a result, we advise that the linking process be done in settings with strong privacy protections in place and that great care is taken when working with data that has been linked.

There are broadly two alternatives to linking data while maintaining the privacy of individuals. The first, and more common, is to complete the linking process using PII, and then encrypt and separate PII from the remainder of the data before any further analysis is done.

In situations when there are stronger restrictions on PII that prohibit its use in the linking process, such as when the datasets include sensitive health information, one can first encrypt all PII using hashing functions, then link the datasets on the hashed identifiers. This paper does not explore methods for linking hashed data, and we refer interested readers to the literature that has developed in the health field, such as the National Center for Biotechnology Information's resources on the subject (Dusetzina et al., 2014; Kho et al., 2015).

Finally, it is important to note that linking together different administrative datasets increases the probability of reidentification, even when the PII has been removed or otherwise obscured. It is therefore important to ensure that access to linked individual-level data is carefully monitored, even once PII has been removed.

The Challenges of Linking Administrative Data

Fundamentally, linking two datasets amounts to comparing pairs of individual records. A single record from one database is compared to a potential match from another database, and an assessment must be made about whether the two records refer to the same individual. This assessment must then be repeated for each candidate pair of records across the two datasets, often many millions of times.

The simplest case is when there is a common unique identifier field in each dataset, known as a "key," that uniquely identifies records within each dataset and that is common across them. For example, when each dataset contains an SSN or student ID, the assessment of whether a pair of records is a match may be as simple as asking whether those IDs are the same. This can be done rapidly; assuming complete and accurate data, merges of millions of records can be completed quickly on even ordinary computer systems, with perfect accuracy.

The task becomes more difficult when some records are missing the key, when the key field is corrupted in one or both of the datasets, or when there is no single key that identifies records and is available in each dataset. This latter problem is our focus, and it is by far the more common scenario when seeking to link public datasets.

An Illustration

To illustrate, we explore a hypothetical link between two very small datasets, with five records each, listed in **Table 1**. There is no key field that uniquely identifies records in the two datasets; instead, we must rely on other information like names, dates of birth, and gender, none recorded with perfect accuracy, to identify matches.

Table 1: Hypothetical Linking Scenario

Dataset A

ID-A	First Name	Last Name	DOB	Gender
A1	Angelina	Jolie	6-4-1975	
A2	Denzel	Washington	12-28-1954	M
A3	Jhon	Legend	12-28-1978	M
A4	Kerry	Washington		F
A5	Russell	Brand	6-4-1975	M

Dataset B

ID-B	First Name	Last Name	Birth Year	Gender	Zip
B1	Zach	Braff	1975	M	90210
B2	Angela	Jolie	1975	F	
B3	John	Legend	1987	M	90049
B4	Denzel	Washington	1954	M	90210
B5	Kerri	Washington	1977	F	90025

In this simple problem, there are 25 possible combinations of one record from dataset A with one record of dataset B. Each must be evaluated to identify the matches between the two samples. With only 25 potential pairs, it is straightforward to do this by hand. We can quickly conclude that there are four true matches between the datasets, but that record 5 in dataset A (Russell Brand) and record 1 in dataset B (Zach Braff) have no corresponding observations in the other dataset.

This seems simple, but some introspection makes clear that the decision rules needed to arrive at this conclusion are quite complex – how did we conclude that record 3 in dataset A matches record 3 in dataset B, despite differences in both the first name and the year of birth? While it is difficult to articulate the algorithm that we use in doing this, there is no alternative than to do it algorithmically when scaling this approach to millions or billions of potential pairs.

An algorithmic approach can be thought of as consisting of several distinct decisions. We must decide (1) which fields to consider; (2) how we will compare them against one another; and (3) how to use the information we gain through those comparisons to decide which record pairs are matches and which are not.

(1) **Choice of fields** is fairly straightforward. In our example, there are three fields that are common to the two datasets: first name, last name, and gender. One additional field contains similar enough information to allow for comparison – dataset A has the birthdate and dataset B has the birth year. These four fields can be used to assess whether a pair is a match. By contrast, the fifth field in dataset B, the zip code, has no analogue in dataset A, so provides no useful information for linking.[2]

(2) The second decision is **how to compare fields** between the two datasets. To illustrate this decision, we show the 25 potential pairs in **Table 2**, using color coding to indicate the degree of agreement between corresponding fields: blue indicates cases where the two records in a pair have identical values for a particular field, red indicates fields that are clearly different, and yellow shows cases where the values are similar but not identical or where one is missing. The use of an intermediate category allows for the possibility of data entry and spelling errors, which are common in real-world datasets, though it is also possible to use a binary match/non-match categorization. Most of **Table 2** is red, for the simple reason that 21 of the 25 pairs are not in fact true matches.

(3) The third decision is **how to select the few true matches from the many non-matches**. It is clear that the one row that is uniformly red (row 7) can be discarded, and the one row that is uniformly blue (row 9) should be kept. But what to do with the others? The ability to identify partially matching records that are not identical on all fields is essential when linking administrative records. One approach to linking would be to count a pair as a match only if the two databases have identical information in all of the overlapping fields – first names, last names, birth years, and gender. This rule would identify only one pair, row 9 in **Table 2**, as a match. In every other case, there is at least one difference between a field in dataset A and the corresponding field in dataset B.

[2] It is worth mentioning that in real world linking problems there are often fields that are common across the two datasets but not used to link records. For example, if the data are drawn at two different points of time, time varying fields will often be omitted, as they can be an unreliable source of information on the quality of a match. Excluded fields could include those capturing information on occupation, income, or address, among others.

Table 2: Potential Pairs

	Variables from Dataset A					Variables from Dataset B				
Row	ID	First Name	Last Name	DOB	Gender	ID	First Name	Last Name	Birth Yr	Gender
1	A1	Angelina	Jolie	6-4-1975	Missing	B1	Zach	Braff	1975	M
2	A1	Angelina	Jolie	6-4-1975	Missing	B2	Angela	Jolie	1975	F
3	A1	Angelina	Jolie	6-4-1975	Missing	B3	John	Legend	1987	M
4	A1	Angelina	Jolie	6-4-1975	Missing	B4	Denzel	Washington	1954	M
5	A1	Angelina	Jolie	6-4-1975	Missing	B5	Kerri	Washington	1977	F
6	A2	Denzel	Washington	12-28-1954	M	B1	Zach	Braff	1975	M
7	A2	Denzel	Washington	12-28-1954	M	B2	Angela	Jolie	1975	F
8	A2	Denzel	Washington	12-28-1954	M	B3	John	Legend	1987	M
9	A2	Denzel	Washington	12-28-1954	M	B4	Denzel	Washington	1954	M
10	A2	Denzel	Washington	12-28-1954	M	B5	Kerri	Washington	1977	F
11	A3	Jhon	Legend	12-28-1978	M	B1	Zach	Braff	1975	M
12	A3	Jhon	Legend	12-28-1978	M	B2	Angela	Jolie	1975	F
13	A3	Jhon	Legend	12-28-1978	M	B3	John	Legend	1987	M
14	A3	Jhon	Legend	12-28-1978	M	B4	Denzel	Washington	1954	M
15	A3	Jhon	Legend	12-28-1978	M	B5	Kerri	Washington	1977	F
16	A4	Kerry	Washington	Missing	F	B1	Zach	Braff	1975	M
17	A4	Kerry	Washington	Missing	F	B2	Angela	Jolie	1975	F
18	A4	Kerry	Washington	Missing	F	B3	John	Legend	1987	M
19	A4	Kerry	Washington	Missing	F	B4	Denzel	Washington	1954	M
20	A4	Kerry	Washington	Missing	F	B5	Kerri	Washington	1977	F
21	A5	Russell	Brand	6-4-1975	M	B1	Zach	Braff	1975	M
22	A5	Russell	Brand	6-4-1975	M	B2	Angela	Jolie	1975	F
23	A5	Russell	Brand	6-4-1975	M	B3	John	Legend	1987	M
24	A5	Russell	Brand	6-4-1975	M	B4	Denzel	Washington	1954	M
25	A5	Russell	Brand	6-4-1975	M	B5	Kerri	Washington	1977	F

Match | Partial Match | Non-Match

An exact matching rule is clearly too conservative. One would like to allow for differences, if nothing else because there have clearly been data entry errors in preparing these datasets. The first name field illustrates: Record A-3 in dataset A has the first name entered as "Jhon," which seems unlikely to be correct. If we interpret this as a misspelling of "John," it matches with record B-3 in dataset B (row 13 of **Table 2**). These same records have birth years of 1978 in dataset A and 1987 in dataset B. Again, this plausibly derives from a transcription error. It seems quite likely that this pair is a true match, despite the yellow-blue-yellow field scoring.

Identifying matches in these data clearly requires flexibility, with tolerance for missingness and misreporting. This necessitates care: one would not want to consider row 10 or 19 to be a match despite the matching last names, as "Washington" is a common last name and each of the other fields is quite different. In a larger dataset, there might be thousands of potential matches that share the same last name and year of birth, but one would not consider this information sufficient to make a match when other fields disagree. Things are improved somewhat when the last names are

rare – two "Jolie" records are more likely to refer to the same person than two "Washington" records – but even so our confidence would be weak.

Perhaps the hardest cases in our example are pairs 2 and 21. The two records in pair 2 match exactly on last name and year of birth and have similar first names. While one of the records is missing information about gender, there is no indication of an inconsistency here. In pair 21, the two records match exactly on the year and gender, and the last names are quite similar as well. (Using the edit distance, a common measure of distance between two strings, "Brand" and "Braff" are as similar as "Angela" and "Angelina.") In fact, this pair is not a match, but it could easily be counted as one. Ideally, our algorithm would be sensitive enough to count pair 2 as a match but not pair 21, but in a large dataset it will be difficult to specify rules sensitive enough to identify all such borderline cases.

With these examples in mind, we can form a decision rule for the pairs in **Table 2**. Suppose that we consider a pair of observations a match if all of the following are true:

1. The last names are identical.
2. The first names have the same first four letters, perhaps with a single transposition, or one of the first names is missing.
3. The years of birth are the same, or one can be obtained from the other with a simple transposition.
4. Gender either matches or is missing in one dataset.

This set of rules links the two datasets perfectly. It correctly identifies the four pairs that are true matches – rows 2, 9, 13, and 20 in **Table 2** –but does not identify any of the other pairs that are not in fact linked, including the difficult pair in row 21. Thus, this decision rule would be perfectly accurate for this dataset.

There are no guarantees that this perfect performance would remain in a larger linking exercise, however – one might easily identify pairs as matches that are not, while still failing to identify other pairs that are matches as such. In particular, Rule 1 is likely to lead to overlooking a great many true matches. It is common for last names to fail to match perfectly across datasets, whether because of misspellings, name changes upon marriage, or different conventions for entering hyphenated or compound names. In real-world settings, rules that are more forgiving are likely to be desirable. But more relaxed rules may lead to identification of some non-matches as pairs.

Unfortunately, it is not possible to construct an overarching set of linking rules that can be applied to all data matching problems. The rules need to take account of the data fields that are available, the frequency of data entry errors in the component datasets, and the relative costs of over- and under-identifying matches. Even the appropriate weight to put on different information will vary across settings. For example, a link between education records of individuals who graduated during the Great Recession and income tax records, conducted to evaluate the effect of the Recession on earnings, might give less weight to birth-year matches, as most individuals in this cohort will have been born within the same 4 or 5-year window. If linking data pertaining to Korean or Korean-American populations, one might want to put lower priority on last name matches, since it is quite common in these populations for unrelated individuals to share the same last name.

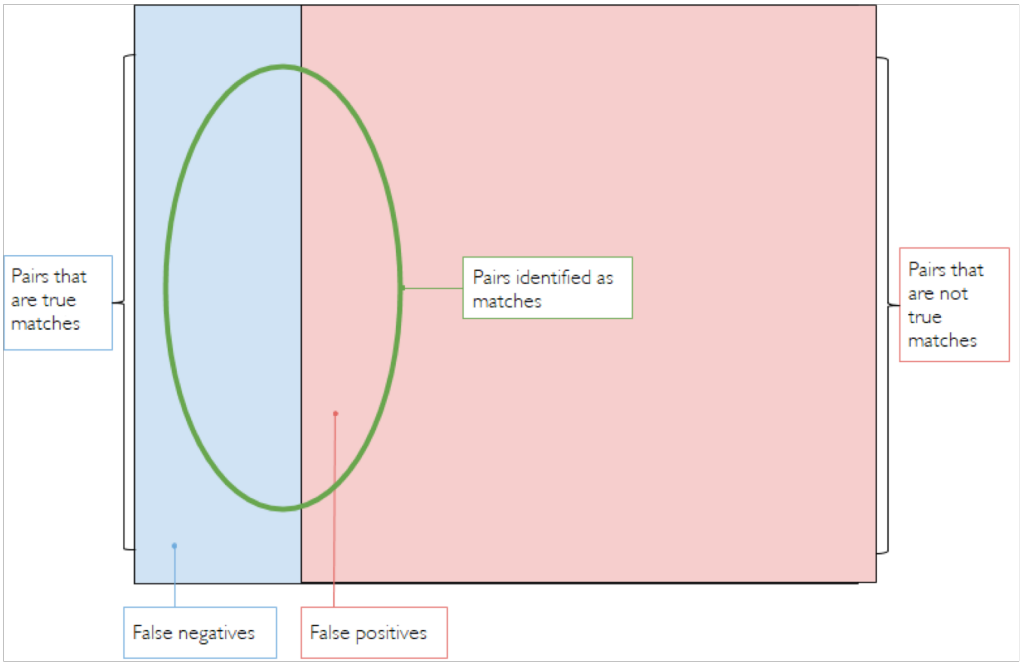
Good linking algorithms require tailoring to the data and extensive calibration, but not all data linking is purely idiosyncratic. This paper highlights a common decision-making framework that is useful across real world data linking problems.

Measuring Accuracy

While we can match the records in the two small datasets in the example above with perfect accuracy, it is unlikely that it would be possible to achieve this in larger samples. It is therefore helpful to distinguish two types of errors that can occur when linking datasets. One is failing to identify a true pair as a match. For example, a rule that is too restrictive might count pair 2 in **Table 2** as a non-match due to the dissimilarity of the first names and a missing gender field. This error is known as a "**false negative**." Another type of error is counting a pair as a match when it is not. A rule that is too permissive might have made this mistake, a "**false positive**," with pair 21.

These two types of errors are illustrated in **Figure 1**. The rectangle represents all possible pairs of records. Here, the blue area represents pairs that are in fact true matches, and the red area represents pairs that are not matches. (This schematic is not to scale. In any matching problem, the blue area will be much, much smaller than the red area – typically a tiny fraction of the full possibility set.) The green oval encloses those pairs that have been identified as matches by a hypothetical matching algorithm. This oval includes more blue pairs than red pairs, but the matched set depicted is not perfect. There are some red pairs inside the green area – these are false positives, pairs incorrectly selected as matches. There are also blue pairs outside the green oval – these are false negatives, or pairs that should have been identified but were overlooked.

Figure 1: Possible Pairs Between 2 Datasets



Both types of errors are undesirable, but it is rarely possible to eliminate either. We can adjust the size of the green oval representing identified matches in **Figure 1** by being more or less stringent in our matching criteria. But this introduces a tradeoff: more stringent criteria for judging a pair to be a match will reduce the number of false positives but increase the number of false negatives, while more lenient criteria will reduce false negatives but drive up false positives.

Whether it is more important to maximize the identification of true matches, thereby avoiding false negatives, or to minimize the identification of false matches, and thereby avoid false positives, is highly dependent on the context of the matching exercise. In some situations, false positives are very costly. An airline matching passengers to a "trusted traveler" list would very much like to avoid giving expedited screening to someone who is not actually on the list – the cost of a false negative is a few minutes of extra screening time for the "trusted traveler" who has been misidentified as a non-trusted traveler, but a false positive might allow a dangerous passenger to slip through.

Conversely, consider a match between a social welfare participant database and child welfare records performed for the purpose of offering additional voluntary parenting assistance services to the welfare program participants who need them. Here, it is more important to identify everyone who might need services, as the cost of over-identification is only to offer services to people who do not need them and are unlikely to take them up. In this case, we might want to minimize false negatives and might be willing to tolerate a higher rate of false positives.

A common way to measure the frequency of the two types of error is via precision and recall rates. **Precision** measures the share of all identified matches that are true matches, and thus the method's ability to avoid false positives. **Recall** measures the share of all true matches that are identified, or the method's ability to avoid false negatives.

Formally, precision is computed as:

$$P = \frac{\text{\# true matches found}}{\text{\#all matches found}} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

Recall is:

$$R = \frac{\text{\# true matches found}}{\text{\#all true matches}} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

As the number of false positives and false negatives move in opposite directions, so too do precision and recall rates. Recall can be increased by relaxing match criteria to properly classify a larger share of true matches (that is, expanding the green oval in **Figure 1**), while this will generally lower precision (by increasing the red area that is captured by the green oval). Conversely, applying more stringent match criteria (shrinking the green oval) will improve the precision but reduce recall (by reducing both the blue and red areas captured by the green oval).

There are only two methods for increasing both precision and recall together (moving the green oval to the left in **Figure 1**). The first is obtaining better data: collecting more information about the

individuals in the two datasets being linked can make it easier to judge whether a pair is a match, thereby increasing both dimensions of accuracy. The second is making better use of the data that are collected. The process we used to arrive at the decision rules for our sample data illustrates this second method. Early alternatives to the set of rules we ultimately selected would have achieved mediocre accuracy. For example, a version of Rule 2 requiring a perfect match on first names would have produced false negatives in pairs 2, 13, and 20, and thus would have reduced recall to 1/4 or 25%, while precision would remain at 100%. However, careful consideration of the data at hand led us to modify this rule and ultimately craft a set of rules that achieved perfect precision and perfect recall.

Unfortunately, in real-world linking problems it is generally not possible to measure precision and recall. Any data that we attempt to link will not come equipped with an answer key to identify the true matches across datasets. When we obtain a set of potential matches, we cannot easily classify them into true and false positives, nor can we isolate the false negatives from among non-matches. While it may be possible to identify false positives, and thus approximate precision, through a careful hand-evaluation of identified matches, it is nearly impossible to accurately estimate the number of matches that an algorithm has failed to identify. Nevertheless, the concepts are useful and provide a framework for considering the accuracy of the different approaches to administrative data linking we describe below.

Another important dimension of accuracy is **representativeness**: whether the likelihood of capturing a true match is similar across different groups in the datasets, so that the set of identified matches is representative of the true matches in the population. This can be hard to achieve – the example discussed above of differences in last name frequencies across ethnic groups suggest that linking rules which rely on last names are likely to achieve lower precision among Korean Americans than among Americans with European-derived last names. These rules may also have lower recall among populations that use Spanish-derived traditions of combining maternal and paternal last names, as such names are often recorded inconsistently in datasets not designed to accommodate them. Like precision and recall, representativeness may not be measurable in real-world linking problems, but it may be an important equity criterion that guides the development of a linking process.

With these important considerations in mind, we turn now to the mechanics of data linking.

Methods of Linking

It is not hard to scan the possible pairs in **Table 2** and distinguish the four real matches from the 21 non-matches, with pairs 2, 13, and 21 representing the only remotely ambiguous cases. The human brain is quite good at this kind of pattern matching task – it is relatively simple to recognize that "Kerri" and "Kerry" are more likely to represent the same person than are "Denzel" and "John." We are also good at identifying a sequence of near misses across multiple records. However, this process is not scalable. We need to examine each pair one at a time, and even someone moving fairly quickly (say, just a couple of seconds per pair) will take a long time to move through a large set of pairs.

Unfortunately, even modest-sized input datasets lead to a very large number of potential pairs that must be considered. Our two sample datasets had only five records each, yielding 25 potential pairs. Two datasets with 100 records each, quite small for any real linking application, would have generated 10,000 potential pairs to consider. In real-world linking exercises, datasets often have tens of thousands of records or more. Clearly, the many millions or billions of resulting potential pairs, of which only a tiny fraction represent true matches, cannot be evaluated by hand; automated methods are required.

These methods typically take a four-step process to identifying matches.

1. **Data Preparation**: Data is cleaned and otherwise prepared, with spurious sources of inconsistencies between the two datasets (e.g., differences in capitalization or abbreviation) resolved.
2. **Blocking**: A preliminary rule (or set of rules) is applied to eliminate a large number of potential pairs that are quite unlikely to represent true matches, and thus dramatically reduce the number of potential pairs that must be considered more carefully. This step is necessary with large datasets, where the number of possible pairs is too large to consider them all carefully.
3. **Field comparisons**: Pairs are compared on a field-by-field basis, and field scores are constructed for each pair of corresponding fields that measure the alignment between them.
4. **Pair comparisons**: The scores from the field comparisons are aggregated across each of the identifying fields (and sometimes across pairs as well) to construct a pair score, and pairs are classified as identified matches or not based on whether this pair score is above or below a threshold.

We discuss each of these steps in turn.

Step 1: Preparing the Data

Two datasets to be linked rarely arrive in forms suitable to be fed directly into automated linking algorithms. An initial data preparation step is generally required to permit meaningful comparisons. **This is not just a formality but requires careful consideration of linking-specific issues.** The goal is to address systematic sources of differences in the way that information is recorded in the two datasets, to make it easier to identify when the corresponding fields in the two input samples represent the same information.

The preparation that is required is entirely specific to the characteristics of the data that are being used. In some cases, it is as simple as converting capital letters to lowercase (or vice versa), so that "John" and "JOHN" are not coded as different in three of their four letters. We might also remove punctuation or abbreviations that may not be included consistently across the two datasets, for example to enable "Boulevard" and "Blvd." (or "Ninth" and "9th") to be recognized as representing the same thing. Similarly, when information is coded differently in the two datasets, we will need to correct this – a computer will need assistance to recognize that month of birth coded as 1 through 12 in one dataset matches to the same information coded as "January" through "December" in the other.

As these examples make clear, there are elements of data cleaning that are common across many linking situations, and there exist a number of resources to assist those engaged in this process, some with example code.[3] However, there are often tricky aspects of the specific datasets being used that require new solutions, and the analyst managing the linking process will typically have to devote a great deal of time to thinking through possible inconsistencies in the datasets. Any spurious inconsistencies that can be identified and fixed here will pay off later in terms of higher recall rates. For example, in one linking exercise that we were part of, initial runs yielded very low match rates. We eventually discovered that names in one dataset had been padded with extra spaces at the end, so that "John " in that dataset did not appear to match with "John" in the other. When the spaces were removed, the recall rate improved dramatically.

This kind of data preparation is tedious and easy to overlook, but it is by far the most important part of a linking exercise. The implications for accuracy of decisions made and time invested in subsequent steps are generally tiny relative to getting the data preparation and cleaning right.

Step 2: Blocking

The number of potential pairs to be considered grows with the product of the number of observations in the two data sets being merged. Imagine a mid-size city wants to study a recently implemented policy designed to combat the school-to-prison pipeline. Measuring the program's effectiveness would require linking data on school discipline records with data from the criminal justice system. A match between 100,000 student records and 10,000 criminal justice RAP sheets, as might arise in such a city, yields one billion potential pairs. This can quickly get overwhelming even for fast computers, particularly if the evaluation of each pair involves a large number of calculations (as is often required to identify similar strings of text).

Importantly, the vast majority of the potential pairs are not true matches. Of the billion pairs in our school-to-prison pipeline example, there can be no more than 10,000 true matches. Thus, the share of all possible pairs that are true matches is under 0.001%. If we can quickly discard many of the 99.999% of pairs that are not matches, we can devote more attention and computing time to the harder cases.

This strategy is called "**blocking**." The idea is to use a preliminary and limited assessment to make an initial cut, discarding the pairs that are clearly not matches. At the most basic level, this involves considering only pairs that match on a particular characteristic. For example, one might block on gender and discard without further consideration any potential pair where gender does not match. This quickly reduces the number of pairs to be considered more carefully by half, but raises concerns about false negatives when gender may be miscoded. Using other characteristics that are more dispersed in the population (e.g., first letter of the last name, or zip code) for blocking cuts the sample of potential pairs much more dramatically – often leaving only a fraction of the initial set of potential pairs.

---

[3] See Chapter 4 of "Linking Data for Health Services Research: A Framework and Instructional Guide" (Dusetzina et al., 2014) for a helpful reference table and example SAS code.

Blocking is not without cost, however. If there are any true pairs that do not match on the blocking variable – perhaps there were transcription errors, or people moved – these will be discarded as well. Thus, it is important to block carefully, only using variables that are likely to be quite stable across the datasets. Blocking is typically performed on relatively crude measures. For example, one might use only the first three digits of the zip code rather than the full five digits to reduce the risk of failing to consider true matches due to transcription errors or short-distance moves. We face a tradeoff – the coarser the blocking criteria, the more pairs that need to be considered; the more specific the criteria, the lower the recall.

The costs of blocking can be mitigated by using multiple overlapping criteria, either simultaneously or in sequence. For example, one can consider all potential pairs that have an exact match on either the first or the last name. This strategy would reduce the number of pairs that need to be considered in our example from **Table 2** from 25 to 6, avoiding the need to more carefully compare Angelina Jolie from dataset A with John Legend in dataset B but still allowing all of the true matches to move forward. One could then more carefully consider possible matches among this subset.[4]

In our simple example, blocking on exact matches on first name would have excluded three of the four matches we arrived at by inspecting them by hand, while the compound rule that uses either first or last name would not have produced any false negatives. As always, however, this block might not be appropriate in other settings. In some datasets, first and last names are sometimes switched; this would lead to failure on both blocking criteria. In fact, we would in general not recommend using names as blocking variables, due to the frequency of misspellings and name changes; we discuss it here merely as an illustration. Gender, year of birth, and large-scale address measures like county or zip code (if people moving between data collection periods is not an issue) are typically more appropriate.

Step 3: Field Comparisons

At the end of the blocking step, we have a long but manageable list of potential pairs to consider, each preprocessed to minimize the number of spurious differences between them. The next step is to score individual field comparisons, as in the color coding in **Table 2**. Here, our scoring may be less crude than the simple criteria we used in blocking. In particular, we can use computationally expensive calculations, as they don't need to be applied to all possible pairs.

In our example above, there are four fields to compare: last name, first name, gender, and birth year. The simplest way to score a field is just to measure whether it matches or not. Here, an exact match between the information in the two component datasets (after preprocessing) yields a score of 1, while a pair that does not match exactly gets a score of 0.

[4] There are more complex and technically advanced ways to block data, which are beyond the scope of this paper. See Human Rights Data Analysis Group's 5-part blog series, for example, for more details (Ball, 2016).

One can clearly improve upon this. The specific methods that are used can be complex, but the ideas are simple and designed to capture intuition about near misses. For example, it can be helpful to give partial credit to cases where the values are non-missing and different, but "close" in some metric that suggests a data entry error. Transpositions are common in data entry, so birth years of 1978 in one dataset and 1798 in the other might be judged closer, and given a higher score, than cases where the differences are less likely to have arisen through data entry errors (say, 1978 and 1954).

Additionally, a failure to match a field because the information is missing from one or the other dataset (e.g., the birth year in pairs 16-20 in **Table 2**) is less indicative of a non-match than is a case where both datasets provide information but it is not aligned. One might want to give a partial score – say, 0.5 – to this field comparison.

Generalizing this, researchers have developed a number of methods for scoring the distance between two strings based on their phonetic pronunciation, the number of character changes it would take to convert one to the other, or the number of characters (or perhaps initial characters) that they have in common. These methods are designed to capture the intuition that "Angela" and "Angelina," for example, are more similar to each other than are "Kerri" and "Denzel," and therefore are more likely to represent different ways of presenting the same underlying information.

At the most basic level, one might give credit to names that match on the first few letters but differ afterward. More elaborately, one common string distance measure would code the distance between "Angela" and "Angelina" as 2 and the distance between "Kerri" and "Denzel" as 5, representing the number of changes that would be needed to convert one to the other.[5] One might then give a score of 1 to any pair with a distance below some specified threshold. A more nuanced alternative to this is to give partial credit based on the string distance, with full scores of 1 reserved for exact matches.

Available data linking software packages often have a large number of built-in comparisons of this sort that can be used to automate the field comparison scoring process. These are often customized for the specific information that the fields contain – one can take advantage of the structure of a birthdate or a street address, for example, to extract more information than is available in a simple string comparison.

Some matching strategies go even further than this, allowing for gradations even among perfect matches. These take advantage of the rarity of the information in the field. Two records with the last name "Jolie," for example, are much more likely to refer to the same person than are two records with the last name "Washington." One can use the distribution of names across observations in the two datasets to make this notion precise, giving a higher score to the "Jolie"

[5] These distances are computed by summing the number of deletions, insertions, and transpositions required to move from one string to another. Transforming "Angela" into "Angelina" requires the insertion of 2 letters and zero deletions or transpositions, so the distance between the two strings is 2. This metric is known as the known as the Levenshtein distance, or edit distance (Gilleland, n.d.).

comparison than to the "Washington" comparison, though typically one would give both comparisons higher scores than imperfect matches (e.g., "Braff"-"Brand").

Step 4: Pair Comparisons

Having scored the individual field comparisons, the next step in linking is to use this information to decide which pairs – of those that survived the blocking step – represent matches, and which do not. In our **Table 2** example, if all fields were compared strictly (requiring an exact comparison to get a score of 1, otherwise getting a score of 0), pair 2, the "Angelina"-"Angela" pair, would have a score list (0, 1, 1, 0) where the first entry represents the score for the first name, the second for last names, the third for birthdates, and the fourth for gender. We need to convert this to a decision: match or not.

The basic approach is simple – high scores across all of the pairwise field comparisons indicate a match, while uniformly low scores indicate a non-match. The implementation can be quite complex, however, using advanced computing algorithms to handle the more difficult cases where some field scores are high and others are low. There are several different approaches available, varying in the extent to which they use artificial intelligence and machine learning methods to substitute for human expertise. We discuss three families of approaches, beginning with the one that relies most on human involvement and moving toward the one that is most hands-off.

Rules-Based Linking

Rules-based linkage methods involve using rules set *ex ante* to decide whether two records are a match, as in our proposed rules for our sample datasets discussed above. That is, the person managing the data linkage process specifies a set of rules that convert the pairwise score lists into decisions. These sets of rules can be simple or quite complex.

A simple, overly strict version of this can illustrate: suppose that we classify only pairs that perfectly match one another on all fields as matches – the score list (1, 1, 1, 1) is counted as a match, and all other score lists are classified as non-matches. This rule is quite rigid and can be expected to produce many false negatives. In our example above, only pair 9 (Denzel Washington) would be identified as a match.

More flexible approaches allow for some non-matches or partial scores. A less strict rule might allow a match so long as all scores are above 0.8 and at least two are exactly 1, or if there are scores of 1 on three of the four fields regardless of the score on the fourth field.

It is often helpful to treat the different field scores asymmetrically – a score of 1, indicating an exact match, is less useful for identifying a true pair when it derives from a gender comparison than from the exact date of birth. One might, for example, require a perfect score for the last name and either the date of birth or first name, with a moderately high score on either the other field or gender.

Although we did not require this in our simple example, matching rules often incorporate information from across pairs, beyond the pure pairwise comparison score lists. In many cases, as in

our example, we are confident that each record in dataset A should match to no more than one record in dataset B and vice versa. This can be used to avoid false positives. Suppose, for example, that a larger version of dataset B contained, in addition to the "Angela Jolie" record, another record for an "Angelina Jolie" who was also born in 1975. The presence of the latter record, which certainly appears to match "Angelina Jolie" in dataset A, reduces our confidence that "Angela Jolie" is a match for that record. A simple pairwise matching rule that uses only pair-level score lists might conclude that both match the "Angelina Jolie" record in dataset A. We know that they cannot both be true matches, however, so our rules might use this information to classify the "Angela Jolie" record as a non-match on the grounds that a stronger candidate match has already been found.

This type of exclusion, which has no effect in our simple example, can be quite important in larger settings. It only functions, however, if we are certain that there are no duplicates in either dataset. If it were the case that "Angela Jolie" and "Angelina Jolie" in dataset B are both records of the same person, with some error in the first name field, any rule restricting each record to one match in the other dataset would yield a false negative for the "Angelina Jolie" - "Angela Jolie" pair, and one would not want to impose it.

A well-crafted set of rules can yield high levels of both precision and recall. Accurate results typically come at a cost, however: a great deal of expert knowledge and time is required to define the rules and check that they are working as intended. The resulting rules are necessarily inflexible and unlikely to be generalizable to other linking problems. Moreover, extensive audits are often needed to understand whether rules inferred from consideration of a small sample of potential pairs are operating reasonably at a larger scale.

Supervised Machine Learning

Fortunately, the task of distilling field-level scores generated in Step 3 into decisions about which pairs to accept as matches turns out to be a task for which machine learning methods are very well suited. These methods can be quite fast to implement, and often generate diagnostic scores that can speed auditing.

There are two broad approaches to record linking with machine learning. The first is known as "supervised" machine learning. Supervision refers to the role of the human operator: in supervised machine learning, the operator classifies a modest number of pairs as matches or not, using his or her judgment. This is known as "training" the algorithm. The list of classified pairs is then turned over to a machine learning algorithm. The algorithm infers a set of rules that can best reproduce the decisions made by the human trainer. Once those rules have been formed, they are applied to additional data to classify pairs as matches or not.

This strategy clearly requires initial human input, which can be time consuming (though less so than it would be to develop a rules-based approach without machine learning). The more training that is done, the better the rules that the algorithm will create. However, a well-crafted algorithm will select pairs for scoring that maximize the information that can be extracted from a small amount of training. It is often possible to achieve a surprising degree of accuracy from training samples of only 50 or 100 pairs.

Unsupervised Machine Learning

A second approach, known as "unsupervised" machine learning, requires no training at all (though human input is still required in constructing the field comparison scoring rules in Step 3). The algorithm is able to infer a rule solely from the distribution of field scores across possible pairs.

This seems magical, but it is actually fairly straightforward. The basic idea is that each pair is either a match or non-match. If it is a match, we expect that the linking fields will generally be closely aligned – there may be missing values and transcription errors, but field match scores will tend to be fairly high across all fields. If it is not a match, the opposite is the case: it is quite unlikely that any single non-matching pair will have a score above zero on any of the matching fields, and more unlikely still that it will have high scores on several of those fields. Therefore, if a pair has high scores on several fields the probability that it is a match is quite high, while a pair with zero scores on most or all fields has a very low chance of being a match.

Implementation of this idea requires a fair amount of computing power, but it can yield surprisingly accurate matches, particularly when the datasets are clean and there are more than two or three fields to use in forming the score lists. As the number of transcription errors or other sources of low pairwise field comparison scores among true matches rises, this method's performance can deteriorate, both in terms of computing difficulty and accuracy.

A key assumption of unsupervised machine learning algorithms is that field-level mismatches are random and independent of one another – that if a pair is a true match, the presence of a transcription error or other mismatch in one field does not affect the probability of an error in another field. As a result, true pairs with one mismatched field are much more common than true pairs with two mismatched fields, which are themselves much more common than true pairs with three mismatched fields (Enamorado, Fifield, and Imai, 2017). However, it is easy to think of real-world situations that would violate this assumption. For example, it is not uncommon to see records in administrative datasets in which the first and last name have been switched, or a middle and last name are separated in one dataset and condensed into a double last name in another. These records will match poorly to their true pairs in other datasets on both fields, a violation of the independence assumption that unsupervised machine learning algorithms rely on. One would expect that unsupervised algorithms will perform poorly in the presence of this type of error, though to our knowledge there has not been a formal exploration of this supposition.

While machine learning approaches generally require less human input and set-up time than rules-based approaches, a drawback relative to a rules-based approach is that the decisions machine learning algorithms make are often opaque. Rules developed by a machine learning algorithm often involve complex combinations of individual field scores, and it may be hard to explain why an algorithm concluded that one pair is a match and another is not. However, when implemented correctly, machine learning algorithms can yield highly accurate results in terms of both precision and recall.

Evaluating Matching Algorithms

Each approach to linking data has strengths and weaknesses, as do the various software options for implementation. One wants a method that is accurate, easy to implement and customize, that runs quickly on available systems with the data that must be linked, and that requires minimal operator time to learn, apply, or train. We discuss some of the key criteria below, followed by profiles of some of the most common software packages available for data linking. Which of these criteria are most important, and which software is most fitting, depends on the context.

Accuracy

Accuracy is arguably the most important metric upon which any method or package should be judged. As discussed earlier, accuracy is commonly measured by precision and recall, and by the representativeness of identified matches. Within any method, there can be tradeoffs between these criteria, with higher precision often achievable at the cost of lower recall (and vice versa), though there may not be similar tradeoffs between packages.

For any algorithm and package, accuracy is likely to vary across specific linking problems. When there are a large number of available identifying fields, and when these are recorded with high fidelity and few changes across the original datasets, any algorithm is likely to yield very high accuracy. Messier data will generally lead to lower accuracy.

Importantly, the decline in accuracy due to mistakes in the data may vary across algorithms. For example, as discussed above, errors across several fields (e.g., switching the first and last name) present a particular problem for unsupervised machine learning algorithms, while rules-based and supervised machine learning algorithms should have no trouble with this type of error, provided that the operator defines rules in Steps 2 and 3 to capture it. The pairwise comparison score list created in Step 3could, for example, include a score for the similarity between the first name in one input dataset and the last name in the other. This score will be high in cases where the two names have been reversed.

Reproducibility

A criterion that relates to accuracy is reproducibility. Some methods – for example, human-written rules-based linking – are reproducible by design: the same rules will identify the same pairs as matches every time they are run. Machine learning-based algorithms may not have that feature. In supervised machine learning algorithms, for example, there is typically some element of randomness in the selection of pairs to be presented to the human operator in the training step. Even with the same training sample, different human operators may make different decisions about how to classify ambiguous pairs. Either may lead to differences in the inferred decision rule, and thus in the set of pairs identified as matches.

Whether this variability represents a problem will, as with other criteria, differ across settings. In at least some settings – for example, where results of the match will lead to decisions that are consequential for the individuals involved, and where those individuals have due process rights – a

high degree of variability could be a very large problem, as it will not necessarily be evident which results should be relied upon when output differs across runs.

Efficiency and usability

Another important aspect of a linking method's performance is its efficiency. Linking of large datasets can take days, weeks, or even longer, especially if the method is inefficient. All things equal, one would prefer a method that economizes on the computer resources required, on the amount of time required to configure the software and implement the match, and on the amount of specific expertise needed. It often makes sense to trade these criteria off against accuracy, accepting a few additional mistakes in order to get a match running quickly at limited expense.

We consider several sub-components of efficiency.

Accessibility of software

The practicality of implementing any particular approach depends at least in part on the availability of relevant software and of operators who are able to use it. Relevant considerations include:

- Is software available to implement the approach?
- Is it costly?
- Is there transparent code available that is easy to interpret and adapt?
- Is the software easy to use, or does it require specialized knowledge?

Our discussion of particular packages below includes information about the accessibility of each.

Customizability

As we discuss above, a successful linking exercise will often require careful tuning of the linking algorithm – adjusting the measures of similarity at the field level, determining the relative importance of false positives and false negatives, and so on. Thus, it is important that a software package be customizable to deal with the particularities of the linking problem to which it is being applied. Moreover, software packages vary in how easy they are to customize – some are designed with user-friendly interfaces, while others can be customized but only by a skilled programmer who spends time learning the details of the package's code.

Supervised machine learning algorithms introduce another aspect of customizability and accessibility. Here, users must train the model on a sample of paired observations. Packages vary in the way that this training is handled. Some offer graphical user interfaces that offer up pairs for the trainer to score, while others are more rudimentary. A feature that can be useful is the ability to feed a pre-labeled dataset of scored pairs to the program in lieu of training observations one by one. However, if the package chooses training pairs that are hard to classify and therefore helpful to developing an accurate algorithm, feeding in pre-labeled data may sacrifice some degree of accuracy.

A final important aspect of customizability relates to what is being matched. Our discussion has focused on linking problems that involve two datasets, each unduplicated, that need to be linked to each other. But there are other sorts of linking problems. Sometimes, the problem to be solved is linking a dataset to itself – identifying sets of records that all refer to the same person (referred to as data deduplication). An example is a dataset of police-citizen encounters, where one wants to aggregate all of a citizen's various encounters. Other times, one wants to link two datasets together that may each contain duplicates. And in some cases, there are more than two datasets that all need to be linked to each other. In our profiles of specific software packages below, we discuss the capability of each to achieve these tasks. However, none of the software packages we reviewed supports linking three or more datasets simultaneously.

Time to implement

Our final criterion is at times unimportant but can become overwhelmingly important in some applications: the time required to implement a match. This criterion is separate from the accessibility considerations mentioned above; it refers to the time required to execute a match between two datasets once the correct software has been acquired and learned. There are two aspects of implementation time: the amount of operator time needed to configure a software package, and the amount of computer time needed subsequently to carry out the match. As noted earlier, the most time-consuming part of preparing for a match is data cleaning, and this generally cannot be avoided. The development of field-specific comparison scores can also take a long time, though some packages make this simpler than others.

Set-up time is generally shortest for unsupervised machine learning algorithms, followed by supervised machine learning – provided that the number of training pairs to be scored is not too large – and then rules-based strategies, which can take a very long time to develop.

Execution time is another matter. As we have noted, even modest-sized linking problems involve consideration of millions or even billions of potential pairs. This can be a challenge even for powerful modern computers. Some packages are perfectly suitable for modest-sized linking problems but are simply too slow to consider for larger problems. Others are optimized for large problems and scale better.

A rough guideline is that algorithms that require more set-up time then execute more quickly: rules-based algorithms can be very fast, while supervised machine learning is somewhat slower and unsupervised slower still. But this depends to a great extent on the degree to which the code has been optimized for speed – a well-written unsupervised algorithm can be much, much faster than a poorly written rules-based strategy. And aggressive blocking can make an enormous difference in speed, enough to make up for large differences in the efficiency of the post-blocking steps.

Software Profiles

There are many different software packages available for the different approaches to data linking. These packages typically include components that implement Steps 2 (blocking), 3 (field scoring), and 4 (pair scoring), but assume that the operator has previously completed Step 1 as a preprocessing stage. Bearing the above criteria in mind, we briefly summarize some of the most

common packages currently used linking here. Next, we present empirical results from our tests of some of these approaches.

Hand-coded rules

Rules-based linking using human-defined rules typically does not require a specialized software package. Rules can be defined directly in a general-purpose data processing language (e.g., SAS, SQL, R, or Python). There is a high return to skilled programming here, as different ways of handling the large number of pairs that must be considered can yield dramatically different performance. SAS and SQL are particularly well suited to matching, as neither requires loading the full set of potential pairs into memory at once, though careful programming can allow other software to be used as well.

When coding rules-based linking by hand, it is tempting to combine blocking-style rules (Step 2) with more sophisticated field-level comparisons (Step 3) in a single step. But this must be done carefully. The goal of blocking is to eliminate cases based on quick, rough assessments, so that the computationally difficult calculations that may be used in Step 3 can be applied only to cases where they are helpful. **Thus it is helpful to maintain the order we describe above**; first, blocking the data to retain only candidate pairs that may constitute a match based on our blocking criteria, and only then implement more computationally expensive field comparisons such as string distance calculations.

RecordLinkage (Python)

RecordLinkage is an open-source Python library that simplifies the linking process by providing an easily customizable blueprint for Steps 2, 3 and 4. The package can support rules-based, supervised, and unsupervised machine learning methods for Step 4 ("Python Record Linkage Toolkit", n.d.).

RecordLinkage explicitly splits Steps 2, 3 and 4 into separate functions, giving users a great deal of flexibility to determine which variables to block on (Step 2), which variables to match on and how to consider them (Step 3), and what approach to use to aggregate these variable comparisons and classify matches and non-matches (Step 4). Although flexible, the algorithms are not carefully optimized for fast performance with large datasets and can be quite slow on larger problems. Generally, performance is acceptable when the number of pairs to be considered is below about 25 million (corresponding to input datasets of 5,000 records each without blocking, or larger if blocking is used). Above this point, it is likely worth considering other packages.

Dedupe (Python)

Dedupe is another Python library that is specialized for supervised machine learning. It can be used both for linking two datasets, as we have discussed here, and for the closely related problem of identifying duplicate records in a single dataset, discussed above (Gregg and Eder, 2018).

Unlike many other packages, Dedupe offers support for common data preprocessing tasks (Step 1), such as removing punctuation and standardizing the case of all string variables. Dedupe also

handles blocking (Step 2) automatically, though it provides little flexibility for the user to control this process. However, the package does offer a great deal of flexibility for Step 3, allowing the user to select which variables to include and how they will be compared.

As Dedupe is a supervised machine learning package, it relies on human training in Step 4. A trainer is presented a set of training record pairs and asked to classify each as a match, non-match, or uncertain. This is done through a rudimentary interface in the free version of the Dedupe package, while a paid version simplifies the steps and offers a graphical interface. When training is complete (after a minimum of 10 matches and 10 non-matches, though users can train for much longer if desired), Dedupe uses the classified pairs to infer a decision rule and applies it to the rest of the input dataset. Our empirical tests, discussed below, indicate quite high performance even with very short training runs.

As an alternative to hand-training, a user can also feed Dedupe a pre-labeled data file that identifies matches and non-matches. This approach is more replicable, but performance is generally lower as hand-training allows the algorithm to select cases that are most useful to it in informing the choice of rules.

Dedupe's decision rules (Step 4) yield not just classifications of pairs as matches or not, but also link scores for each identified match. These scores are numbers between 0 and 1 that represent the algorithm's judgment of the probability that the two records are actually a match, where lower scores correspond to more questionable matches. This can aid auditing, by allowing the auditor to focus on the hardest cases.

A limitation of Dedupe is that it does not allow for multiple matches – where one record in one dataset matches to several in the other. In many administrative record settings, there may be multiple component records for the same individual, and one wants to match all of them to the other information. To implement this in Dedupe, one needs to first deduplicate each of the two datasets, then link the deduplicated datasets together. But potentially useful information is lost at the deduplication stage, making this an unattractive option.[6]

Dedupe's design allows it to scale quite flexibly to extremely large datasets. It separates the development of the rules, for which it uses a sample of the full set of pairs, from their application, and it implements blocking by default. A consequence of this is that the rules do not take much longer to develop for a large matching problem than for a smaller one, while the application of these rules to the full set of pairs is quite fast.

fastLink (R)

FastLink is an R package that uses unsupervised machine learning to match records (Enamorado, Fifield, and Imai, n.d.). The open source code is very easy to adapt to any pair of datasets.

[6] An alternative that we do not explore here is to append the two datasets to be linked and then deduplicate that combined dataset. This is less flexible with respect to data measured differently in the two data sets than are the matching approaches we discuss, but can be useful when measurement is similar (e.g., when linking the same data measured at different times).

Relative to the other packages considered above, the level of human input required to run fastLink is very low. This is largely because there is no need to develop rules or train the algorithm. Users must complete the data cleaning requirements described in Step 1 prior to using fastLink to match two datasets.

Users can create blocks manually (Step 2), and fastLink can also look for clusters of records that look most similar on a given measure. Steps 3 and 4 are done independently on each block, then the matches found within the various blocks can be aggregated.

The program allows users to classify field-level matches (Step 3) either as exact matches – such that "John" would not match with "Jhon" – or as a function of their string distance, and it lets users select from a number of different string comparison options appropriate for different types of fields. FastLink also allows for field comparisons to yield "partial matches" rather than a match or non-match. This contrasts with many other packages, which may compute the string distance but then dichotomize that information into "matches" that are below a threshold distance and "non-matches" above it. FastLink allows for three categories. The thresholds for what constitute a match and a partial match can also be adjusted by the user. This additional gradation can help identify additional matches, but it is computationally more costly.

As in Dedupe, the difficult computational task of inferring a matching rule can be conducted on a subset of the two datasets to be matched, then extrapolated to the full datasets. The speed benefits are similar. However, fastLink is generally slower on large problems than is Dedupe, in part because it lacks Dedupe's capacity for automatic blocking.

As with Dedupe, fastLink assigns each pair a score between 0 and 1 that represents the probability that it is a match, and settings can be altered to adjust the threshold for accepting a pair. Unlike Dedupe, fastLink can accommodate matching problems in which there may be multiple matches for a single record.

ChoiceMaker (Java)

ChoiceMaker is another supervised machine learning method, written in Java with a custom language overlay ("ClueMaker"), that allows for highly customizable types of pairwise field comparisons ("ChoiceMaker", n.d.). The package is difficult to configure and to use and does not have robust publicly available documentation. However, the Children's Data Network have used it to great success, including linking records across several departments within the California Health and Human Services Agency. We do not present performance evidence for Choicemaker here.

BigMatch (C++)

BigMatch is one of the older linking packages in active use. Written in C++, it was built by the US Census Bureau for use as an initial processing step when linking two files, one large and one of moderate size, with the idea that its results could be further evaluated by other matching programs (Yancey, 2002).

BigMatch is particularly well suited for enormous data files, where even re-sorting the input data files is computationally expensive. It is designed to permit multiple blocking runs in succession without resorting the data in the large input file. It also incorporates an unsupervised machine learning algorithm following the initial blocking step. It does not include extensive capabilities for flexible field-level scoring (Step 3).

Users can interact with BigMatch through graphical user interfaces without themselves knowing C++. The user feeds two text files into the program, and receives a set of potential matched pairs, along with the pair's likelihood of being a match. The user can then define a threshold above which a pair would be accepted as a match. We do not present performance evidence for BigMatch here.

Exploring performance

Unfortunately, there are few general rules for the performance of different algorithms and packages – a package that performs well on a particular problem may perform quite poorly by the very same criteria on another problem. Nevertheless, there are some general patterns. To explore this, we applied three of the approaches and packages discussed above to an example linking problem. Specifically, we used Dedupe (supervised machine learning), fastLink (unsupervised machine learning), and our own rules-based approach, developed in SAS.[7]

Our problem is to link records in Michigan voter registration databases across years. We used random samples of one million records from the publicly available files listing Michigan registered voters in 2015 and 2017.[8] These datasets are very clean, so the linking problem is easier than in many real-world settings. Crucially, the data files also contain an ID field that is accurately recorded and can be used to test the quality of matches obtained from the other fields. While the voter ID is subject to change over time if, for example, someone re-registers to vote and is assigned a new ID, we assume that all records with the same voter ID refer to the same individual, and that records with different voter IDs refer to different individuals. When we implement our matching algorithms, we remove the ID field, to explore their ability to recover matches without it. Our accuracy measures are based on comparisons to matches using the ID, which we treat as reflecting the truth – if these ID-based matches generate false positives we may be under-estimating precision, while if they fail to capture some true matches then we may be over-estimating recall.

We emphasize that each of the performance metrics we consider varies importantly across linking problems. We do not claim that the Michigan database example is typical; we see it as illustrative only.[9]

---

[7] Additional resources, including code and publicly available data can be found at https://github.com/californiapolicylab/data-linking.
[8] This data was retrieved from "Registered voters in the State of Michigan, U.S.A. as of 31 October 2017" (2017).
[9] We obtained similar results when we implemented a similar battery of tests on a pair of datasets from Christen (2008).

Step 1

The Michigan data required little cleaning and preparing relative to real-world administrative data with which we have worked. The files used for this linking exercise contain fixed-width fields, with spaces filling the extra characters in the various fields. We trimmed these spaces, but did not need to adjust for inconsistencies in the recording of addresses, street types, abbreviations, and birthdates across datasets. The trimming procedures were undertaken manually before linking with fastLink and the rules-based approach, while Dedupe implements the trimming procedures by default.

Step 2

Given the large number of potential comparisons, we used birth year and gender as blocking variables in our rules-based tests, comparing only records that matched on both of these fields.[10] We did not include any explicit blocking criteria for linking runs with Dedupe and fastLink, though Dedupe includes a blocking step by default.

Step 3

The fields that we used for linking are first name, last name, middle name, birth year, gender, and address information (which includes street name, address number, city, and zip code). To construct field comparison scores, we relied on the capabilities built into the various packages. For our rules-based method, we used a common string distance measure (the Levenshtein or edit distance) for the name and street name fields, and exact matches on the other fields. For fastLink, we compared all fields based on their string distance, using the Jaro-Winkler distance measure. With Dedupe, we used exact comparisons for gender and birth year, and compared all remaining fields using Dedupe's built-in string comparator, which uses the affine gap string distance measure.

Step 4

Our rules-based method was developed based on our knowledge of the data to address issues we had uncovered. We came up with four different sets of rules by which two records could match, counting a pair of records as a match if they satisfied any of the sets. As noted earlier, we first blocked on birth year and gender. We then used the following rules to identify matches:

- **Set 1:** Exact alignment of street number and zip code and edit distance ratios less than 0.25 (on a scale of 0 to 1) for last name, first name, middle name, and street;
- **Set 2:** Exact alignment of zip code and edit distances less than 0.25 for first and last names;
- **Set 3:** Exact alignment of street number and zip code and edit distances less than 0.25 for first name and street name;
- **Set 4:** Exact alignment of gender and edit distances less than 0.25 for first, middle, and last names.

These rules were applied sequentially, and multiple matches were not allowed. Thus, if a match was identified for a pair by Set 1, any pairs containing either of the component observations were

[10] Note that with other data, it may make sense to use two overlapping blocks, keeping all records that match either on gender or on birth year. This would allow for records to err on one of the two fields. We did not employ this method due to the integrity of this data and the large number of records.

not evaluated by Sets 2-4.[11] Due to this, Set 1 – which identifies only pairs that would also be identified by Set 2 – is not redundant; in cases where there are multiple potential matches satisfying Set 2, Set 1 ensures that the pairs that are closest in other dimensions will be selected first.

Dedupe, the supervised learning model, requires training to execute Step 4. We trained the program on 100 pairs that the model selected and ran it with its default settings. We found that Dedupe seemed to prioritize recall over precision, with many fewer false negatives than false positives. We thus ran the algorithm a second time, using a higher threshold to accept pairs as matches. This amounts to requiring more confidence that a pair is a match before accepting it.

FastLink, the unsupervised machine learning algorithm, does not require training. We ran it twice as well, once using the default threshold for the probability that the match constitutes a pair, and once increasing that threshold to require almost perfect confidence that the pair is a match before accepting it.

Relative accuracy of different methods

To keep the problem manageable, we restricted the datasets to 1 million records each, then further reduced the data to consider only voters with birth years of 1985 and 1986. This created datasets small enough that each of the packages could handle them in reasonable time, yielding 36,525 records from the 2015 file and 30,493 records from the 2017 file. Of the more than 1 billion possible pairs between these two datasets there are 20,332 true matches.

Table 3 presents the results of our tests, as well as the results of an exact match between the two datasets as a baseline. To be considered an exact match, a pair must match perfectly on all fields used in the other matching exercises (first name, last name, middle name, birth year, gender, and address information). Unsurprisingly, exact matching yields near-perfect precision, but a lower recall score than the other methods – it misses twice as many pairs as the next method (rules-based).

Generally, we saw little difference among the different algorithmic approaches in overall accuracy. All of the methods were extremely accurate, with precision scores uniformly above 0.90 and as high as 0.99 for the more restrictive Dedupe run, and with recall scores of 0.98 or higher. In other words, the methods missed very, very few true pairs, but (with one exception) falsely identified somewhat more non-matching pairs as matches. As expected, when we used fastLink with more restrictive settings, precision improved, though in these data recall did not get meaningfully worse. The settings had less of an impact for Dedupe, which achieved high precision and recall on both runs (though there is still clearly a tradeoff, albeit small, between precision and recall across the two Dedupe runs). With the exception of the less restrictive fastLink run, accuracy was high enough in each iteration for just about any linking application. This likely reflects the high quality of the input data; in other, more difficult problems, accuracy will be lower.

[11] In principle, a single rule could have identified multiple matched pairs for a single observation. In practice, this happened only once. We reviewed the pairs manually to identify the correct match in this case.

Table 3: Accuracy Results

Method	Pairs found	True matches found	False positives	False negatives	Precision	Recall
Exact Matching	19,437	19,426	11	906	1.00	0.96
Dedupe (less restrictive)	20,782	20,145	637	187	0.97	0.99
Dedupe (more restrictive)	20,028	19,910	118	422	0.99	0.98
fastLink (less restrictive)	22,671	20,317	2,354	15	0.90	1.00
fastLink (more restrictive)	21,075	20,315	760	17	0.96	1.00
Rules-based	20,443	19,865	578	467	0.97	0.98

We were interested in whether the methods made the same mistakes – whether the false positives and false negatives in one method were the same as in another. Comparing the rules-based approach with the output generated through the more restrictive settings for Dedupe and fastLink, we found that all 3 methods correctly identified the same 19,609 of the 20,332 matches. This represented 96% of the true matches in the data, and 97-99% of the true matches identified by each of the individual methods. Among the false positives, 14% (167) of all false positives incorrectly identified as a match by any method were identified by all 3 methods. The rest of the false matches were relatively idiosyncratic – there was a great deal of overlap between the false positives identified by the rules-based approach and Dedupe, though fastLink's mistakes differed from the others.

Given the limited overlap of the mistakes made by the three methods, there might be the potential to improve accuracy by using an "ensemble" approach. We considered two, and the results are presented in **Table 4** below. The first, designed to maximize recall, counted a pair as a match if it was identified as such by any of the three methods. (We used the more restrictive Dedupe and fastLink runs for this, to avoid generating too many false positives.) This strategy produced 4 false negatives (for a recall rate of 0.9998), but 1,170 false positives (for a precision of 0.95). Our second ensemble method counted a pair as a match only if it was identified as such by 2 of the three methods. (Here, we used the less restrictive runs). This had better precision, 0.97, with only an almost undetectable sacrifice in recall. Ensemble methods appear worth exploring in situations where implementation time is not a major concern and maximizing either precision or recall is an important objective.

Table 4: Accuracy results of ensemble methods

Method	Pairs found	True matches found	False positives	False negatives	Precision	Recall
1 out of 3	21,498	20,328	1,170	4	0.95	1.00
2 out of 3	20,880	20,299	581	33	0.97	0.99

Reproducibility

Our second investigation focused on the reproducibility of the set of matches identified by a given run of a program. Both rules-based and unsupervised machine learning methods can be expected to generate the same set of matches each time they are run, assuming that the input data and rules or parameters don't change,[12] but supervised machine learning algorithms depend on the decisions made by the human trainer. These decisions are judgment calls, and in ambiguous cases different trainers can make different decisions. We were interested in how important this is to the consistency of the set of matches identified.

To test the variability of Dedupe, the only supervised machine learning approach we explored in depth, two different individuals trained the model to match the same two Michigan datasets. Each trainer considered 100 training pairs, allowing the algorithm to select them. (The algorithm selected different pairs in each case, so the runs differed not just in the trainers' judgments but in the pairs they were asked to evaluate.) We also trained the process a third time, again using 100 training pairs but this time referring to the ID key as a reference to ensure that all training decisions were made correctly. This run, which we refer to as the "ideal run" below, can be seen as an upper bound to the accuracy that would be achievable by even the best human trainer in a relatively short training session.

We used the same samples used in the previous accuracy assessment, with 20,332 true matches along with 10,000-15,000 unmatched observations in each component sample.

[12] Some methods depend on a random number generated within the algorithm, or perhaps on the original sort order of the data. In these cases, reproducibility depends on beginning with the same sort order and using the same random number seed.

Figure 2: Agreement on true matches across training runs

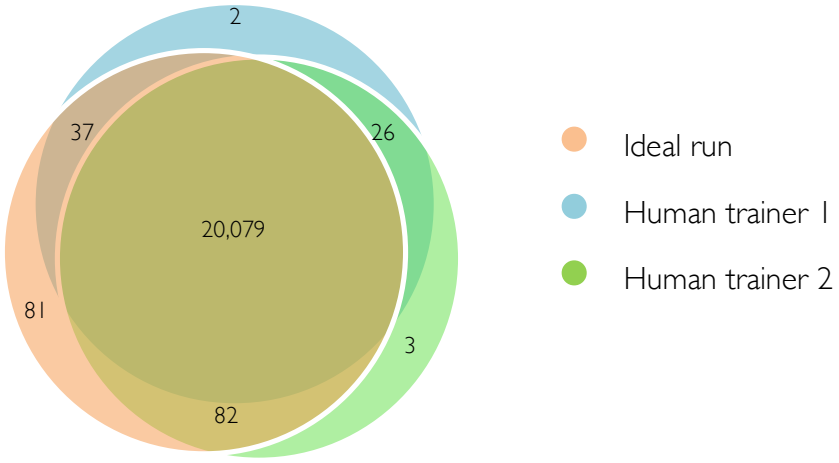


Figure 2 demonstrates the very high agreement on true matches across the three training runs. All three runs correctly identified 99% of the true matches in the data. 99.6% of the pairs identified by Trainer 1's run were also identified by Trainer 2, and 99% of the pairs identified by the ideal run were identified by one or both of the human trainers. The ideal run identified only 81 true matches that were not identified by either of the human-trained runs. Moreover, there were only 22 true matches that were not identified by any run.

These results are naturally affected by the cleanness of the data on which we ran these tests – it is possible, and even probable, that the variability across trainers would be greater in settings where messier data yields more ambiguous pairings, making it difficult to distinguish between true matches and non-matches.

False positives were more of a problem than false negatives in these data. There were 747 pairs that were falsely identified as matches by one or more of the methods. Of these, nearly all were identified by multiple methods: 66% were identified as pairs by both of the human-trained runs, but not the ideal run, and 16% were identified by all three runs. Only 93 pairs were falsely identified by the ideal run, but not by either human trainer, and only 40 pairs were falsely identified by one of the human trainers but not the other. This suggests that the human trainers tended to make mistakes in the same way – classifying the same or similar non-pairs as pairs during the training, which led the models they trained to identify the same false positives, by and large.

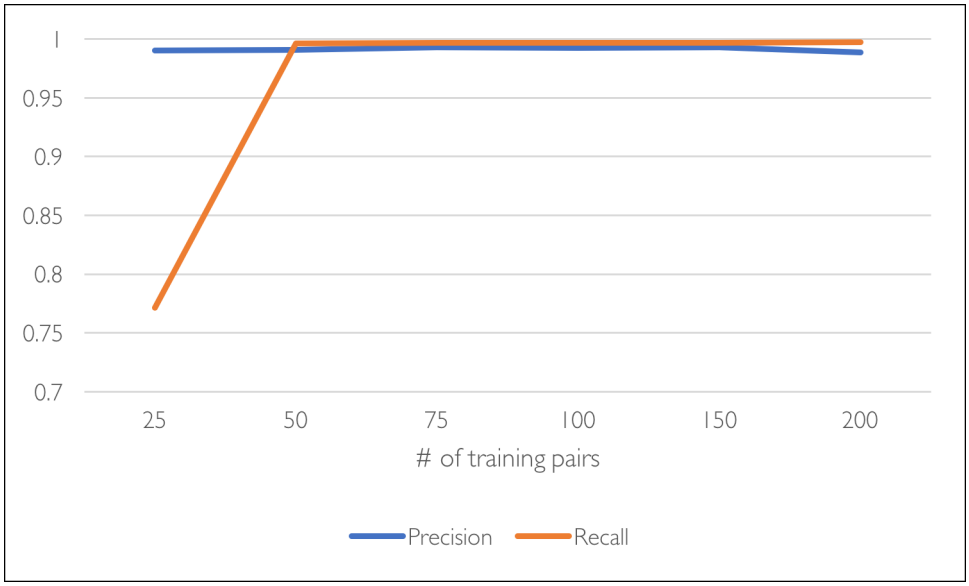
Overall, these results point to a high degree of accuracy and stability in the Michigan voter matching exercise, despite differences in who the trainer is or how the model is trained. While these results are subject to change in different settings and with different data, this finding suggests that sensitivity to operator training may be a smaller issue than we initially expected. This is an important expectation to carry forward to real-world linking exercises where the true matches across datasets are not known.

Sensitivity to amount of training

Even more important than variability across training runs is the impact of additional training. In the above analysis, each trainer considered only 100 pairs when training Dedupe. This surely produces some inaccuracy that could be avoided with a larger investment in training.

Surprisingly, this intuition was only partly borne out in our analysis. We implemented several Dedupe matching runs on the same Michigan samples, increasing the number of training pairs from 25 up to 200. **Figure 3** shows how precision and recall varied across these runs. Precision was extremely high even on the shortest training run; it remained steadily at 0.99 across all runs, while recall improved from 0.77 to 0.99 when we increased the number of training pairs from 25 to 50, but improved very little beyond that.

Figure 3: Training sensitivity



Overall, we found no benefit in terms of accuracy to training runs of more than 50 pairs – all were equally precise, and recall was steadily high beyond a training on 50 pairs. For clean datasets like these, minimal training is required to achieve high precision and recall. The amount of training required for accurate results is likely to be higher where the datasets are messier.

Scalability

A final important criterion is execution time. In large linking problems, this can become unmanageable, and it is important to choose methods that handle computationally difficult work efficiently. To assess this, we tested each method on samples of varying sizes from the Michigan data, using a powerful server available for this purpose at the California Policy Lab. We considered

problems of six sizes, linking two initial datasets of n=1,000, 5,000, 10,000, 15,000, 100,000, and 500,000 observations. **Table 5** below shows the runtime (in seconds) for each method at each sample size.

Table 5: Runtimes by size of input datasets

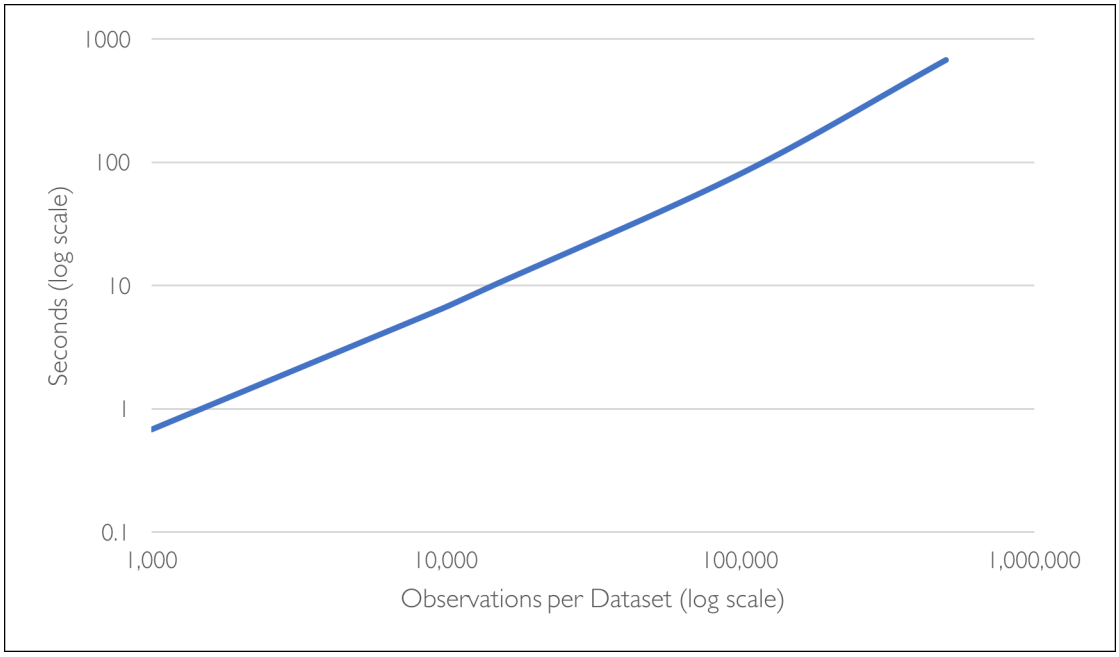
n	Possible pairs	Rules-based	Dedupe	fastLink
1,000	1 million	0.63 seconds	0.68 seconds	65.35 seconds
5,000	25 million	8.10 seconds	3.36 seconds	126.19 seconds
10,000	100 million	30.55 seconds	6.70 seconds	216.64 seconds
15,000	225 million	91.68 seconds	10.41 seconds	327.12 seconds
100,000	10 billion	73.3 minutes	80.31 seconds	100.1 minutes
500,000	250 billion	30.9 hours	11.2 minutes	48.1 hours

There was wide variability across the methods. Dedupe was fastest by a large margin, on all but the smallest problem. Our rules-based method, implemented in SAS, was acceptably fast for small problems, but execution time rose more than linearly with the size of the component datasets; it appears that this approach becomes prohibitively slow for a problem of size n=500,000 or bigger. FastLink was the slowest method across the board, though it, too, is acceptably fast for smaller problems and only becomes unmanageably slow at around n=500,000 where it took a full 48 hours.

The performance of Dedupe was the most remarkable. **Figure 4** shows how its execution time grew with the number of records in the component datasets. The curve is very nearly linear, indicating that execution time rises roughly proportionally with the number of records across a very large range, even though the number of pairs that must be considered rises much faster, with the square of the number of records. This is attributable to two factors: first, this table reports on execution time alone and does not include set-up time, which for Dedupe includes the hand-training of the model. Second, Dedupe's default settings block datasets before linking them, meaning that the number of possible pairs to evaluate is greatly reduced before Dedupe does any time consuming field-level comparisons.

Nevertheless, the speed with which Dedupe matched the various sized datasets relative to the other methods tested constitutes a strong argument for this approach in very large linking problems.

Figure 4: Dedupe run time on unblocked datasets (log-log scale)



Conclusions

Linked data are extremely valuable, both for program administration and for research. But available data do not always have unique, overlapping identifiers. In these cases, linking requires combining imperfect information from the two datasets, attempting to make judgments about which records match (often despite some inconsistencies), and which records do not.

This is extremely time consuming to carry out by hand. Fortunately, there are a number of different computer packages available that can perform data matching, using a variety of strategies. These can often achieve accurate matches in a fraction of the time. The extent to which one would want to use resulting linkages for individual decisions without further confirmation would vary based on thresholds for accuracy, but for many purposes, including most research, the achievable accuracy is quite adequate.[13] Even when near-complete confidence in the individual matches is required, automated linking algorithms can be a first step that leads to further confirmation efforts where appropriate.

[13] Tahamont et al. (forthcoming) explore how imprecise matching impacts the quality and reliability of different types of economic analyses.

This paper has reviewed the most common families of data matching strategies, the steps required to implement them, the hazards that an analyst may encounter, and the criteria that may be used to decide among alternative approaches. There is no one-size-fits-all strategy for data linking. A successful matching exercise requires close knowledge of the datasets to be matched, the types of errors that arise in these data, and a careful consideration of the objective of the exercise. In some cases, the most important criterion is to minimize the number of matches that are incorrectly identified; in others, it is to recover as many of the true matches as possible; and in still others the major problem is to construct an algorithm that is implementable in a reasonable timeframe given the size of the datasets involved and the amount of available computing power. The exercise will be most successful if analytical choices are made in pursuit of the appropriate objectives.

Sources

Ball, P. (2016). "A Geeky Deep-Dive: Database Deduplication To Identify Victims Of Human Rights Violations." Available from: https://hrdag.org/2016/01/08/a-geeky-deep-dive-database-deduplication-to-identify-victims-of-human-rights-violations/

Bell, A. et al. (2017). "Who Becomes an Inventor in America? The Importance of Exposure to Innovation." Available from: http://www.equality-of-opportunity.org/assets/documents/inventors_paper.pdf

Chetty, R. et al. (2011). "How Does Your Kindergarten Classroom Affect Your Earnings? Evidence from Project STAR," *The Quarterly Journal of Economics*, Oxford University Press, vol. 126(4), pages 1593-1660. Available from: http://www.nber.org/papers/w16381

Chetty, R., Friedman, J. and Rockoff, J. (2014). "Measuring the Impacts of Teachers I: Evaluating Bias in Teacher Value-Added Estimates." *American Economic Review*, 104(9): 2593-2632. Available from: https://www.aeaweb.org/articles?id=10.1257/aer.104.9.2593.

"ChoiceMaker" (n.d.). Available from: https://oscmt.sourceforge.io/content/intro/index.html

Christen, P. (2008). "Febrl – An Open Source Data Cleaning, Deduplication and Record Linkage System with a Graphical User." Available from: http://users.cecs.anu.edu.au/~Peter.Christen/publications/kdd2008christen-febrl-demo.pdf

Christen, P. (2012). *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Berlin: Springer Science & Business Media.

Dunn, H. (1946). "Record Linkage." *American Journal of Public Health 36*(12),1412-16

Dusetzina SB, Tyree S, Meyer AM, et al. (2014) Linking Data for Health Services Research: A Framework and Instructional Guide [Internet]. Rockville (MD): Agency for Healthcare Research and Quality (US). Available from: https://www.ncbi.nlm.nih.gov/books/NBK253313/

Enamorado, T., Fifield, B., and Imai, K. ``fastLink: Fast Probabilistic Record Linkage." available through The Comprehensive R Archive Network. https://cran.r-project.org/web/packages/fastLink/index.html

Enamorado, T., Fifield, B., and Imai, K. (2017). Using a Probabilistic Model to Assist Merging of Large-scale Administrative Records. *American Political Science Review*, Forthcoming. Available from: https://imai.fas.harvard.edu/research/files/linkage.pdf

Fellegi, I.P. and Sunter, A.B. (1969). A theory for record linkage. *Journal of the American Statistical Association, 64*(328):1183-1210

Gilleland, M. (n.d.) "Levenshtein Distance, in Three Flavors." Available from: https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm

Gregg, F. and Eder, D. (2018). Dedupe. https://github.com/dedupeio/dedupe.

Harron, K. (2016). "Introduction to Data Linkage" in Mackay, E. & Elliot, M. (Eds.) *Better Knowledge Better Society*. Essex, UK: Administrative Data Research Network

Kho, A. N., Cashy, J. P., Jackson, K. L., Pah, A. R., Goel, S., Boehnke, J., Humphries, J. E., Kominers, S. D., Hota, B. N., Sims, S. A., Malin, B. A., French, D. D., Walunas, T. L., Meltzer, D. O., Kaleba, E. O., Jones, R. C., … Galanter, W. L. (2015). Design and implementation of a privacy preserving electronic health record linkage tool in Chicago. *Journal of the American Medical Informatics Association : JAMIA*, 22(5), 1072-80. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5009931/

Kopcke, H. and Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69: 197-210

"Python Record Linkage Toolkit" (n.d.). Available from: https://recordlinkage.readthedocs.io/en/latest/index.html#

"Registered voters in the State of Michigan, U.S.A. as of 31 October 2017". (2017). Retrieved from: http://michiganvoters.info/

Yancey, William E. (2002). BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage. U.S. Bureau of the Census, Statistical Research Division. Available from: https://www.census.gov/srd/papers/pdf/rrc2002-01.pdf