Noname manuscript No.
(will be inserted by the editor)

Computing expectations and marginal likelihoods for permutations

Ben Powell · Paul A. Smith

Received: date / Accepted: date

Abstract This paper demonstrates how we can re-purpose sophisticated algorithms from a range of fields to help us compute expected permutations and marginal likelihoods. The results are of particular use in the fields of record linkage or identity resolution, where we are interested in finding pairs of records across data sets that refer to the same individual. All calculations discussed can be reproduced with the accompanying R package `expperm`.

Keywords linkage error · identity resolution · object tracking · random permutation · matrix permanent

1 Introduction

1.1 Context

The work presented here is motivated by the need to make best use of multiple, potentially inconsistent, data sources pertaining to a common population of individuals. More specifically, we are concerned with quantifying the probabilities that particular individuals in one data set correspond with particular individuals in another.

Note that in this document we will use the word 'match' to refer to a pair of records that truly do correspond to the same individual. A 'link' or 'assignment' refers to a pairing for records that we have proposed. A 'linkage solution' or 'assignment solution' refers to a set of proposed pairings of records. Given an initial assignment solution, we can permute the entries of one of the

B. Powell
University of York, UK
E-mail: ben.powell@york.ac.uk

P.A. Smith
University of Southampton, UK
E-mail: p.a.smith@soton.ac.uk

data sets in order to produce a new assignment solution. In this way we can identify assignment solutions with permutations. An 'expected assignment' refers to a probability-weighted mean of assignments. More precisely, we will identify an expected assignment with the expected effect of the corresponding permutation on the rows of an identity matrix. Although this description is made precise in Section 2, the following example helps us illustrate the concept more intuitively. Suppose we have a pair of data sets, each containing three records. The first data set contains records with labels $\{1, 2, 3\}$ and the second data set has records with labels $\{a, b, c\}$. Suppose also that we initially assign record 1 to record a, record 2 to record b and record 3 to record c. We can understand this assignment as a bijection

$$\begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}a\\b\\c\end{pmatrix},$$

where the elements of one vector map to corresponding elements in the other. Alternative assignments can then be written as assignments to permuted versions of the initial assignment's image set, e.g.

$$\begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}a\\b\\c\end{pmatrix} = \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}a\\b\\c\end{pmatrix}, \qquad \begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}a\\c\\b\end{pmatrix} = \begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}\begin{pmatrix}a\\b\\c\end{pmatrix},$$
$$\begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}c\\a\\b\end{pmatrix} = \begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}\begin{pmatrix}a\\b\\c\end{pmatrix}, \qquad \begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}c\\b\\a\end{pmatrix} = \begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}\begin{pmatrix}a\\b\\c\end{pmatrix},$$
$$\begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}b\\c\\a\end{pmatrix} = \begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}\begin{pmatrix}a\\b\\c\end{pmatrix}, \qquad \begin{pmatrix}1\\2\\3\end{pmatrix} \mapsto \begin{pmatrix}b\\a\\c\end{pmatrix} = \begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix}\begin{pmatrix}a\\b\\c\end{pmatrix},$$

where the matrices of zeros and ones encode the permutations. It is via an expectation for this permutation matrix that we will encode an expected assignment solution.

The identification and assessment of a good set of assignments between individuals is a well-studied problem. Indeed, the problem appears in many different fields of application, where the context adds significantly to the manner in which it is approached. In official, or administrative, statistics, for example, the problem is studied under the title of *record linkage* or *identity resolution.* [1] was a landmark paper in this field and established a framework for many further developments, including Bayesian interpretations and extensions of the methodology ([2]). [3] provide an authoritative review of many of these developments and illustrate their use in several case studies. The data sets involved in the official statistics setting are often extremely large, but the evidence on which linkage is based tends to be relatively good. Official statisticians might, for instance, be interested in linking individuals' records based on different versions of their name and temporally fuzzy information on their address. As well as papers on the theory of record linkage, the literature includes reviews of its practical implementation at National Statistical Institutes around the world including Italy's Istat ([4]) and the US Census Bureau ([5]).

Another application of record linkage problems arises when researchers find multiple versions of the same paper from an Internet search drawing on multiple databases. Correcting this failure to match records is known as deduplication. A review of this problem is provided by [6].

In signal processing, the problem occurs as part of *object tracking* (see [7], for example). In this context we are interested in identifying the same objects in different images, perhaps recorded at different times or from different locations. Here, the number of individuals to link is commonly smaller, but the identifying information is much more variable.

The identification of an optimal, most likely linkage solution is arguably the most extensively researched aspect of the linkage problem. The relevant optimization problem can be formulated as a linear program and solved exactly using highly-optimized routines, namely auction algorithms (see [8]). Around the set of optimal assignments, however, exist a large number of alternative assignments to which we can attribute varying degrees of likelihood. Given a prior distribution over assignments, the likelihood induces a posterior distribution encoding meaningful probabilistic statements about the true but unknown assignment. In particular the posterior distribution admits the computation of a posterior expectation.

Procedures for computing expected assignments have received less attention than those for computing optimal assignments. One reason for this is the combinatorial explosion in the number of possible assignments to average over, the vast majority of which need never be visited by an optimization algorithm. In this paper we describe and discuss a range of procedures for computing expected assignments.

Note that for the time being we will assume we are given a method for assessing the plausibility, or likelihood, of any two measurements genuinely corresponding to a common individual. Typically such likelihoods will be computed as exponentiated and normalized distances between noisy observations. In the context of official statistics, for example, the distance could be an edit-distance between character strings, while in the context of object tracking the distance could actually be a physical distance or a deviation from a given trajectory.

We will also assume that all individuals have measurements in both data sets. Although it is commonly appropriate to deviate from the latter assumption, doing so will distract significantly from the methods we discuss below. We reserve discussion of this issue for Section 5.

To our knowledge there are no existing R packages available for efficient computation of expected permutations, nor of permanents, whose relevance to the assignment problem we discuss in Section 2. There are, however, packages employing similar algorithms to those presented below to perform calculations related to permutation tests. The package `pspearman` ([9]), for example, employs Ryser's inclusion-exclusion method to compute a p-value as part of a hypothesis test for Spearman's rank correlation. Several packages, including `permute` ([10]) and `permutations` ([11]), include functions for enumerating all permutations of a small number of items. These can help us compute expected permutations, but, as we will see below, complete enumeration is liable to be an untenably demanding approach to take for all but the smallest problems.

1.2 The practical value of expected permutations and marginal likelihoods

An expected assignment, conditioned on observed data, is liable to be of interest to statisticians in its own right. Although, in general not actually describing a single assignment, it may still be seen as an estimator for the true assignment. Further, this estimator has the potential, via the use of partial assignments, to convey far more information about the appropriate degree of confidence to invest in it, than does a single, optimal assignment. The estimator is also liable to be more robust to latching onto solutions that, while having high likelihood, are supported by very little probability mass. The expectation for the true assignment also allows us to compute an expectation for the correct variables from one data set to associate with a measurement in the other data set. In turn, we can compute expectations for estimators that are linear in the first variable. Examples of using the expected permutation matrix in this way in order to estimate regression coefficients in the presence of linkage error can be found in [12], [13] and [14].

The marginal likelihood for any assignment of the records is the average over permutations of the product of likelihoods for each individual assignment. This statistic, made precise in equation (8), quantifies the evidence in support of our modelling assumptions. These assumptions include: the appropriateness of the chosen distance, and corresponding likelihood function, for potentially matching pairs of individuals; and the idea that all individuals really do have a correct match in the data set. The potential to quantify the likelihood of the latter assumption holding is particularly important due to its relevance to data 'blocking' or 'clustering'. At least in the context of official statistics, sorting large numbers of records into small blocks, in which their true match is assumed to belong, is important because some statistical procedures for making inferences in the presence of linkage error are only feasible due the decomposability blocking brings about. Small, but plausibly correct blocks are also of practical utility when records lacking a single, definitively most likely assignment are designated for manual (human) checking. For example, a human is likely to be able to spot quickly the correct matches in small blocks of records, but small blocks may be less likely to contain correct matches. A clustering algorithm based on the marginal likelihood for block membership can be used to make this compromise precise and to compose blocks optimized for manual checking. Record blocking has motivated its own small literature, with notable contributions from [15], [16] and [17].

It is not the purpose of the current paper to formally investigate the relative properties of optimal and expected assignments. We will pursue neither the assessment of likelihood functions, nor algorithms for data-blocking. These topics are mentioned here to motivate the algorithms described below. Our own work on blocking procedures, however, is in progress and indeed makes use of the algorithms.

2 Mathematical background

To assist in explaining the meaning and significance of various mathematical quantities and procedures we find it convenient to refer to a particular idealized assignment problem. In this problem, individuals have personal records stored on a central database that has been carefully constructed and well looked after. We imagine that the database needs to be augmented as a matter of urgency with a new variable. A survey is sent to all n members of the population requesting that they provide the extra data along with a binary ID code that will allow their data to be correctly added to their record in the database. We know, however, that, with a given probability q, an individual will misremember a digit in their ID code. We thus question our ability to correctly assign the new data to the old records.

In our example, the degree of correspondence between ID codes in the database and the ID codes from the survey can be expressed in the form of a distance matrix D with elements

$$D_{ij} = |d_i - s_j|_0, \tag{1}$$

where d_i and s_j denote database and survey ID codes and $|\cdot|_0$ denotes a Hamming distance that simply counts the positions in which the codes differ. Given knowledge of the error process for the survey ID codes and the relative frequencies of the database ID codes, the distances can be used to specify likelihoods for all potentially matching pairs,

$$A_{ij} = \pi(s_j, d_i) = \pi(s_j \mid d_i)\pi(d_i) = (1-q)q^{D_{ij}}\pi(d_i). \tag{2}$$

Looking ahead, we will see that all likelihoods for assignment solutions will include the marginal $\pi(d_i)$ terms via the product $\prod_{j=1}^{n} \pi(d_j)$. This product will disappear when the posterior induced by the likelihood is normalized. These terms are also redundant when we treat marginal likelihoods as conditional on the d_i. This is not unreasonable since we are generally not interested in the process by which the ID codes were generated.

Let P_{ij} be the indicator function for the event 'database record i truly corresponds with survey record j'. Collectively, these random variables form the elements of a permutation matrix P that, when pre-multiplying a column vector of survey data, reorders them to match up with the data in the database.

The elements of a valid permutation matrix are either zero or one, and their row- and column-sums are equal to one. They are also orthogonal matrices, satisfying

$$P^T P = I, \tag{3}$$

where I is the $n \times n$ identity matrix. As a consequence, pre-multiplying a column vector by a permutation matrix and then its transpose returns the original column vector. In this way the transposed permutation matrix undoes the permutation. This can be made more explicit by considering the equations

$$y' = Py, \qquad y = P^T y', \qquad y, y' \in \Re^n, \tag{4}$$

where a 1 in the i^{th} row and j^{th} column of P means that the permutation matrix effectively assigns the j^{th} element of y to the i^{th} element of y'. The matrix P^T has a corresponding 1 in its j^{th} row and i^{th} column and sends the i^{th} element of y', which is the j^{th} element of y, back to the j^{th} position. A more practical consequence of the orthogonality of the permutation matrices is that we can consider permutation matrices either to rearrange the survey records to match the database records, or to rearrange the database records to match the survey records. The latter will just be transposes of the former.

Note that a general linear combination of permutation matrices, such as an expected permutation matrix, is not necessarily itself a permutation matrix since its elements can be non-binary. It is also not necessarily unitary. Its elements are still non-negative, however, and it still possesses rows and columns that sum to one.

The likelihood for any particular permutation matrix P is the product of likelihoods for all the assignments it encodes,

$$\pi(D \mid P) = \prod_{i,j=1}^{n} A_{ij}^{P_{ij}}, \tag{5}$$

where the role of the permutation matrix here is to pick out likelihood terms to multiply together.

In the absence of any extra prior information, the likelihood function induces an unnormalized joint posterior density over the elements of the true permutation matrix. We could use this to compute the relative posterior probabilities of two complete sets of proposed assignments for example. What we cannot do with it, at least directly, is compute marginal expected values for each element of P.

The difficulty here arises from the fact that while the likelihood for the whole matrix factorizes conveniently into terms corresponding to each of its elements, the effective prior, saying that P is a permutation matrix, does not. This is because of the correlations induced by the sum-to-one constraints for its rows and columns.

The marginal expectations for the entries of P can be written as the ratio of marginal likelihoods

$$\pi(P_{ij} = 1 \mid D) = \frac{\sum_{P \in \mathcal{M} \cap P_{ij}=1} \pi(D \mid P)}{\sum_{P \in \mathcal{M}} \pi(D \mid P)} \tag{6}$$

where we sum over all permutation matrices, denoted $\mathcal{M}$ here, and all permutations such that $P_{ij} = 1$ in the denominator and numerator, respectively.

The sums of products of likelihoods that we encounter in equation (6) can be identified with instances of an algebraic device called the *permanent*. This realization unlocks a significant literature of theoretical results and numerical experiments for us to draw on. A particularly nice guide to various statistical problems to which the permanent is relevant is given in [18].

The permanent of a matrix is defined as a sum of products of its elements. These elements are selected from each row and each column according to a particular permutation. The sum is then taken over all permutations. We can write this as

$$\operatorname{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)} = \sum_{P \in \mathcal{M}} \prod_{i,j=1}^{n} A_{ij}^{P_{ij}}, \tag{7}$$

where σ denotes a permutation, S_n denotes the set of permutations of n objects (also known as the symmetric group) and $\mathcal{M}$ denotes the set of all permutation matrices.

We can now see that, given a uniform prior of permutations, the marginal likelihood, or evidence, for all survey ID codes being assignable to all database ID codes can be written as a scaled permanent

$$\pi(D) = \frac{1}{n!} \sum_{P \in \mathcal{M}} \prod_{i,j=1}^{n} A_{ij}^{P_{ij}} = \frac{1}{n!} \operatorname{perm}(A). \tag{8}$$

The expectation for an element of P can be written as

$$\pi(P_{ij} = 1 \mid D) = \frac{A_{ij} \operatorname{perm}(A_{(i,j)})}{\operatorname{perm}(A)}, \tag{9}$$

where $A_{(i,j)}$ denotes the sub-matrix obtained from deleting the i^{th} row and j^{th} column of A. The permanent is defined in a similar way to the determinant of a matrix,

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sign}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}, \tag{10}$$

but is significantly harder to compute. This statement was formalized by [19], who showed that computing permanents exactly is in the class of #P problems. Without further discussion of this class, it is enough to note here that improving on existing methods is likely to be highly challenging. What we can do, however, is mine the literature on permanents for methods to compute marginal likelihoods. This is trivial since the permanent and marginal likelihood differ only by a known multiplicative factor. What is more challenging is re-purposing the machinery used to compute permanents in order to efficiently compute expected permutation matrices. We describe novel approaches for doing so below.

3 A selected review of algorithms

3.1 Brute-force enumeration

Our first algorithm involves a naive, brute-force enumeration of all possible assignments. Its computational cost is of order $\mathcal{O}(n!)$, and so is only applicable for small $n < 10$. The most pleasing, and the only non-trivial, part of the algorithm is its use of Heap's algorithm ([20]) to produce the full set of assignments sequentially and without repetition. Crucially, a full list of all assignments, which is liable to fill up a computer's RAM, need never be constructed.

The following function uses Heap's algorithm to run through assignments while keeping track of a running likelihood-weighted mean. Its output is the final value of this mean, along with the total weight which is equal to the permanent of the matrix of likelihoods.

```
brute <- function(A, return.permanent = FALSE) {
    n <- nrow(A)
    cnt <- rep(0, n)
    ind <- 1:n
    indmat <- cbind(1:n, ind)
    W <- prod(A[indmat])
    EP <- diag(n)

    i <- 0
    while (i < n) {

        # The outer parts of this while loop implement Heap's algorithm for
        # enumerating permutations.
        if (cnt[i + 1] < i) {
            if (i%%2 == 0) {
                ind[c(1, i + 1)] <- ind[c(i + 1, 1)]
            } else {
                ind[c(i + 1, cnt[i + 1] + 1)] <- ind[c(cnt[i + 1] + 1, i + 1)]
            }
            cnt[i + 1] <- cnt[i + 1] + 1
            i <- 0

            # This part of the code updates the running weighted mean of permutation
            # matrices.
            indmat <- cbind(1:n, ind)
            w <- prod(A[indmat])
            W <- W + w
            EP <- EP * (1 - w/W)
            EP[indmat] <- EP[indmat] + w/W
        } else {
            cnt[i + 1] <- 0
            i <- i + 1
        }
    }
    if (return.permanent) {
        attr(EP, "permanent") <- W
    }
    EP
}
```

3.2 The Ryser method

The Ryser method, named after its first appearance in [21], involves computing the permanent of a matrix using an inclusion-exclusion argument. More specifically, it cleverly breaks the permanent down into a sum of terms that correspond to intersections of unions of assignments,

$$\text{perm}(A) = \sum_{\mathcal{J}\subseteq\mathcal{I}} (-1)^{|\mathcal{I}|-|\mathcal{J}|} \prod_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} A_{i,j}, \qquad \mathcal{I} = \{1, 2, \ldots, n\}. \tag{11}$$

The outer sum here is over all 2^n ways of selecting a subset of A's columns. The summands are products of row-sums of sub-matrices consisting only of the selected columns. The sign term alternates according to the parity of the number of sets whose intersection the summand corresponds to.

The Ryser method effectively reduces the computational cost of the permanent from $\mathcal{O}(n!n)$ to $\mathcal{O}(2^n n^2)$. In one respect this is a huge saving. However, the cost still scales badly, with the effect that only problems with $n < 20$ are practical. Enumeration of the Ryser method's summands can be achieved efficiently using a Gray code binary generator such as Knuth's algorithm L (see page 10 of [22]), or a generator of combinations such as the Cool-lex algorithm (see [23]). Like Heap's algorithm, these algorithms produce one term in a sequence from the preceding one, avoiding the construction of a full list. We have opted for Knuth's algorithm L in our implementation of the Ryser method due to the computational efficiency afforded by its lack of internal loops.

The same inclusion-exclusion strategy employed by the Ryser method for computing the permanent can be used to compute an expected permutation matrix. As far as we are aware, this strategy, implemented below, has not been exploited elsewhere in the literature. The resulting algorithm involves the selection of subsets of columns of A to set to zero, leading to modified matrices $A^{(k)}$ where k labels the subset. The modified matrix's rows are normalized by dividing them by the row sums. We then add the normalized matrix to a running sum after multiplication by a weight equal to the product of the row sums.

```
ryser <- function(A, return.permanent = FALSE) {
    n <- ncol(A)
    W <- 0
    EP <- matrix(0, n, n)
    a <- rep(0, n)
    f <- 0:n
    p <- (-1)^n
    while (f[1] < n) {

        # This part of the algorithm is Knuth's algorithm L for generating a binary
        # Gray code.
        p <- -p
        j <- f[1]
        f[1] <- 0
        f[j + 1] <- f[j + 2]
        f[j + 2] <- j + 1
        a[j + 1] <- 1 - a[j + 1]
        Ak <- A

        # This part of the algorithm is computing the terms of the incl/excl sum in
        # the same way as Ryser's method for computing the permanent.
        Ak[, a == 0] <- 0
        rs <- rowSums(Ak)
        w <- p * prod(rs)
        if (!(w == 0)) {
            Ak <- sweep(Ak, 1, rs, FUN = "/")
            W <- W + w
            EP <- EP + w/W * (Ak - EP)
        }
    }
    if (return.permanent) {
        attr(EP, "permanent") <- W
    }
    EP
}
```

3.3 The tridiagonal special case

The class of tridiagonal matrices is very special insofar as having permanents that can be computed with cost $\mathcal{O}(n)$. In terms of the assignment problem, these matrices correspond to two ordered sets of individuals that can only be assigned to others whose place in the order differs by at most one.

In [24], the authors describe the geometric relevance of the matrix permanent and many of its less obvious properties. Most importantly for us, they introduce the notion of *contractibility*, which provides the key to computing the permanent of a tridiagonal matrix efficiently. The authors define a matrix, A, to be contractible on column k if column k, denoted $A_{\cdot,k}$, contains exactly two non-zero elements. Say that these two elements reside in rows i and j. The *contraction* of A is defined to be a copy of A with its i^{th} row, denoted $A_{i,\cdot}$, replaced with $A_{j,k}A_{i,\cdot} + A_{i,k}A_{j,\cdot}$, and with the j^{th} row and k^{th} column removed. Lemma 3.2 in [24] states that the permanent of A and its contraction are equal. We note that, in the context of the assignment problem, a contraction can be understood as marginalizing, or integrating, over the possible assignments for a particular individual.

When presented with a tridiagonal A, we can contract it by taking $j = k = 1$ and $i = 2$. We can then repeatedly contract the matrix like this until it is a single number equal to the permanent of the original matrix. It can be shown that the top-left elements of the sequence of contracted matrices in fact provide us with the permanents of submatrices of A. Explicitly, defining the sequence of contracted matrices $F^{(m)} \in \Re^{(n-m)\times(n-m)}$ as

$$F^{(m)} = \begin{cases} A & m = 0, \\ \mathrm{C}(F^{(m-1)} : j = k = 1, i = 2) & m = 1, \ldots, n-1, \end{cases} \tag{12}$$

where $\mathrm{C}(\cdot : j = k = 1, i = 2)$ denotes the result of the contraction procedure described above, it is the case that

$$F_{1,1}^{(m-1)} = \mathrm{perm}(A_{1:m,1:m}), \tag{13}$$

where $A_{1:m,1:m}$ is the top-left m by m submatrix of A. Similarly, we can contract from the bottom-right to produce the sequence of contractions $B^{(m)} \in \Re^{(n-m)\times(n-m)}$ such that

$$B^{(m)} = \begin{cases} A & m = 0, \\ \mathrm{C}(B^{(m-1)} : j = k = n-m, i = n-m-1) & m = 1, \ldots, n-1, \end{cases} \tag{14}$$

and

$$B_{n-m,n-m}^{(m)} = \mathrm{perm}(A_{(n-m):n,(n-m):n}). \tag{15}$$

The two sequences of permanents can be used, via equation (9), to compute the diagonals of the expected permutation matrix, i.e.

$$\pi(P_{ii} = 1 \mid D) = \frac{A_{ii}\,\mathrm{perm}(A_{(i,i)})}{\mathrm{perm}(A)} \tag{16}$$

$$= \frac{A_{ii}\,\mathrm{perm}(A_{1:(i-1),1:(i-1)})\,\mathrm{perm}(A_{(i+1):n,(i+1):n})}{\mathrm{perm}(A)}. \tag{17}$$

The off-diagonals can now be calculated by leveraging our knowledge of the row- and column-sum constraints for the expected permutation matrix, and of its symmetry. This symmetry follows from the fact that, in the tridiagonal case, the only admissible permutations are swaps between consecutive pairs. This means that if the i^{th} record in the first data set matches with the $(i+1)^{th}$ record in the second data, then the $(i+1)^{th}$ record in the first data set must match with the i^{th} record in the second data set. Since the two matching 'events' must happen simultaneously they are essentially the same event and must therefore have the same probability of occurring. The computational strategy outlined in this section is implemented with the following function, which we refer to as the BG algorithm in reference to Brualdi and Gibson, who provided us with the contraction result.

```r
BG <- function(A, return.permanent = FALSE) {
    if (!is.tridiagonal(A)) {
        warning("Input is not tridiagonal.
            This function only works for tridiagonal matrices!")
    }
    n <- nrow(A)

    # The algorithm begins by computing two sequences of permanents via the BG
    # contractions.
    Fmat <- A
    for (i in 2:n) {
        Fmat[i, i:n] <- Fmat[i - 1, i - 1] * Fmat[i, i:n] + Fmat[i, i - 1] *
            Fmat[i - 1, i:n]
    }
    f <- diag(Fmat)
    Bmat <- A
    for (i in (n - 1):1) {
        Bmat[i, 1:i] <- Bmat[i + 1, i + 1] * Bmat[i, 1:i] + Bmat[i, i + 1] *
            Bmat[i + 1, 1:i]
    }
    b <- diag(Bmat)

    # The permanents are used to compute the diagonal of EP, the off-diagonals
    # are determined by the sum-to-one constraints.
    EP <- matrix(0, n, n)
    EP[1, 1] <- A[1, 1] * b[2]/b[1]
    EP[n, n] <- A[n, n] * f[n - 1]/b[1]
    if (n > 2) {
        for (i in 2:(n - 1)) {
            EP[i, i] <- A[i, i] * f[i - 1] * b[i + 1]/b[1]
        }
    }
    for (i in 1:(n - 1)) {
        EP[i, i + 1] <- EP[i + 1, i] <- 1 - sum(EP[i, 1:i])
    }
    if (return.permanent) {
        attr(EP, "permanent") <- b[1]
    }
    EP
}
```

3.4 A variational approximation

The computational strategy described in this section leads only to approximations for the expected permutation and marginal likelihood. It is inspired by [25], whose interest in permanents is motivated by the object tracking problem. They suggest that a set of approximate expectations, called beliefs, satisfying a minimal-energy condition are calculated. For a specific value of its hyperparameter, their energy function coincides with the Kullback-Leibler divergence between the beliefs and the likelihood. For this reason we designate the beliefs as variational approximations to the exact expectations.

Explicitly, we take as our approximation to the expectation the minimizer of

$$F(P, A) = \sum_{i,j=1}^{n} P_{ij} \log\left(\frac{P_{ij}}{A_{ij}}\right) \tag{18}$$

with respect to P, subject to sum-to-one constraints on its rows and columns.

The beliefs minimizing the divergence turn out to be surprisingly easy to compute. They can be found by applying the iterated Sinkhorn operation (see [26]) to A. This operation simply iterates between normalizing its argument's rows and columns. The function below implements the iterated Sinkhorn operation while keeping track of the products of the row- and column-sums. We can use Corollary 25 of [25] to show that the product of these products serves as an upper bound on the permanent of A.

```
sink <- function(A, maxit = 99, return.permanent.bound = FALSE) {
    n <- nrow(A)
    u <- rep(1, n)
    v <- rep(1, n)
    its <- 0
    rsums <- 0
    while (its < maxit) {
        its <- its + 1
        rsums <- rowSums(A)
        u <- rsums * u
        A <- sweep(A, 1, rsums, FUN = "/")
        csums <- colSums(A)
        v <- csums * v
        A <- sweep(A, 2, csums, FUN = "/")
    }
    W <- prod(u, v)
    if (return.permanent.bound) {
        attr(A, "permanent bound") <- W
    }
    A
}
```

The relevance of the iterated Sinkhorn operation to the variational approximation is made precise in appendix 6.1. More formal analyses of the operation may be found in [27], for example, who refer to it in terms of *Iterative Proportional Scaling* or *matrix raking*.

4 Examples

4.1 Code tests

In this section we report on a set of checks and numerical experiments to test the algorithms described above. These experiments can be replicated using the `expperm` package, which contains implementations of the algorithms in `R` and `C++` as well as a data file with the experiment inputs. The `R` code is intended to act as functional pseudocode to help the reader understand and modify the algorithms for their own purposes. The `C++` code is intended to better demonstrate the speed of the algorithms, and to provide solutions to real problems.

We begin by testing whether the algorithms actually produce the same output for a particular seven-by-seven matrix of simulated likelihoods. We observe the elements of the algorithms' outputs to differ by less than 5×10^{-14} here, suggesting that rounding errors in the manipulations and summations performed by the algorithms are not significant at this scale. We also observe the column- and row-sums of the outputs to differ from one by less than 1×10^{-13}. The reader is invited to reproduce these results using commands such as:

```
library(expperm)
data(A)
data(triA)
max(abs(ryser(A) - brute(A)))
max(abs(rowSums(ryser(A)) - 1))
```

In Figure 1 we examine the differences between an example likelihood matrix A, a maximum-likelihood permutation (maximizing (5)) and the expected permutation matrix given A. It is clear that both the maximum-likelihood and expected permutation matrices allocate high probability to the assignments

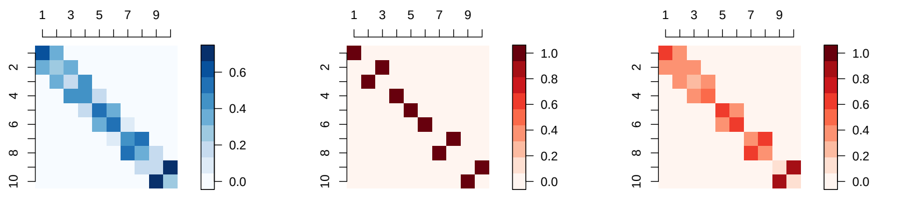

(a) An example matrix of likelihoods, A.
(b) A maximum likelihood permutation matrix given A.
(c) The expected permutation matrix given A.

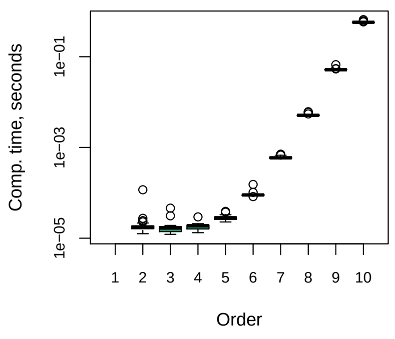
Fig. 1: Heat map plots of matrices used to test the algorithms for computing expected permutations.

with high likelihood. This can be seen from the darker colours appearing mostly in corresponding positions in the heat maps in each subfigure. It is also clear that there is more than one assignment solution with appreciable likelihood. This can be seen from the diffuseness of the shades in Figure 1c relative to Figure 1b.

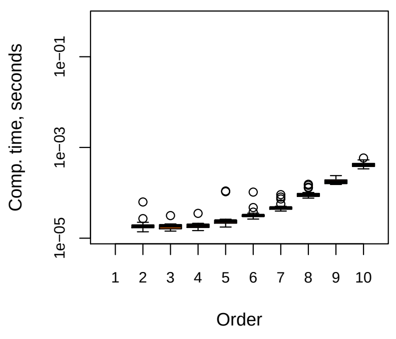
We proceed to investigate the algorithms' run-times with the help of the `microbenchmark` package of [28]. Our numerical experiments involve simulating $r = 32$ tridiagonal matrices of likelihoods for assignment problems of increasing size. The time required by each algorithm to compute corresponding expected permutation matrices is then recorded. The resulting computation times are plotted in Figure 2. We expect to see super-exponential scaling for the brute force algorithm, exponential scaling of the Ryser algorithm and the sub-exponential scaling of the variational and BG algorithms. The boxplots of Figure 2 cannot be said to verify these expectations, but appear to be approximately consistent with them. We observe that the first three algorithms return their outputs at similar speeds for matrices of order $n \approx 5$. After this point the Ryser algorithm beats the brute-force algorithm, and the variational algorithm beats the Ryser algorithm. The BG algorithm comfortably beats all the others, but is applicable only to tridiagonal likelihood matrices. Analogous experiments on simulated likelihood matrices with nonzero entries in all positions produced almost identical results for the first three algorithms, so are not presented here.

4.2 Simulated example

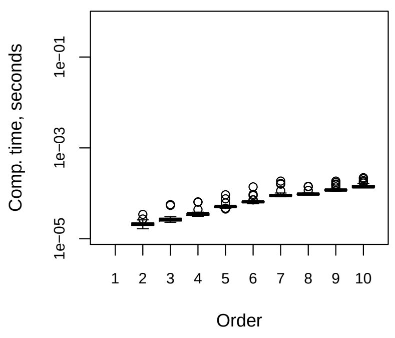
The data examined in this section were produced to facilitate a workshop on the subject of record linkage run as part of a European Statistical System network (ESSnet) project on data integration. The data, which were produced by [29] from the UK's Office for National Statistics (ONS), consist of three



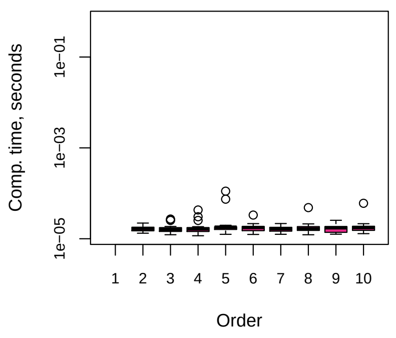
(a) Brute force algorithm.



(b) Ryser algorithm.



(c) Variational algorithm.



(d) BG algorithm.

Fig. 2: Computation times for algorithms computing expected permutation matrices from simulated matrices of likelihoods. The x-axis corresponds to the number of records to assign and the order of the resulting matrices. 32 simulations are made for each order and the resulting distribution of computation times is summarized using boxplots. Note the log-scaling of the y-axis.

sets of approximately 25000 fictitious records for individuals, all of which are subject to transcription error.

We consider now small, equally sized subsets of records from two of the data sets. To introduce more uncertainty, and so better demonstrate the value of the expected rather than optimal linkage solution, we restrict our attention only to recorded names and birth years. The data are presented in Table 1.

The subsets are selected by first running a naive, greedy assignment algorithm on the two data sets. This involves working through the records in the first data set and assigning them to their closest match in the second data set according to a Damerau-Levenshtein edit distance (see [30]), which counts the (minimum) numer of edits required to transform one word into another. All distances are thus integer valued and ties are broken arbitrarily when identifying the closest match. We select a subset of $n = 18$ assigned pairs, which are close to each other according to the edit distance.

We then produce a matrix of edit distances between the subsets of records. On the assumption that edits are independent and occur with probability $q = 0.2$, this distance matrix is used to induce a matrix of likelihoods according to equation (2). The likelihood matrix is then used to calculate an expected permutation matrix using the Ryser algorithm. Note that this expectation is conditional on all records in the subset having a match within the subset.

The distance matrix, maximum likelihood permutation matrix and the expected permutation matrix for the subset of records are presented in Figure 3. We observe that the maximum likelihood matrix, which is computed using the `R` package `lpSolve` ([31]), does not coincide with the identity matrix. This shows how the naive assignment procedure described above has failed to find an optimal assignment. Indeed, the total number of edits between assigned records in the optimal solution is 19 while the total number is 33 for the naive solution. However, the maximum likelihood solution shows us just one highly likely solution out of many candidates. Uncertainty regarding the true matches is manifested in the intermediate shades appearing in Figure 3c. In particular, the rows and columns containing no dark red cells identify records without a single convincingly most likely assignment.

We consider this example to be a useful caricature of genuine record linkage methodology. As discussed further in Section 5, it demonstrates how the results of a naive but computationally convenient assignment procedure can be analsyed and potentially called into question thanks to our algorithms.

5 Discussion

In the course of the work documented in this paper we have seen how the search for good assignments between records in a pair of data sets is a rich and demanding problem. We have seen how the problem relates to numerous subfields of pure and applied mathematics, and how contributors to those fields have provided us with tools for constructing sophisticated algorithms for computing expected permutations. The sophistication of these algorithms does not imply that they are unwieldy. Indeed they are implemented in tens of lines of code and wrapped up in our R package `expperm`, which we encourage readers to take apart and customize.

We have intentionally prioritized the exposition of computational procedures over the explanation of their role in data analysis problems. Nevertheless, appreciation of the practical relevance of the computations to the motivating

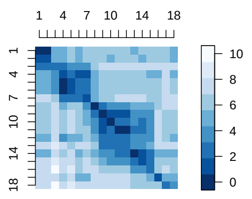
PERNAME1	PERNAME2	DOB_YEAR
ALEX	ANDERSON	1951
ALEX	ANDERSON	1952
ALEXIS	ANDERSON	1955
LEWIS	ANDERSON	1956
LEWIS	ANDERSON	1954
LEWIS	ANDERSON	1954
LEWIS	ANDEBSON	1996
ELISE	ANDERSON	1999
LIAM	ANDERSON	1999
LIAM	ANDERSON	1992
LIAM	ANDERSON	1949
LIAM	ANDERSON	1954
LIAM	ANDEVSON	1947
LILY	ANDERSON	1941
LILY	ANDEASON	1942
LILY	AWDERSOW	1966
BECKY	ANDERSON	1916
TUBY	ANDERSUN	1913

(a) Example data subset 1.

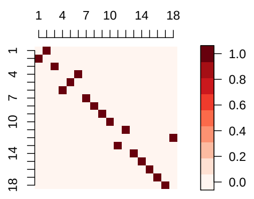
PERNAME1	PERNAME2	DOB_YEAR
ALEX	ANDERSON	1951
ALEX	ANDERSON	1951
RLEXIS	RNDERSON	1955
LEWIS	ANDERSON	1954
LEWIS	ANDEVSON	1954
LEWLS	ANDERSON	1956
LEWIS	ANDERSON	1996
ELISE	ANDERSON	1999
LIAM	ANDERSON	1999
LIAM	ANDERSON	1992
LIAM	ANDERSON	1949
LIAM	ANDERSON	1949
LILY	ANDERSON	1941
LILY	ANDERSON	1942
LILY	ANDERSON	1966
BELKY	ANDERSON	1916
TOBY	ANDERSON	1913
RUBY	ANDERSON	1956

(b) Example data subset 2.

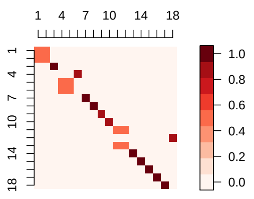
Table 1: Two subsets of the ESS data whose matching probabilities are calculated as part of Section 4.2.



(a) The distance matrix between the two subsets of records.



(b) A maximum-likelihood permutation matrix for the records.



(c) The expected permutation matrix encoding the matching probabilities for the records.

Fig. 3: Heat map plots of matrices relating to the example record linkage data of Section 4.2. The matrix rows and columns refer to the records in subsets 1 and 2 , respectively.

problems, as described in the introduction, remains important. With this in mind, we now discuss the value and limitations of the algorithms described above.

We suggest that the algorithms are used as tools for assessing a proposed assignment solution. As the expected permutations are clearly not themselves permutations, our algorithms are not tools for finding assignments and are not competing with algorithms for finding an optimal assignment. What the expected permutation matrices show very well is the potential for uncertainty for a true match. Informally, we can say that the less an expected permutation matrix looks like the maximum likelihood permutation matrix, the less confident we should be that the maximum likelihood estimate is correct. This sort of evaluation is useful even if it only scales up to moderately large blocks of proposed assignments, as in the example of Section 4.2.

Although of value in their own right, expected permutation matrices are also a vital component of statistical inference procedures, such as those described in [12], [13] and [14]. Typically these inferences are applied given an *exchangeable linkage assumption*, leading to expected permutation matrices that, although mathematically convenient, are naive, since they do not use record similarity measures in their construction. We have explained how our algorithms can help us compute more appropriate expected permutation matrices, which we would expect to improve statistical inferences, even if they were not exact due to the computational demand involved in scaling up to the whole of a large data set. We might ask, for example, how seriously does specifying the wrong expectation for linkage error undermine inference? And, how significantly is this problem improved with a less wrong misspecifation? Answering these questions will require the investigation of inferential procedures in the presence of parameter misspecificiation, which is likely to be a considerable task and one that will benefit from expert knowledge on what a 'correct' specification should really look like. This would make for valuable future work.

We have also assumed that all records to be assigned have a true match in the data sets available to us. In applications to official statistics this assumption is likely to be particularly hard to justify. We are currently working on extensions of the methods described above for which the assumption can be compromised. Specifically, we are investigating averages over assignment solutions for which some records remain unassigned or perhaps have multiple assignments due to accidental record duplication. Permutations may no longer be the objects to focus on in this new context, but related combinatorial mathematics remains relevant. As well as pursuing this methodology, it is also interesting to consider how badly it is needed - how badly deviations from the matchability assumption can undermine our inferences. We anticipate that the expected assignment for an unmatchable record will be mostly uninformative. It is not useful to know, for example, that the record with name 'George' is closer to 'Susan' than it is to 'Christobelle'. The degree to which an unmatchable record undermines the expected assignments for the matchable records is less obvious however. This robustness issue is another topic for future research.

Acknowledgements

We would like to thank an anonymous reviewer for advice and suggestions that have greatly improved this paper.

References

1. I.P. Fellegi, A.B. Sunter, Journal of the American Statistical Association **64**(328), 1183 (1969)
2. T.R. Belin, D.B. Rubin, Journal of the American Statistical Association **90**(430), 694 (1995)
3. T. Herzog, F. Scheuren, W. Winkler, *Data Quality and Record Linkage Techniques* (Springer New York, 2007)
4. N. Cibella, M. Fortini, M. Scannapieco, L. Tosco, T. Tuoto, et al., Rivista di statistica ufficiale **9**(2-3), 55 (2007)
5. W.E. Yancey, Bigmatch: A program for extracting probable matches from a large file for record linkage. Tech. Rep. 1, US Census Bureau (2002). URL https://www.census.gov/srd/papers/pdf/rrc2002-01.pdf
6. A.K. Elmagarmid, P.G. Ipeirotis, V.S. Verykios, IEEE Transactions on knowledge and data engineering **19**(1), 1 (2007)
7. H. Pasula, S. Russell, M. Ostland, Y. Ritov, in *International Joint Conferences on Artificial Intelligence*, vol. 99 (1999), vol. 99, pp. 1160–1171
8. D.P. Bertsekas, Mathematical Programming **21**(1), 152 (1981)
9. P. Savicky, *pspearman: Spearman's rank correlation test* (2014). URL https://CRAN.R-project.org/package=pspearman. R package version 0.3-0
10. G.L. Simpson, *permute: Functions for Generating Restricted Permutations of Data* (2016). URL https://CRAN.R-project.org/package=permute. R package version 0.9-4
11. R.K.S. Hankin, *permutations: Permutations of a Finite Set* (2017). URL https://CRAN.R-project.org/package=permutations. R package version 1.0-2
12. F. Scheuren, W.E. Winkler, Survey Methodology **19**(1), 39 (1993)
13. P. Lahiri, M.D. Larsen, Journal of the American Statistical Association **100**(469), 222 (2005). DOI 10.1198/016214504000001277
14. G. Kim, R. Chambers, Computational Statistics & Data Analysis **56**(9), 2756 (2012). DOI https://doi.org/10.1016/j.csda.2012.02.026
15. M. Michelson, C.A. Knoblock, in *Association for the Advancement of Artificial Intelligence* (2006), pp. 440–445
16. M. Bilenko, B. Kamath, R.J. Mooney, in *Data Mining, 2006. ICDM'06. Sixth International Conference on* (IEEE, 2006), pp. 87–96
17. S.E. Whang, D. Menestrina, G. Koutrika, M. Theobald, H. Garcia-Molina, in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data* (ACM, 2009), pp. 219–232
18. P. Diaconis, R. Graham, S.P. Holmes, Lecture Notes-Monograph Series pp. 195–222 (2001)
19. L.G. Valiant, Theoretical computer science **8**(2), 189 (1979)
20. B.R. Heap, The Computer Journal **6**(3), 293 (1963). DOI 10.1093/comjnl/6.3.293
21. H. Ryser, *Combinatorial Mathematics*. Carus mathematical monographs (Mathematical Association of America, 1963)
22. D. Knuth, *The Art of Computer Programming: Generating all tuples and permutations*. Addison-Wesley series in computer science and information proceedings (Addison-Wesley, 2005)
23. F. Ruskey, A. Williams, Discrete Mathematics **309**(17), 5305 (2009)
24. R.A. Brualdi, P.M. Gibson, Journal of Combinatorial Theory, Series A **22**(2), 194 (1977). DOI https://doi.org/10.1016/0097-3165(77)90051-6
25. M. Chertkov, A.B. Yedidia, The Journal of Machine Learning Research **14**(1), 2029 (2013)
26. R. Sinkhorn, Ann. Math. Statist. **35**(2), 876 (1964). DOI 10.1214/aoms/1177703591
27. Y. She, S. Tang, Journal of Computational and Graphical Statistics pp. 1–13 (2018)
28. O. Mersmann, *microbenchmark: Accurate Timing Functions* (2018). URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-6
29. P. McLeod, D. Heasman, I. Forbes, Simulated record linkage data. Tech. rep., Office for National Statistics (2011). URL https://ec.europa.eu/eurostat/cros/content/job-training$_en$
30. F.J. Damerau, Communications of the ACM **7**(3), 171 (1964)
31. M. Berkelaar, others, *lpSolve: Interface to 'Lpsolve' v. 5.5 to Solve Linear/Integer Programs* (2015). URL https://CRAN.R-project.org/package=lpSolve. R package version 5.6.13

6 Appendix

6.1 (Outline) Proof for the iterated Sinkhorn operation leading to the variational approximation

Consider the objective function combining the Kullback-Leibler divergence with Lagrange terms encoding sum-to-one constraints

$$\mathcal{L} = \sum_{i,j=1}^{n} \hat{P}_{ij} \log \frac{\hat{P}_{ij}}{A_{ij}} + \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{n} \hat{P}_{ik} - 1 \right) + \sum_{j=1}^{n} \eta_j \left(\sum_{k=1}^{n} \hat{P}_{kj} - 1 \right), \quad (19)$$

where λ_i and η_j are the Lagrange multipliers.

The derivative of the Lagrangian with respect to $\hat{P}_{ij}$ is

$$\frac{\partial \mathcal{L}}{\partial \hat{P}_{ij}} = \log \hat{P}_{ij} + 1 - \log A_{ij} + \lambda_i + \eta_j, \quad (20)$$

which is zero when

$$\hat{P}_{ij} = u_i A_{ij} v_i, \quad (21)$$

where

$$u_i = \exp(-1 - \lambda_i), \qquad v_j = \exp(-\eta_j). \quad (22)$$

Equation (21) is telling us that the Lagrangian has a stationary point when $\hat{P}$ is a version of A with its rows and columns scaled by factors u_i and v_j. Further, we know that these scalings must result in $\hat{P}$ having unit row- and column-sums. [26] shows us that this can always be done for an A with strictly positive elements. He also shows us that the iterative process of alternately re-normalizing the rows and columns of A allows us to approximate $\hat{P}$ to arbitrary precision.