Generalized Full Matching*

Fredrik Sävje† Michael J. Higgin‡ Jasjeet S. Sekhon§

March 8, 2017

Abstract

Matching methods are used to make units comparable on observed characteristics. Full matching can be used to derive optimal matches. However, the method has only been defined in the case of two treatment categories, it places unnecessary restrictions on the matched groups, and existing implementations are computationally intractable in large samples. As a result, the method has not been feasible in studies with large samples or complex designs. We introduce a generalization of full matching that inherits its optimality properties but allows the investigator to specify any desired structure of the matched groups over any number of treatment conditions. We also describe a new approximation algorithm to derive generalized full matchings. In the worst case, the maximum within-group dissimilarity produced by the algorithm is no worse than four times the optimal solution, but it typically performs close to on par with existing optimal algorithms when they exist. Despite its performance, the algorithm is fast and uses little memory: it terminates, on average, in linearithmic time using linear space. This enables investigators to derive well-performing matchings within minutes even in complex studies with samples of several million units.

Keywords: Causal inference; Observational study; Optimal matching; Treatment effects

*We thank Peter Aronow, Justin Grimmer, Ben Hansen, Walter Mebane, Sam Pimentel, Liz Stuart, Cyrus Samii, Yotam Shem-Tov and José Zubizarreta for helpful comments and discussions. This research is partially supported by Office of Naval Research Grants N00014-15-1-2367 and N00014-17-1-2176.

†Travers Department of Political Science & Department of Statistics, University of California, Berkeley

‡Department of Statistics, Kansas State University

§Travers Department of Political Science & Department of Statistics, University of California, Berkeley

1 Introduction

When treatment assignment is beyond the control of an investigator, direct comparisons between treatment groups will, with few exceptions, be misleading. Non-randomized assignment processes tend to favor certain types of units. Comparisons between treatment groups will, therefore, not only capture the causal effect of the treatments under study, but also systematic differences between the groups that existed before treatment was assigned. The analysis is confounded.

Under the assumption that we observe to all variables that confound the comparison, we can correct for the biases by equalizing, in expectation, the distributions of confounders between the treatment groups. A straightforward way to do the adjustment is *matching*. The method forms matched groups of similar units and weights the units proportionally to the relative prevalence of the treatment conditions within the groups. The reweighted treatment groups will have similar covariate distributions.

The construction of the matching mainly involves two considerations. First, we want to make the matched groups as homogeneous as possible. This will make the comparisons across treatment conditions, within matched groups, unconfounded under the assumption that all confounders have been observed. Second, we want to use the information in the sample as effectively as possible. Everything else equal, more variation in the weights implied by the matching leads to a higher variance of our estimator. In order to reduce the variation, we want to discard as few units as possible (i.e., to maximize the number of units assigned to groups). We also want to avoid that the groups differ too much in their composition.

The overall performance of our estimator depends both on the treatment group balance and the weight variation. The two considerations are, however, often conflicting, and it can be computational challenging to derive optimal solutions. It is common that investigators simplify the matching problem by imposing superfluous constraints and use heuristic

algorithms, both of which may produce matchings that perform poorly.

In this paper, we contribute to the matching literature in two ways. First, we introduce a generalization of *full matching*. Full matching is a flexible matching method that is optimal for a large set of common cases (Rosenbaum 1991, Hansen 2004). In particular, among all matching methods that do not leave units unassigned, full matching has the least within-group heterogeneity. The method is, however, restricted to studies with two treatment conditions where the investigator requires no more than one unit of each treatment condition in the matched groups. Subsequently, full matching cannot be used with more complex designs, and investigators are forced to resort to suboptimal solutions in these situations. Two such designs are studies with more than two treatment conditions and when we require the matched groups to contain more than two units (e.g., for estimation of variances or heterogeneous effects). The method we introduce, *generalized full matching*, allows for complex designs with multiple treatments and arbitrary size restrictions.

Second, we present an efficient algorithm that can be used to derive near-optimal generalized full matchings. The algorithm is intended to complement existing matching algorithms. In particular, the most widely used algorithm for deriving full matchings (Hansen & Klopfer 2006) derives optimal solutions, but it is restricted to the traditional formulation and cannot be used to derive the generalized full matchings that a more complex design requires. Furthermore, due to computational constraints, the algorithm by Hansen & Klopfer cannot be used with large samples. Our algorithm solves both of these issues; it derives well-performing generalized full matchings and does so quickly even in large samples. We prove that the produced matchings are guaranteed to be within a constant factor of the optimal solutions and that the algorithm terminates in linearithmic time on average. This makes full matching possible in the cases where it previously was infeasible.

Simulations show that the algorithm typically performs close to on par with the optimal full matching algorithm in cases where the optimal algorithm can be use. Our algorithm

is, however, several orders of magnitude faster than the existing solutions; a sample with one million units can be matched within a few seconds on an ordinary laptop computer.

2 The matching problem

We should suspect that systematic differences exist between treatment groups when assignment is not randomized. If we, for example, are interested in the effect of aspirin on headaches, we face the problem that people with more severe headaches are more likely to take aspirin; aspirin-takers and non-takers differ in more dimensions than just their drug consumption.

Matching methods make treatment groups comparable by down-weighting, implicitly or explicitly, units that have a treatment assignment that is overrepresented given their characteristics. That is, units assigned to a treatment condition that is uncommon locally in the covariate space are given a larger weight than units with a common condition. In our example, people with severe headaches who still do not take aspirin—an uncommon type—will be given large weights, while aspirin-takers with severe headaches are given smaller weights due to their commonness. If the weighting is successful, the distribution of initial headache severity will be close to identical between the re-weighted aspirin-takers and non-takers, and any remaining difference can be attributed to the difference in drug consumption.

We might be able to perfectly equalize the covariate distribution between the treatment groups when the confounders are few and coarse. That is, we can construct an *exact matching*. This is achieved by stratifying the sample based on the confounders so that all units within a matched group are identical. In more realistic settings—with high dimensional confounders—exact matchings are not feasible. Whenever a sample consists of units with a unique set of characteristics (e.g., when we have continuous covariates), some strata will

contain only a single unit, and we cannot perfectly balance the treatment groups. We could discard the unique units and focus only on those that have identical matches. However, that practice often results in an empty sample as it is common that all units are unique. Even if exact matches exist, the effect among the units not discarded might not be the same as the effect of interest. Generally, a more attractive solution is to accept less than perfect matchings, an *approximate matching*.

2.1 Approximate matching methods

Exact and approximate matchings are similar in that they both partition a sample into matched groups. The requirement that all units in a group are identical is, however, relaxed with approximate matching. We still strive to make the groups as similar as possible, but we accept less than perfect groups. By allowing heterogeneity, there is no longer an unambiguous grouping of the units; the matching problem becomes an optimization problem. Given some constraints on the structure on the matched groups (e.g., that each group must contain one unit of each treatment condition), we try to find the best matching among a set of admissible partitions with respect to some measure of quality. Approximate matching methods differ mainly along three dimensions: (1) the choice of objective function: how we measure the quality of matches; (2) the imposed constraints: what type of matching structures we are looking for; and (3) the algorithm we use to derive the matching.

Match quality—the objective of the optimization—is typically assessed through pairwise distances between units. For any two units in the sample, we use a metric to assign a distance to the pair indicating how similar they are, where a greater distance indicates less similarity. If a matched group contains units at close distances to each other, the group is considered to be of high quality. Common choices of such measures are the absolute difference between propensity scores (Rosenbaum & Rubin 1983) and Euclidean and Mahalanobis distances on the covariate space (Cochran & Rubin 1973). A measure

of the overall quality is achieved by aggregating within-group distances. Examples of such aggregation functions are the maximum, sum or average distance. In this paper, we are agnostic to the choice of dissimilarity measure: any metric can be used. However, as detailed below, we focus on the case where within-group distances are aggregated by their maximum.

Most studies require some structure on the matched groups. For example, we often estimate the treatment effects by calculating differences in mean outcomes between treatment conditions within matched groups. This requires that each treatment condition is represented in each group. The matching constraints are also important because they implicitly resolve the bias-variance trade-off that is inherent to matching methods. In particular, in order to equalize the covariate distributions between the treatment conditions, we want to form groups with units that are very similar. However, taken to its extreme (i.e., without any structural constraints), this approach leads to skewed weights or even that large parts of the sample are discarded from the analysis. While the very point of matching is to re-weight the sample, too much weight variation will lead to inefficient use of the available information. In other words, allowing the weights to vary greatly could reduce the bias, but it could also lead to an increased variance. For the remainder of this section, we will focus on how the most common matching methods resolve this trade-off through the constraints they impose.

The archetypical matching method is *nearest neighbor matching without replacement* (also called 1:1-matching). This method requires that each matched group contains exactly one treated unit and exactly one control unit. Units without matches are discarded. The groups can be formed optimally or heuristically. The former performs better than the latter, but in most cases the difference is not noteworthy compared to the effect of the matching constraints. An illustration of the method is shown in Panel A in Figure 1.

While 1:1-matching without replacement typically yields fairly high-quality matches,

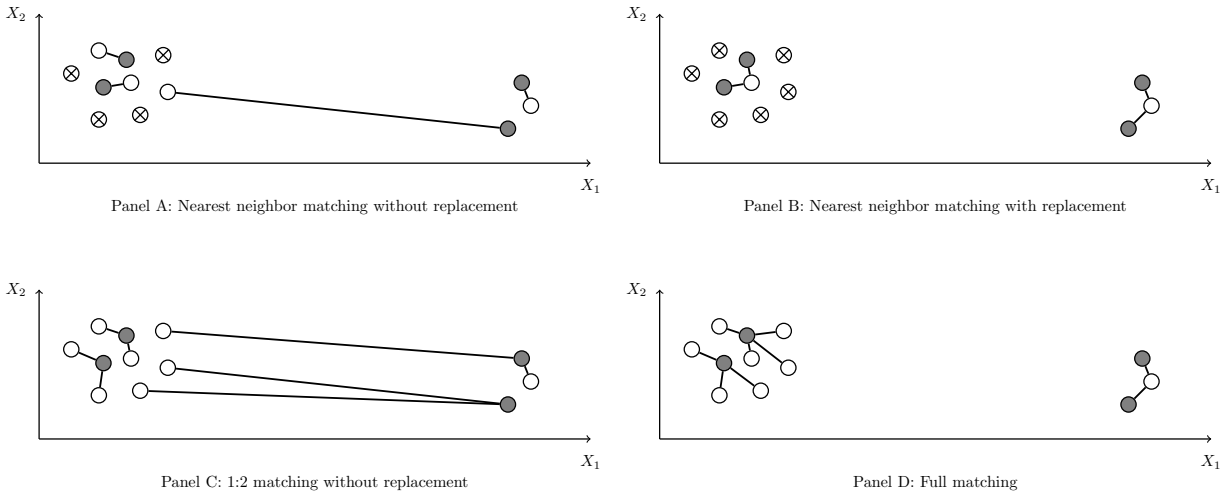


Figure 1: Illustration of different matching methods. The sample consists of 12 units belonging to either of two treatment conditions. We observe two covariates, X_1 and X_2, for each unit. The units are presented as circles in the covariate plane. The color of the circles indicates the units' treatment assignment. Edges indicate matched groups, and crossed out circles indicate discarded units. Well-performing matchings avoid discarded units and long edges (as they correspond to matches between dissimilar units).

the match structure entails two disadvantages. First, as matches cannot be reused, good matches will often run out in sparsely populated regions of the covariate space. This can be seen in the right cluster in in Panel A where there are two treated units but only a single control. As the constraints only allow one of the treated units to be matched with the control, the other unit is forced to seek a match in the left cluster—a match of low quality. Second, if there are several potential matches with about the same quality, 1:1-matching will only use one of them. As a consequence, large parts of the sample will potentially be discarded. This problem can also be seen in Panel A. In the left cluster, where controls are abundant, more than half of them are discarded as there are not enough treated units to match with.

Both problems have simple solutions. Unfortunately, each solution exacerbates the problem it does not solve. In particular, to improve match quality we can use nearest

neighbor matching *with replacement*. This method does not require the matched groups consist of exactly one treated and one control. Instead, several treated units can be matched to the same control (i.e., it only requires that each group consists of exactly one control). The method is illustrated in Panel B in Figure 1. As we see, matches in the sparser right cluster are of much higher quality. Unlike the previous method, both the treated units are allowed to be matched with the single control unit, and we avoid the low-quality match. However, looking at the left cluster, we see that even more controls are discarded. It turns out that a single control unit is a very good match for both of the treated units. All the other controls in the left cluster will, thus, be left without a match and are discarded. While matching with replacement typically leads to superior match quality, it does so by discarding many potentially useful matches.

We can use *1:k-matching* to reduce the number of discarded units and thereby resolve the second problem. Instead of requiring that each treated unit is matched with one control unit, this method requires that the unit is matched with k controls. The constraint increases the demand for matches, and fewer units will be unmatched as a result. In densely populated regions, this leads to a more efficient use of the information because fewer units are discarded. However, in sparse regions, there is typically already a shortage of good matches. Requiring additional matches will exacerbate this shortage, and match quality will suffer. In fact, match quality decreases with k, and at the extreme where no units are discarded, 1:k-matching performs no better than the unadjusted sample. This effect is illustrated in Panel C in Figure 1. Here, we require that each treated unit is matched to two controls; we, thereby, ensure that no controls are discarded from the analysis. However, as a consequence, the treated units in the right cluster must seek even more matches in the left cluster.

The ideal would be to combine the desirable properties from both of these methods. *Full matching* (Rosenbaum 1991, Hansen 2004) does exactly this. It adapts the matching

structure to the data at hand. That is, it works like matching with replacement in sparse regions of the covariate space and like 1:k-matching in denser regions. In particular, full matching imposes three constraints. First, all units must be assigned to a matched group: we do not discard units. Second, all groups must contain at least one unit of both treatment conditions. Finally, each matched group must contain *exactly* one treated (i.e., similar to 1:k-matching) or *exactly* one control unit (i.e., similar to matching with replacement). This set of constraints admit flexibility in the composition of the matched groups, which allows the matching to include all units without sacrificing match quality. In fact, among matchings that do not discard units, full matching will yield the best match quality for all common objective functions (Rosenbaum 1991). Panel D in Figure 1 illustrates how the sample in our example would be matched using the method. We see that the two treated units in the left cluster divide the controls between them so that all units belong to a matched group—similar to 1:k-matching in that subsample. In the right cluster, however, the two treated units share the single close-by control, i.e., the matching we would expect from matching with replacement in this subsample.

The traditional formulation of full matching requires a very particular design. It can only be used with studies that have two treatment conditions, where the investigator accepts matched groups with only two units. While most observational studies conform to this design, many do not. In a more complex setting—for example, when there are several treatment conditions or when larger groups are needed for heterogeneous treatment effect analysis or variance estimation—the traditional formulation is unsatisfactory. Currently, such studies must use cruder matching methods which might introduce bias and increase variance. As rigorously introduced in the next section, we introduce a generalization of traditional full matching that can be used in these more complicated settings.

3 Generalized full matching

Consider a sample, $\mathbf{U}$, consisting of n units indexed by $1, 2, \cdots, n$. Before the study, nature has assigned the units to one of k treatment conditions indexed by $1, 2, \cdots, k$. Let W_i denote the condition that unit i is assigned to. We construct a set for each condition, $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_k$, collecting the units assigned to the corresponding treatment:

$$\mathbf{w}_j = \{i \in \mathbf{U} : W_i = j\}.$$

A matched group, $\mathbf{m}$, is a non-empty set of unit indices. A matching, $\mathbf{M}$, is a set of disjoint matched groups: $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots\}$. We note that the term "matching" has a different meaning in the computer science literature, where it typically is used to refer to an independent edge set in a graph. We will follow the causal inference literature and use term as defined here.

A matching problem is defined by a set of constraints and an objective function. The constraints describe a collection of admissible matchings, $\mathcal{M}$, and the objective function maps from the admissible matchings to a real-valued measure of match quality: $L : \mathcal{M} \to \mathbb{R}$.

Definition 1. An *optimal matching* $\mathbf{M}^*$ is an admissible matching that minimizes the matching objective:

$$L(\mathbf{M}^*) = \min_{\mathbf{M} \in \mathcal{M}} L(\mathbf{M}).$$

Generalized full matching imposes the constraint that each unit is assigned to exactly one group. The investigator can also impose constraints on the structure of the matched groups. In particular, for each treatment condition i, one can require that each matched group contains at least c_i units assigned to the corresponding condition. One can also require that each group contains at least t units in total (irrespectively of treatment assignment).

Definition 2. An *admissible generalized full matching* for constraints $\mathcal{C} = (c_1, \cdots, c_k, t)$ is a matching $\mathbf{M}$ that satisfies:

1. (Spanning) $\bigcup_{\mathbf{m}\in\mathbf{M}} \mathbf{m} = \mathbf{U}$,

2. (Disjoint) $\forall \mathbf{m}, \mathbf{m}' \in \mathbf{M}, \mathbf{m} \neq \mathbf{m}' \Rightarrow \mathbf{m} \cap \mathbf{m}' = \emptyset$,

3. (Treatment constraints) $\forall \mathbf{m} \in \mathbf{M}, \forall i \in \{1, \cdots, k\}, |\mathbf{m} \cap \mathbf{w}_i| \geq c_i$,

4. (Overall constraint) $\forall \mathbf{m} \in \mathbf{M}, |\mathbf{m}| \geq t$.

Let $\mathcal{M}_\mathcal{C}$ collect all admissible generalized full matchings for constraints $\mathcal{C}$.

When we restrict the matching constraints so to require only one unit of each treatment condition in the matched groups, we recover the traditional full matching definition for studies with an arbitrary number of treatment conditions.

Definition 3. A *traditional full matching* in a study with k treatment conditions is a generalized full matching with the matching constraints $\mathcal{C} = (1, 1, \cdots, 1, k)$.

As an example, consider a study with three treatment conditions. The constraint $\mathcal{C} = (2, 2, 4, 10)$ would restrict the admissible matchings to those where each matched group contains two units of the first and second treatment conditions, four units of the third condition and ten units in total.

Our definition of full matching differs from the original definition in Rosenbaum (1991). In particular, as detailed in Section 2.1, for studies with two treatment conditions ($k = 2$), the conventional definition requires, in addition the conditions in Definition 2, that we impose the matching constraints $\mathcal{C} = (1, 1, 2)$ and that each matched group contains exactly one treated unit or exactly one control unit:

$$\forall \mathbf{m} \in \mathbf{M}, |\mathbf{m} \cap \mathbf{w}_1| = 1 \vee |\mathbf{m} \cap \mathbf{w}_2| = 1.$$

This convention prevents us from using the method in a general setting. Our generalization builds on an implication of Proposition 1 in Rosenbaum (1991). The proposition shows

that when $k = 2$, the optimal generalized full matching with constraints $\mathcal{C} = (1, 1, 2)$ is by necessity a full matching according to the original definition. As a result, we can disregard the additional conditions imposed by Rosenbaum (1991) and equivalently define full matchings as the optimal solution to the broader class of matching problems given by Definition 2. This arrangement is readily generalizable to more complex settings, and it is the one we present above.

3.1 Near-optimal matchings

As the number of units in a matching problem grows large, it may become intractable to derive optimal solutions. In fact, generalized full matching is an **NP**-hard problem (Higgins et al. 2017). However, as we will see in the next section, the matching problem becomes tractable if we allow for approximately optimal generalized full matchings. That is, matchings that are guaranteed to be within some factor of the optimal solution.

Definition 4. An *α-approximate matching* $\mathbf{M}^\dagger$ is an admissible matching that is within a factor of α of an optimal matching:

$$L(\mathbf{M}^\dagger) \leq \min_{\mathbf{M} \in \mathcal{M}} \alpha L(\mathbf{M}).$$

4 A generalized full matching algorithm

We introduce a computationally efficient algorithm that constructs near-optimal generalized full matchings. In particular, it produces a 4-approximate generalized full matching in linearithmic time on average. The algorithm is an extension of the blocking algorithm described in Higgins et al. (2016). Blocking is an experimental design where similar units are grouped together into blocks and treatment is assigned within the blocks. Matching and blocking are similar in that they try to balance covariate distributions. However, because treatment has not yet assigned in blocking problems, such algorithms only need to

consider overall size constraints. To solve matching problems, we must be able to impose more detailed structural constraints.

4.1 Matching objective

Consistent with the existing matching literature, we will restrict our focus to objective functions based on summaries of pairwise distances between units. Let $\mathrm{d} : \mathbf{U} \times \mathbf{U} \to \mathbb{R}^+$ be a distance metric capturing similarity between any pair of units, where lower values indicate greater similarity. A distance metric is any function that satisfies:

1. (Non-negativity) $\forall i, j \in \mathbf{U}, \mathrm{d}(i, j) \geq 0$,
2. (Self-similarity) $\forall i \in \mathbf{U}, \mathrm{d}(i, i) = 0$,
3. (Symmetry) $\forall i, j \in \mathbf{U}, \mathrm{d}(i, j) = \mathrm{d}(j, i)$,
4. (Triangle inequality) $\forall i, j, \ell \in \mathbf{U}, \mathrm{d}(i, j) \leq \mathrm{d}(i, \ell) + \mathrm{d}(\ell, j)$.

All commonly-used similarity measures, such as absolute differences between propensity scores and Euclidean or Mahalanobis distances on pretreatment covariates, satisfy these conditions.

The objective function used in traditional full matching is either a weighted mean of within-group distances between treated and control units (Rosenbaum 1991) or the sum of such distances (Hansen 2004). We will depart from this tradition in two ways. First, we will focus on the maximum within-group distance, a *bottleneck* objective function. The main motivation for this shift is that the bottleneck objective facilitates the computationally efficient algorithm we present below. However, while we only prove approximate optimality with respect to the maximum distance, we expect that the algorithm performs well also on the mean distance. Apart from computational considerations, minimizing the

maximum distance has the advantage of avoiding devastatingly poor matches that might be undetected by, for example, the mean distance.

The second departure is that we consider all within-group distances, not only those between units assigned to different treatment conditions as in the existing literature. In the traditional full matching setting, there is little difference between the two objectives. However, with more treatment conditions and larger matched groups, the traditional objective risks ignoring important within-group distances. To maintain consistency with the previous literature, we will also investigate the bottleneck objective that only includes within-group distances between units assigned to different conditions. We show that when using traditional full matching constraints, our algorithm provides the same optimality guarantees for both aggregation functions. In symbols, the objectives we consider are:

$$L^{Max}(\mathbf{M}) = \max_{\mathbf{m}\in\mathbf{M}} \max\{\mathrm{d}(i,j) : i,j \in \mathbf{m}\}, \tag{1}$$

$$L^{Max}_{tc}(\mathbf{M}) = \max_{\mathbf{m}\in\mathbf{M}} \max\{\mathrm{d}(i,j) : i,j \in \mathbf{m} \wedge W_i \neq W_j\}. \tag{2}$$

4.2 Preliminaries

The description of the algorithm and the proofs of its properties rely heavily on graph theory. The most central concepts are defined in this section. Refer to the supplementary materials for a brief overview of additional concepts and terminology.

Definition 5. A *closed neighborhood* of vertex i in digraph $G = (V, E)$ is a subset of vertices $\mathrm{N}[i] \subset V$ consisting of i itself and all vertices $j \in V$ with an arc from i to j:

$$\mathrm{N}[i] = \{j \in V : (i,j) \in E\} \cup \{i\}. \tag{3}$$

Definition 6. An **IJ**-digraph, denoted $G(\mathbf{I} \to \mathbf{J})$, is a graph $G = (\mathbf{I} \cup \mathbf{J}, E_{\mathbf{IJ}})$ with arcs drawn from all vertices in $\mathbf{I}$ to all vertices in $\mathbf{J}$. That is:

$$E_{\mathbf{IJ}} = \{(i,j) : i \in \mathbf{I} \wedge j \in \mathbf{J}\},$$

Self-loops (i.e., arcs from i to i) are drawn for all vertices $i \in \mathbf{I} \cap \mathbf{J}$.

Definition 7. A *κ-nearest neighbor digraph* of $G = (V, E)$ is a spanning subgraph of G where an arc $(i, j) \in E$ is in the nearest neighbor digraph if j is one of the κ closest vertices to i according to $\mathrm{d}(i, j)$ among all its outward-pointing arcs. That is, for each $i \in V$, sort $(i, j) \in E$ by $\mathrm{d}(i, j)$ and keep the κ smallest items. If ties exist, give priority to self-loops and otherwise resolve them arbitrarily. We denote κ-nearest neighbor graphs as $\mathrm{NN}(\kappa, G)$.

4.3 The algorithm

The following steps describe how a matching is constructed given a sample $\mathbf{U}$, matching constraints $\mathcal{C} = (c_1, \cdots, c_k, t)$ and distance metric $\mathrm{d}(i, j)$. Figure 2 provides an illustration.

1. For each treatment condition $j \in \{1, 2, \cdots, k\}$, construct the c_j-nearest neighbor digraph of the $\mathbf{U}\mathbf{w}_j$-digraph. Construct the union of these graphs:

$$G_w = \mathrm{NN}(c_1, G(\mathbf{U} \to \mathbf{w}_1)) \cup \cdots \cup \mathrm{NN}(c_k, G(\mathbf{U} \to \mathbf{w}_k)).$$

2. Let $r = t - c_1 - \cdots - c_k$ be the number of units needed to satisfy the overall size constraint in excess of the treatment-specific constraints. Construct digraph G_r by drawing an arc from i to each of its r nearest neighbors (of any treatment status) given that this arc does not exist in G_w:

$$G_r = \mathrm{NN}(r, G(\mathbf{U} \to \mathbf{U}) - G_w),$$

   where $G(\mathbf{U} \to \mathbf{U})$ is the complete digraph over $\mathbf{U}$ and the graph difference $G(\mathbf{U} \to \mathbf{U}) - G_w$ removes all arcs in $G(\mathbf{U} \to \mathbf{U})$ that exist in G_w.

   We refer to the union $G_{\mathcal{C}} = G_w \cup G_r$ as the *$\mathcal{C}$-compatible nearest neighbor digraph.*

3. Find a set of vertices $\mathbf{S} \subset \mathbf{U}$, referred to as *seeds*, so that their closed neighborhoods in $G_{\mathcal{C}}$ are non-overlapping and adding any additional vertex to $\mathbf{S}$ would create some overlap. That is, $\mathbf{S}$ has the following two properties with respect to $G_{\mathcal{C}}$:

- (Independence) $\forall i, j \in \mathbf{S}, \mathrm{N}[i] \cap \mathrm{N}[j] = \emptyset$.
- (Maximality) $\forall j \notin \mathbf{S}, \exists i \in \mathbf{S}, \mathrm{N}[i] \cap \mathrm{N}[j] \neq \emptyset$.

4. Assign a label to each seed. Assign the same label to all vertices in the seed's neighborhood in $G_{\mathcal{C}}$. We refer to vertices that are labeled in this step as *labeled vertices*.

5. For each vertex i without a label, find its closed neighborhood $N[i]$ in $G_{\mathcal{C}}$ and assign it the same label as one of the labeled vertices in the neighborhood.

When the algorithm terminates, each vertex has been assigned a label. Vertices that share the same label form a matched group. The collection of labels thus forms a matching. Let $\mathbf{M}_{alg}$ denote this matching.

4.4 Properties

The algorithm and the matching it constructs have two important properties. First, $\mathbf{M}_{alg}$ is an admissible generalized full matching, and the maximum within-group distance in the matching is guaranteed to be less or equal to four times the maximum within-group distance in an optimal matching. In other words, $\mathbf{M}_{alg}$ is a 4-approximate generalized full matching. Second, the algorithm terminates quickly. In this section, we discuss the two properties in detail. Formal proofs are presented in the supplementary materials.

4.4.1 Optimality

Approximate optimality follows from two properties of the $\mathcal{C}$-compatible nearest neighbor digraph, as shown by the following two lemmas.

Lemma 8. *The closed neighborhood of each vertex in the $\mathcal{C}$-compatible nearest neighbor digraph satisfies the matching constraints $\mathcal{C} = (c_1, \cdots, c_k, t)$:*

$$\forall i \in V, \forall j \in \{1, \cdots, k\}, |\mathrm{N}[i] \cap \mathbf{w}_j| \geq c_j, \qquad \textit{and} \qquad \forall i \in V, |\mathrm{N}[i]| \geq t. \tag{4}$$

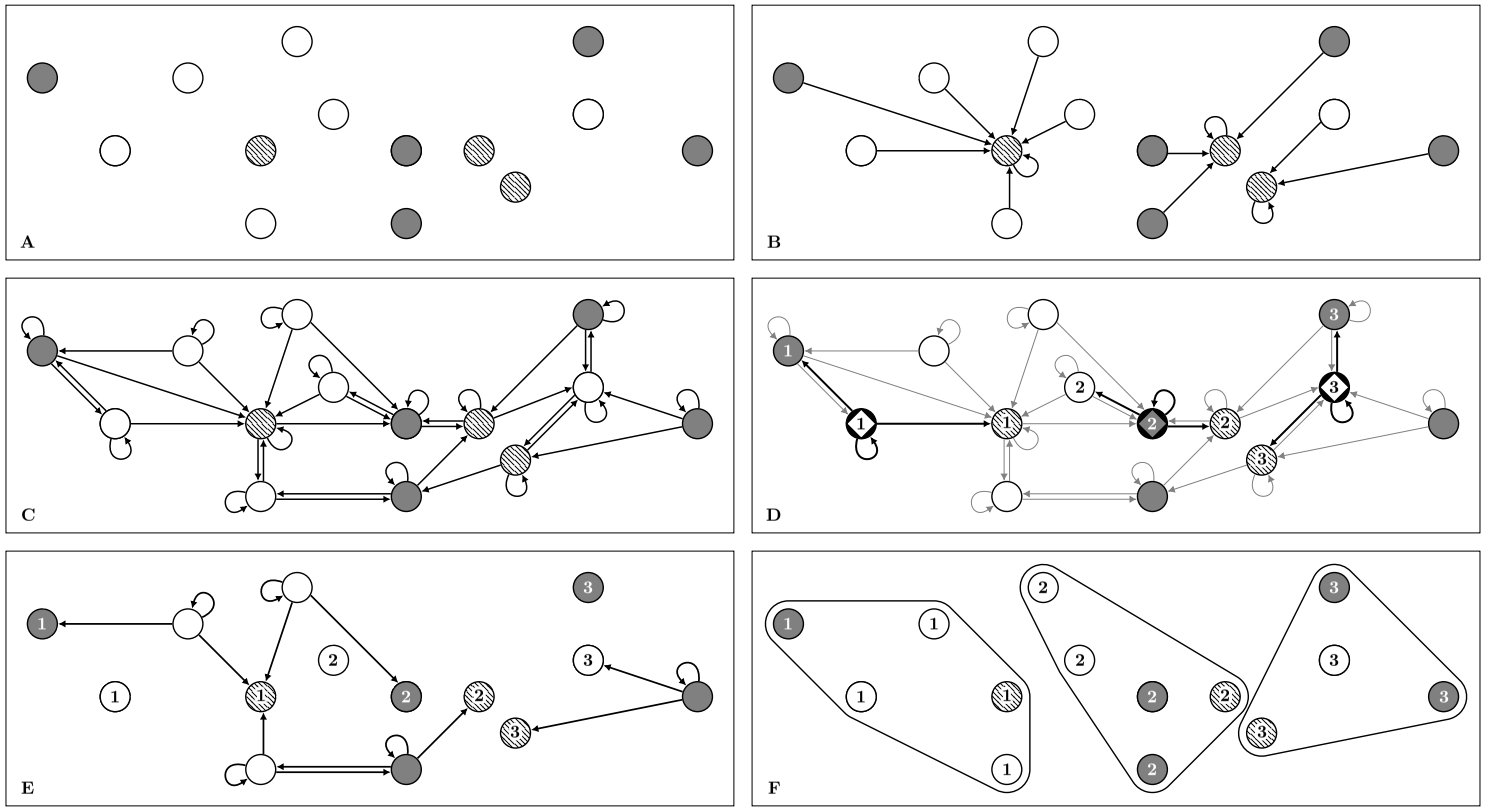


Figure 2: The generalized full matching algorithm. The sample consists of 14 units divided between three treatment conditions. We require that each matched group contains at least one unit of each treatment condition and at least three units in total. The distance metric is the Euclidean distance based on two covariates. **(A)** The units are presented as circles on the covariate plane. The treatment conditions are indicated by the units' color and pattern. **(B)** Step 1: One of the building blocks of G_w is shown, namely the nearest neighbor digraph between the whole sample and the patterned units: $\mathrm{NN}(1, G(\mathbf{U} \to \mathbf{w}_{\text{patterned}}))$. **(C)** Step 2: The $\mathcal{C}$-compatible nearest neighbor digraph, $G_{\mathcal{C}}$, is created. Note that all vertices in this graph are pointing to one vertex of each treatment condition and that there exists no other graph with shorter arcs that satisfy this condition. **(D)** Step 3: A set of seeds is found. Seeds are indicated with a diamond shape enclosed in their circles. The arcs pointing out from the seeds are highlighted. Note that no two seeds are pointing to a common unit. Step 4: Each seed and its neighbors are given a unique label as indicated by the numbers. **(E)** Step 5: Some units are still unlabeled. Each such unit is assigned a label that is represented in its neighborhood. All outward-pointing arcs from unlabeled units are shown in this panel. **(F)** The algorithm has terminated. Matched groups are formed by units sharing the same label as indicated by the enclosed groups of units.

Lemma 9. *The distance between any two vertices connected by an arc in the $\mathcal{C}$-compatible nearest neighbor digraph, $G_{\mathcal{C}} = (V, E_{\mathcal{C}})$, is less or equal to the maximum within-group distance in an optimal matching:*

$$\forall (i,j) \in E_{\mathcal{C}},\ \mathrm{d}(i,j) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} L^{Max}(\mathbf{M}). \tag{5}$$

Lemma 8 states that the $\mathcal{C}$-compatible nearest neighbor digraph encodes the matching constraints in the units' neighborhoods in $G_{\mathcal{C}}$. Admissibility of $\mathbf{M}_{alg}$ follows from that each matched group is a superset of a closed neighborhood. Since each neighborhood satisfy the matching constraints, so will each group.

Lemma 9 provides a connection between the arc weights in the $\mathcal{C}$-compatible nearest neighbor digraph and the maximum distance in the optimal solution. This follows from that a digraph compatible with $\mathcal{C}$ (in the sense that it satisfies Lemma 8) can be constructed as a subgraph of the cluster graph induced by an optimal matching. A $\mathcal{C}$-compatible digraph constructed in this way satisfies the property in Lemma 9 as we would not add any arcs not already in the optimal matching. As we show in the supplementary materials, $G_{\mathcal{C}}$ is the smallest digraph compatible with $\mathcal{C}$. Consequently, the distances in $G_{\mathcal{C}}$ must be bounded in the same way as in the subgraph induced by an optimal matching, and Lemma 9 holds.

Approximate optimality follows from that the triangle inequality of distance metrics. In particular, as shown in supplementary materials, two units in the same matched group is at most at a geodesic distance of four arcs in the $\mathcal{C}$-compatible nearest neighbor digraph. By the triangle inequality property, the worst case is when the five vertices connected by the arcs are lined up on a straight line in the metric space. In that case, the distance between the two end vertices is the sum of the distances of the intermediate arcs. Lemma 9 provides a bound for the intermediate arc distances and, thus, a bound for all within-group distances.

Theorem 10. $\mathbf{M}_{alg}$ *is a 4-approximate generalized full matching with respect to the match-*

ing constraint $\mathcal{C} = (c_1, \cdots, c_k, t)$ *and matching objective* L^{Max}*:*

$$\mathbf{M}_{alg} \in \mathcal{M}_{\mathcal{C}}, \qquad \textit{and} \qquad L^{Max}(\mathbf{M}_{alg}) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} 4L^{Max}(\mathbf{M}). \tag{6}$$

A similar strategy can be used to bound the L_{tc}^{Max} objective for the matching produced by the algorithm. In particular, for traditional full matching problems (see Definition 3), Lemma 9 holds also for the bottleneck function for distances between treated and controls.

Theorem 11. *For matching constraints* $\mathcal{C} = (1, \cdots, 1, k)$*,* $\mathbf{M}_{alg}$ *is a 4-approximate traditional full matching with respect to matching objective* L_{tc}^{Max}*:*

$$\mathbf{M}_{alg} \in \mathcal{M}_{\mathcal{C}}, \qquad \textit{and} \qquad L_{tc}^{Max}(\mathbf{M}_{alg}) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} 4L_{tc}^{Max}(\mathbf{M}). \tag{7}$$

4.4.2 Complexity

We assume that the number of treatment conditions (k) and the matching constraints are fixed in the following theorem. The time complexity is still polynomial if we let the number of treatment conditions and the matching constraints grow proportionally with n. However, as this rarely is the case in observational studies, we opt for the current set-up in order to simplify the derivations.

Theorem 12. *In the worst case, the generalized full matching algorithm terminates in polynomial time using linear memory.*

The algorithm can be divided into two parts. The first and more intricate part is the construction of the $\mathcal{C}$-compatible nearest neighbor digraph (i.e., the first two steps). This part essentially acts as a preprocessing step of the remainder of the algorithm. The idea is that $G_{\mathcal{C}}$ encodes sufficient information about the sample to ensure approximate optimality, but that it is sparse enough to ensure quick execution. Once $G_{\mathcal{C}}$ is constructed, the remaining steps are completed in linear time.

As discussed in the proof of Theorem 12 in the supplementary materials, the $\mathcal{C}$-compatible nearest neighbor digraph can be constructed by $n(k+1)$ calls to a nearest neighbor search function. For an arbitrary metric, each such call has a time complexity of $O(n \log n)$ and a space complexity of $O(n)$ (see, e.g., Knuth 1998). When k is fixed, it follows that the overall worst-case time complexity is $O(n^2 \log n)$. This is, thus, the complexity of the algorithm.

Specialized nearest neighbor search algorithms exist for the most commonly-used distance metrics. For example, when the metric is the Euclidean or Mahalanobis distance in the covariate space or absolute difference in propensity scores, large improvements can be expected by storing the data points in a kd-tree (Friedman et al. 1977). Given that the covariate distribution is not too skewed, each search can then be completed in $O(\log n)$ time on average. Using this approach, the overall average time complexity would be reduced to $O(n \log n)$. However, these data structures do not scale well with the dimensionality of the metric space. Feasible approaches in such cases include reducing the dimensionality prior to matching (e.g., by matching on estimated propensity scores, Rosenbaum & Rubin 1983), repeated matching in random low-dimensional projections of the covariate space (Li et al. 2016) and using approximate nearest neighbor search algorithms (Arya et al. 1998).

4.5 Extensions

The algorithm admits several extensions and refinements. First, the set of seeds derived in the third step of the algorithm is not unique. The properties discussed above hold for any set of seeds, but, heuristically, the performance of the matching depends on the units we pick. A valid set of seeds are equivalent to a maximal independent vertex set in the graph described by the adjacency matrix $\mathbf{A}\mathbf{A}' + \mathbf{A} + \mathbf{A}'$ where $\mathbf{A}$ is the adjacency matrix of $G_{\mathcal{C}}$. We expect improvements if a larger maximal independent set is used as seeds.

Second, in the fifth step of the algorithm, unassigned vertices are assigned to groups based on the $\mathcal{C}$-compatible nearest neighbor digraph. However, as all matching constraints

have already been fulfilled in the fourth step, the restrictions encoded in $G_{\mathcal{C}}$ are no longer necessary. By restricting the matches to arcs in $G_{\mathcal{C}}$, we might miss matched groups that are closer to the unassigned units. We could improve quality by searching for the closest labeled vertex among all vertices.

Third, it is sometime beneficial to relax the restriction that all units must be assigned to a group. For example, if some regions of the covariate space is sparse with respect to a treatment condition, we could be forced to construct very dissimilar groups in order to avoid discarding units. A common way to avoid this is to apply a *caliper*. That is, we restrict the maximum allowable distance within any matched group to some value. Units that cannot be assigned a group without violating the caliper is discarded. In our algorithm, such a caliper can be imposed in the construction of the $G_{\mathcal{C}}$. In particular, by restricting the length of the arcs in $G_{\mathcal{C}}$, we implicitly restrict the maximum allowable distance in the resulting matching. If the second refinement is implemented, we can impose a caliper (perhaps of different magnitude) also when assigning units in the fifth step of the algorithm.

Fourth, we are occasionally interested in estimating treatment effects only for some subpopulation. For example, it is common to estimate the average treatment effect only for treated units (ATT). To estimate such an effect, we only need that units from the subpopulation of interest are assigned to matched groups; other units can be left unassigned. In most cases, it is still beneficial to assign all units to groups as we then exploit all information in the sample. However, if there are sparse regions in the covariate space, this procedure might lead to poor match quality. The algorithm allows us to focus the matching to a certain set of units. In particular, in the first two steps, by substituting $\mathbf{U}$ with some set $\mathbf{S}$, we ensure that all units in $\mathbf{S}$ are assigned to matched groups. Units not in $\mathbf{S}$ are only assigned to groups insofar as they are needed to satisfy the matching constraints. The unassigned units might later be assigned to groups in the fifth step (preferably with a caliper to avoid affecting match quality).

5 Simulation study

We present the results from a small simulation study of the performance of an implementation of the presented algorithm compared to implementations of commonly used matching methods. In particular, we investigate the algorithm with and without the first two refinements discussed in section 4.5, and compare it with 1:1-matching, 1:1-matching with replacement, 1:2-matching and traditional full matching. For 1:1-matching and 1:2-matching we include both an implementation that derive optimal solutions and a greedy implementation. The traditional and generalized full matching methods use the same matching constraints, namely at least one treated and control unit in each group.

We focus on a simple setting where each unit has two covariates distributed uniformly on a plane:

$$X_{1i}, X_{2i} \sim \mathcal{U}(-1, 1). \tag{8}$$

The units are randomly assigned to one of two treatment conditions, $W_i \in \{0, 1\}$, based on a logistic propensity score that maps from the covariates to treatment probabilities:

$$\Pr(W_i = 1 | X_{1i}, X_{2i}) = \text{logistic}\left[\frac{(X_{1i}+1)^2 + (X_{2i}+1)^2 - 5}{2}\right]. \tag{9}$$

Units with higher covariate values are thus more likely to be treated. The treatment probability ranges from 7.6% at $(-1, -1)$ to 81.8% at (1,1). The unconditional treatment probability is 26.5%. The outcome is given by:

$$Y_i = (X_{1i} - 1)^2 + (X_{2i} - 1)^2 + \varepsilon, \tag{10}$$

where ε is standard normal. The outcome does not depend on treatment; the treatment effect is constant at zero.

We use a version of the estimator discussed by Abadie & Imbens (2006) and Imbens & Rubin (2015) to estimate the average treatment effect for the subpopulation of treated units (ATT). We derive the mean outcome difference between treated and control units

within each matched group and then average the differences over all groups weighted by the number of treated units:

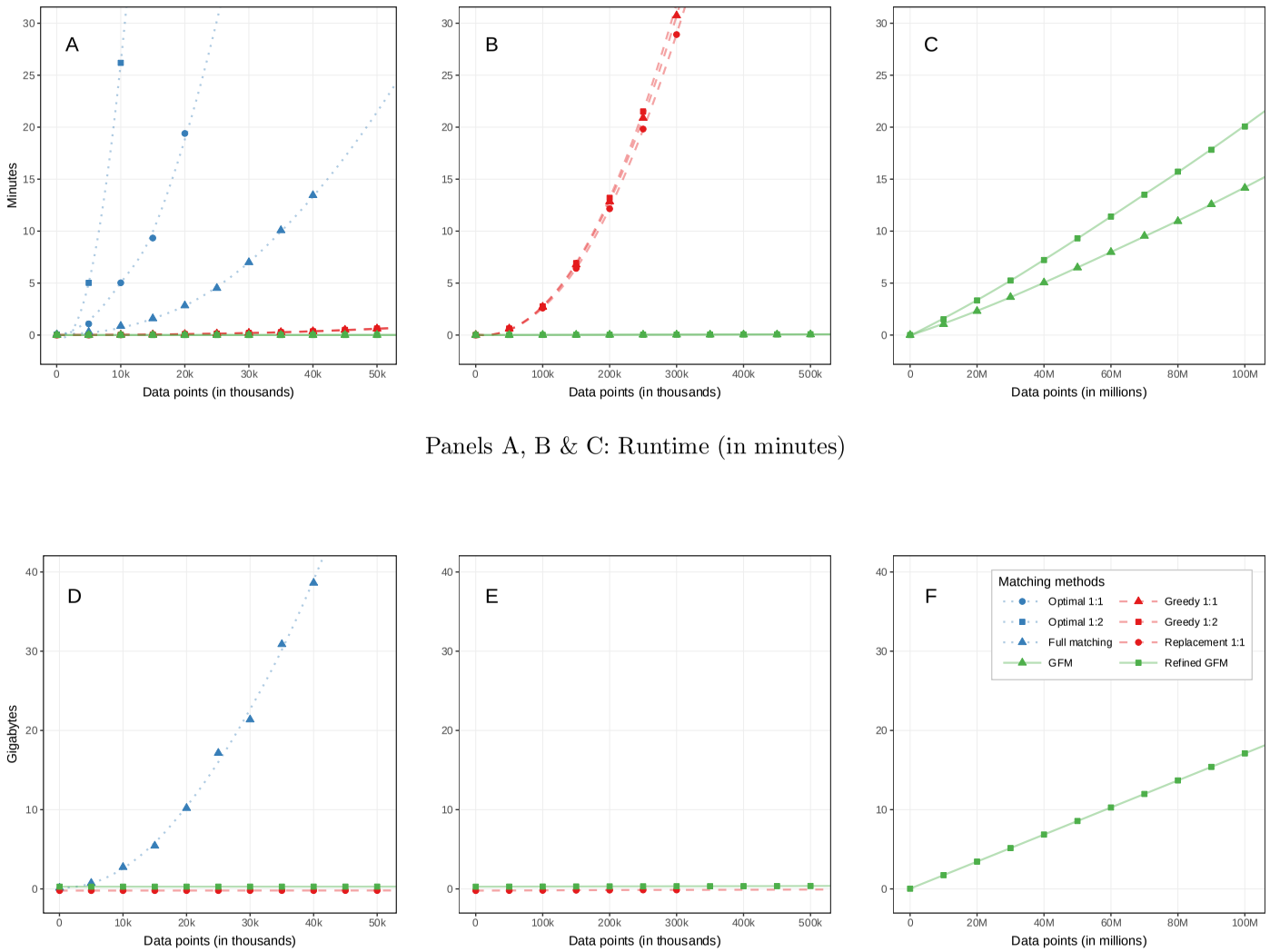
$$\widehat{\tau_{\text{ATT}}}(\mathbf{M}) = \sum_{\mathbf{m} \in \mathbf{M}} \frac{|\mathbf{w}_1 \cap \mathbf{m}|}{|\mathbf{w}_1|} \left[\frac{\sum_{i \in \mathbf{m}} W_i Y_i}{|\mathbf{w}_1 \cap \mathbf{m}|} - \frac{\sum_{i \in \mathbf{m}} (1 - W_i) Y_i}{|\mathbf{w}_0 \cap \mathbf{m}|} \right]. \tag{11}$$

Other approaches to estimation are discussed by Stuart (2010). In particular, matching facilitates permutation-based inference which many investigators find useful (Rosenbaum 2002, 2010).

The Savio cluster at UC Berkeley was used to run the simulations using version 3.3.2 of R (R Core Team 2016). Each simulation round was assigned a single CPU core, largely reflecting the performance of a modern computer. To derive generalized full matchings, we used a development version of the `scclust` R package. Optimal 1:1-matchings, optimal 1:2-matchings and traditional full matchings were derived using version 0.9-7 of the `optmatch` R package (Hansen & Klopfer 2006). Finally, we used version 4.9-2 of the `Matching` R package (Sekhon 2011) to derive greedy matchings and matchings with replacement. We used the Euclidean distances on the covariate plane as similarity measure in all cases. The `scclust` package uses the maximum within-group distance as its objective function, as discussed above. The `Matching` and `optmatch` packages use the sum of within-group distances between treated and control units as objective. All functionality beside the matching functions (e.g., estimators) were implemented independently and are common for all matching methods.

5.1 Run time and memory

We matched 1,000 randomly generated samples with each matching method for sample sizes ranging from 100 units to 100 million units. Figure 3 presents the computational resources used by each implementation as a function of sample size (see Table S5 in the supplementary materials for more details). Average runtimes are presented in the first



Panels A, B & C: Runtime (in minutes)

Panels D, E & F: Memory (in gigabytes)

Figure 3: Runtime and memory by matching method. Marker symbols are actual simulation results, and connecting lines are interpolations. Results are presented with different scales due to the large differences in performance. All methods are included for sample sizes up to 50,000 units. Results for sample sizes up to 500,000 units are presented for methods in the `Matching` and `scclust` packages, and only the latter package is included for sample sizes up to 100 million units. Memory use was identical for methods from the same package, so we present results for only one implementation from each package. Each measure is based on 1,000 simulation rounds; simulation errors are negligible.

three panels, and memory usage is presented in the subsequent three panels. The results are split into several panels with different scales due to the large differences in performance.

Panels A and D present results for samples with up to 50,000 units. For small sample sizes, all implementations perform well. However, as the sample grows, the `optmatch` package struggles both with respect to runtime and memory. Already with 10,000 units, optimal 1:1-matching takes more than 25 minutes to terminate on average, and with sample sizes over 40,000 units, the package allocates more than 40 gigabytes of memory. The implementations in `optmatch` are the only ones that derives optimal solutions, but this comes at a large computational cost; the other packages terminate close to instantly with negligible memory use for these sample sizes.[1]

Results for samples with up to 500,000 units are presented in Panels B and E. Implementations from the `scclust` package still terminates virtually instantly with negligible memory use. The `Matching` package terminates quickly for samples with less than 200,000 units, but its runtime increases after that. More than 30 minutes are required for samples larger than about 300,000 units. Memory use is, however, still negligible. The generalized full matching algorithm uses a version of matching with replacement as a subroutine in its first two steps. Subsequently, it is possible to construct an implementation of matching with replacement with strictly lower computational requirements than generalized full matching. The `Matching` package implements additional functionality beyond the standard formulation studied here, which explains why its implementation is slower. Matching without replacement is, however, not necessarily faster than our algorithm since the search index needs to be updated after each match in that case.

[1]The `optmatch` package translates matching problems into network flow problems. This admits optimal solutions to 1:1-, 1:k- and full matching problems using a single algorithm. However, in the 1:1-matching case, the matching problem is a minimum cost independent edge set problem. We expect a sizable decrease in runtime with an implementation specifically targeted to derive 1:1-matchings using, e.g., the Hungarian algorithm (Kuhn 1955). To the best of our knowledge, no such implementation exists for R.

Finally, Panels C and F present samples with up to 100 million units. Both implementations of the generalized full matching algorithm terminate quickly for sample sizes less than 20 million units. With a sample of 100 million units, the implementation without refinements terminates within 15 minutes on average, while the version with refinements adds about 5 minutes to the runtime. Memory use increases at a slow, linear rate. At 20 million units, it uses about 4 gigabytes of memory on average. At 100 million units, it requires slightly more than 17 gigabytes. With the `scclust` package, an ordinary laptop computer is able to derive generalized full matchings for samples up to about 20 million units. It should be feasible to match samples up to 100 million units on a modern workstation.

5.2 Match quality

To investigate the quality of the matchings produced by the implementations, we matched 10,000 randomly generated samples containing 1,000 and 10,000 units. The results differ little with the sample size, so we will focus on the samples with 10,000 units here. Please see the supplementary materials for results for samples with 1,000 units.

5.2.1 Distances

We investigate five different functions aggregating within-group distances:

$$L^{Max}(\mathbf{M}) = \max_{\mathbf{m}\in\mathbf{M}} \max\{\mathrm{d}(i,j) : i,j \in \mathbf{m}\},$$

$$L_{tc}^{Max}(\mathbf{M}) = \max_{\mathbf{m}\in\mathbf{M}} \max\{\mathrm{d}(i,j) : i,j \in \mathbf{m} \wedge W_i \neq W_j\},$$

$$L^{Mean}(\mathbf{M}) = \sum_{\mathbf{m}\in\mathbf{M}} \frac{|\mathbf{w}_1 \cap \mathbf{m}|}{|\mathbf{w}_1|} \,\mathrm{mean}\{\mathrm{d}(i,j) : i,j \in \mathbf{m} \wedge i \neq j\},$$

$$L_{tc}^{Mean}(\mathbf{M}) = \sum_{\mathbf{m}\in\mathbf{M}} \frac{|\mathbf{w}_1 \cap \mathbf{m}|}{|\mathbf{w}_1|} \,\mathrm{mean}\{\mathrm{d}(i,j) : i,j \in \mathbf{m} \wedge W_i \neq W_j\},$$

$$L_{tc}^{Sum}(\mathbf{M}) = \sum_{\mathbf{m}\in\mathbf{M}} \sum \{\mathrm{d}(i,j) : i,j \in \mathbf{m} \wedge W_i \neq W_j\}.$$

L^{Max} is the maximum within-group distance between any two units, and L_{tc}^{Max} is the maximum distance between treated and control units. They are the objectives discussed in Section 4.1 and are the ones used by the `scclust` package. L_{tc}^{Mean} is the average within-group distance between treated and control units weighted by the number treated units in the groups. It is the objective function discussed by Rosenbaum (1991) when he introduced full matching. As Rosenbaum notes, this objective is neutral in the sense that the size of the matched groups matters only insofar as it affects the within-group distances. To contrast with L^{Max}, we include L^{Mean}: a version of the mean distance objective that also considers within-group distances between units assigned to the same treatment condition.

Finally, L_{tc}^{Sum} is the sum of within-group distances between treated and control units. With the terminology of Rosenbaum (1991), this function favors small subclasses and is, thus, not neutral. As a consequence, if we were to use L_{tc}^{Sum} as our objective, we would accept matchings with worse balance if the matched groups were sufficiently smaller. When the matching structure is fixed (as with 1:1- and 1:k-matching without replacement), L_{tc}^{Sum} is proportional to L_{tc}^{Mean} and, thus, identical for practical purposes. Both the `optmatch` and `Matching` packages use the sum as their objective.

Panel A in Table 1 presents the distance measures for the different methods. As distances have no natural scale, we normalize the results by traditional full matching. We see that 1:1-matching with replacement greatly outperforms the other methods, especially on L_{tc}^{Sum} which is its objective function. This is exactly what we expect from the discussion in Section 2.1; when we allow matching with replacement, we can avoid bad matches by discarding inconvenient units. The implementations of both traditional and generalized full matching perform largely identically, with a slight advantage to `optmatch` on the L_{tc}^{Sum} measure. 1:1- and 1:2-matching without replacement performs considerably worse than the other methods, in particular on the measures they do not use as their objective. Predic-

tively, the optimal implementations produce shorter distances than the greedy versions, but the differences are small.

Comparisons in aggregated distances between methods that impose different matching constraints can be awkward, as the methods solve different types of matching problems. For example, 1:2-matching will necessarily lead to larger distances than 1:1-matching, but the former can be preferable if, for example, we are interested in ATT and control units vastly outnumbers treated units. Comparisons between methods with identical matching constraints are, however, always informative.

5.2.2 Group structure

Panel B in Table 1 presents measures of the group structure for the different matching methods. The first measure is the average size of the matched groups. 1:1- and 1:2-matching without replacement have a fixed group size of either two or three units. The group size for matching with replacement depends on the sparseness of the control units. Overlap is reasonably good with the current data generating process, and the average group size increases with only 20% compared to matching without replacement. The full matching methods lead to larger groups since they do not discard units. Given the unconditional propensity score of 26.5%, the expected minimum group size among matchings that do not discard units is 3.77 units, which is close to what the methods produce. The groups are slightly smaller with traditional full matching. This is likely a result of both that implementation's optimality and its use of a non-neutral objective function (i.e., L_{tc}^{Sum}). In the second column, we present the standard deviation of the group sizes. We see that the full matching methods have considerably higher variation. This is a result of their ability to adapt the matching to the distribution of units in the covariate space.

Next, we investigate the share of the sample that is discarded. For a given level of balance, we want to drop as few units as possible. Predictably, 1:1-matching leads to that

Table 1: Performance of matching methods with samples of 10,000 units.

	Panel A: Distances					Panel B: Group structure			
	L^{Max}	L^{Max}_{tc}	L^{Mean}	L^{Mean}_{tc}	L^{Sum}_{tc}	Size	σ(Size)	% drop	σ(wgh)
Greedy 1:1	5.68	8.26	2.85	3.06	0.87	2.00	0.00	47.03	1.81
Optimal 1:1	4.83	7.04	2.55	2.74	0.78	2.00	0.00	47.03	1.81
Replacement 1:1	0.48	0.53	0.65	0.67	0.19	2.41	0.87	54.73	2.85
Greedy 1:2	10.31	15.00	8.05	11.92	6.77	3.00	0.00	20.54	0.85
Optimal 1:2	10.15	14.77	9.50	11.29	6.41	3.00	0.00	20.54	0.85
Full matching	1.00	1.00	1.00	1.00	1.00	4.24	3.51	0.00	1.97
GFM	1.00	1.00	0.99	0.98	1.05	4.73	3.55	0.00	2.15
Refined GFM	0.94	1.30	0.98	1.10	1.19	4.54	3.30	0.00	2.07

	Panel C: Covariate balance					Panel D: Estimator			
	X_1	X_2	X_1^2	X_2^2	X_1X_2	Bias	SE	RMSE	$\frac{\text{Bias}}{\text{RMSE}}$
Unadjusted	500.32	502.00	101.07	100.69	131.37	1087.12	1.48	39.81	0.999
Greedy 1:1	50.19	50.32	62.74	62.56	139.13	53.68	1.04	2.22	0.884
Optimal 1:1	50.10	50.25	62.24	62.07	139.80	54.14	1.04	2.24	0.885
Replacement 1:1	0.41	0.41	0.73	0.73	0.80	0.32	1.17	1.17	0.010
Greedy 1:2	244.42	245.03	145.45	145.07	385.24	429.95	1.61	15.82	0.995
Optimal 1:2	244.48	245.20	146.26	145.83	383.47	429.41	1.60	15.80	0.995
Full matching	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.037
GFM	0.71	0.71	0.71	0.72	0.76	0.57	1.03	1.03	0.020
Refined GFM	1.03	1.03	0.97	0.97	1.09	1.19	1.01	1.01	0.043

Notes: Each panel presents measures of the investigated matching methods' performance. The first panel investigates the five functions aggregating within-group distances as discussed in the text. The second panel investigates the composition of the matched group. In particular, the average group size, the standard deviation of the size, share of units not assigned to a group and the standard deviation in the weights of the control units implied by the matchings. The third panel presents balance between the treatment groups on the first two moments of the covariates. The final panel investigates the treatment effect estimator. When applicable, measures are normalized by traditional full matching. Each measure is based on 10,000 simulation rounds; simulation errors are negligible.

a sizable portion of the sample are left unassigned. This is especially the case when we match with replacement. Fewer units are discarded with 1:2-matching, and by construction, no units are discarded with the full matching methods.

The fourth column reports the standard deviation of the estimator weights implied by the matching. As discussed in Section 2.1, weight variation is necessary to balance an unbalanced sample. However, for a given level of balance, we want the weights to be as uniform as possible. Since we are estimating ATT, the implied weights for treated units are fixed at $|\mathbf{w}_1|^{-1}$ for all methods. Weights for control do, however, vary. The implied weight for control unit i assigned to matched group $\mathbf{m}$ is:

$$\text{wgh}_i = \frac{|\mathbf{w}_1 \cap \mathbf{m}|}{|\mathbf{w}_1| \times |\mathbf{w}_0 \cap \mathbf{m}|}, \tag{12}$$

and zero if not assigned to a group. Examining the results, we see that the amount of variation is correlated with how well the methods are able to minimize distances. For example, 1:1-matching with replacement produces the shortest distances, but as a result, also the most weight variation. The choice of method depends on how one resolves the trade-off between weight variation and balance, which, in turn, depends on how strongly the covariates are correlated with the outcome and treatment assignment. For this reason, the best choice of matching method will differ depending on the data generating process. It appears, however, that all full matching methods lead to matchings with substantially smaller distances than 1:1-matching without replacement with only slightly higher weight variation (i.e., close to a Pareto improvement). Similarly, but less pronounced, the `optmatch` package dominates the `scclust` package; the former produces about the same distances but with less weight variation.

5.2.3 Covariate balance

The aim of matching is to equalize covariate distributions. Panel C presents the absolute mean difference between the adjusted treatment groups on the first two moments of the

covariates. The balance measure is weighted in the same way as the estimator, i.e., by the number of treated units in each group. We include the results for the unadjusted sample (i.e., when no matching is done). As with the distance measures in Panel A, we normalize the results by traditional full matching since the scaling is arbitrary. How well the methods balance the sample depends on the data generating process. If, for example, the propensity score is constant, matching would only correct sampling variations and, thus, only lead to minor improvements compared to the unadjusted sample. While we do not claim that the current model is representative of observational studies in general, we expect the qualitative conclusions to hold across settings.

The results largely follow the same pattern as for the distance measures. Matching with replacement leads to be best balance with the full matching methods as a close second. The other methods are more than an order of magnitude worse. They even result in worse balance than no matching in some of the higher moments. This is largely an effect of that those moments are fairly balanced already in the unadjusted sample, but the behavior is still disconcerting. We note that generalized full matching without refinements leads to better balance than the other full matching methods. We have not found an explanation for this behavior and do not expect it to hold across settings.

5.2.4 Treatment effect estimator

We finally investigate the properties of the matching estimator. As with the balance measures, these results depend on the details of the data generating process. While the exact results might not generalize, the qualitative conclusions should. The results are presented in Panel D. The measures are, again, normalized by traditional full matching to ease interpretation. The first column presents the bias of the methods. As expected, the unadjusted sample has substantial bias. 1:1- and 1:2-matching without replacement reduces the bias by about 95% and 60%. While this is a substantial improvement, it is still more than

an order of magnitude greater than the bias for 1:1-matching with replacement and full matching.

The second column reports the estimator's standard error. The variance of the estimator depends mainly on two factors. First, while matching is primarily used to adjust for systematic covariate differences between treatment group, it will also adjust for unsystematic differences. Such chance-imbalances lead to increased estimator variance and can, thus, be prevented with matching (compare with adjustment in randomized experiments). Second, for a given level of balance, more weight variation (as investigated in Panel B) will lead to more estimator variance as the information in the sample is used less effectively. The trade-off between balance and weight variance is reflected in the standard errors. 1:k-matching leads to the highest standard error even if it has the least weight variation; it does not produce sufficient balance. On the opposite extreme, matching with replacement leads to excellent balance but large weight variation. Subsequently, its standard error is 17% higher than with full matching. Reflecting the increase in weight variations, the generalized full matching leads to slightly larger standard errors than traditional full matching.

The final two columns investigate the root mean square error (RMSE) of the estimator and the bias's share of the RMSE. As the full matching methods lead to both low bias and variance, they perform well on the RMSE. Matching with replacement is 17% worse than traditional full matching, but it is still considerably better than 1:1- and 1:2-matching. The bias to RMSE ratio shows how accurate our inferences will be. In particular, if the systematic error (i.e., the bias) is a large part of the total error, variance estimators will not capture the uncertainty of the estimation and our conclusions will be misleading. This measure is scale-free and is, therefore, not normalized. As expected, the RMSE consists almost exclusively of bias in the unadjusted sample—any conclusions from such analyses are likely very misleading. 1:1-matching and 1:2-matching result in only minor improvements. In stark contrast, matching with replacement and full matching lead to large reductions in

the ratio; the bias is only between 1% and 4% of the RMSE. This ratio critically depends on the dimensionality of the covariate space (Abadie & Imbens 2006), but we expect a similar pattern to hold across settings.

6 Concluding remarks

Matching is an important tool for empirical researchers. It has, however, not always been possible to use the method to its fullest potential. Algorithms with guaranteed optimality properties have limited scope and require vast computational resources; they are rarely useful when designs are complex or samples are large. Investigators have, therefore, been forced to use inferior methods to construct their matchings, either by simplifying the problem by allowing for a less than ideal matching structures or by using ad hoc procedures such as greedy matching.

In this paper, we have sought to remedy this situation. We have introduced a new matching method—generalized full matching—that is applicable to a large number of complex designs. As indicated by its name, the method generalizes the idea behind full matching. Both methods admit great flexibility in the construction of the matching. The flexibility allows for improved match quality without discarding large parts of the sample. However, unlike traditional full matching, our method is not restricted to one particular design; it can accommodate any number of treatment conditions and any size constraints over those conditions. So far, studies with such designs were forced to solve several matching problems and merge the resulting matchings in a post-processing step. Besides being tiring and error-prone, the approach rarely maintains optimality even if the underlying methods are optimal with respect to each separate matching problem. Generalized full matching allows the investigator to construct a single matching that directly corresponds to the desired design.

We introduced an algorithm aimed to construct generalized full matchings. We proved that it derives near-optimal solutions for any matching problem. In particular, the maximum dissimilarity between two units in a matched group is at most four times as large as the maximum dissimilarity in the optimal matching. This bound is likely conservative. We expect the typical performance to be much closer to optimal. For example, in our simulation study, our algorithm performs almost on par with the optimal traditional full matching algorithm.

The algorithm's main strength is its frugal use of computational resources. As data sets grow in size, so does the demand for computationally efficient methods to analyze them. Sample sizes that were unheard of a few decades ago are now commonplace. While vast amounts of data typically make our inferences more credible, it does not solve the fundamental problem of causal inference (Holland 1986, Titiunik 2015). Confounding is a pressing issue no matter the sample size. In fact, because standard errors are often negligible when data is abundant, bias is an even more acute problem in large samples. However, it is often an insurmountable computational problem to derive well-performing matchings even in samples of rather modest sizes. The presented algorithm provides an attractive alternative in these settings.

We want to stress that our algorithm should be used as a complement to existing matching methods. There are settings where we would discourage its use. Unlike existing algorithms based on network flows (see, e.g., Hansen & Klopfer 2006), the one presented in this paper does not necessarily derive optimal solutions. For this reason, best practice is still to use existing optimal algorithms when possible (i.e., when the design conforms to the classic two-treatment setup and the sample is not too large). Furthermore, several refinements to the traditional full matching algorithm have been developed since its conception. Hansen (2004) shows, for example, how to impose bounds on the ratio between the number of treated and control units within the matched groups. This limits the weight

variation of the matching and allows the investigator to directly control the trade-off between balance and weight variation. When used with care, this can reduce the estimation error considerably. While a similar effect can be achieved by adjusting the constraints in a generalized full matching, it is a blunt solution without the same level of control as in Hansen's formulation.

Network flow algorithms can also be adapted to construct matching with *fine balance* (Rosenbaum et al. 2007). That is, treatment groups with identical marginal distributions on a set of covariates in settings where exact matches are not possible. Pimentel, Kelz, Silber & Rosenbaum (2015) introduces a refinement of fine balancing where categorical covariates can be prioritized so that matches are constructed in a hierarchical fashion; the method ensures fine balance on covariates deemed more important before improving the balance in the rest of the covariate distribution. The current implementation of our algorithm cannot accommodate such global objectives. In another refinement, Pimentel, Yoon & Keele (2015) devise a method to combine variable-ratio matching (as in full matching) with fine balance. However, unlike both traditional and generalized full matching, their procedure requires that the ratio between treated and controls is *estimated* through the propensity score in a preprocessing step. While this retains much of the benefits of full matching, it is less adaptive to the data and will run to similar issues as other matching methods with fixed group sizes.

If the sample is small enough, it is sometimes feasible to use algorithms with an exponential time complexity. In such cases, Zubizarreta (2012) provides a general framework which admits great flexibility in both the constraints and objective of the matching. When feasible, this approach gives the investigator the greatest control of the matching problem which, when used with care, will allow for superior performance.

References

Abadie, A. & Imbens, G. W. (2006), 'Large Sample Properties of Matching Estimators for Average Treatment Effects', *Econometrica* **74**(1), 235–267.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R. & Wu, A. Y. (1998), 'An optimal algorithm for approximate nearest neighbor searching fixed dimensions', *Journal of the ACM* **45**(6), 891–923.

Cochran, W. G. & Rubin, D. B. (1973), 'Controlling bias in observational studies: A review', *Sankhyā: The Indian Journal of Statistics, Series A* **35**(4), 417–446.

Friedman, J. H., Bentley, J. L. & Finkel, R. A. (1977), 'An algorithm for finding best matches in logarithmic expected time', *ACM Transactions on Mathematical Software* **3**(3), 209–226.

Hansen, B. B. (2004), 'Full matching in an observational study of coaching for the SAT', *Journal of the American Statistical Association* **99**(467), 609–618.

Hansen, B. B. & Klopfer, S. O. (2006), 'Optimal full matching and related designs via network flows', *Journal of Computational and Graphical Statistics* **15**(3), 609–627.

Higgins, M. J., Sävje, F. & Sekhon, J. S. (2016), 'Improving massive experiments with threshold blocking', *Proceedings of the National Academy of Sciences* **113**(27), 7369–7376.

Higgins, M. J., Sävje, F. & Sekhon, J. S. (2017), On the computational complexity of size constrained clustering problems.

Holland, P. W. (1986), 'Statistics and causal inference', *Journal of the American Statistical Association* **81**(396), 945–960.

Imbens, G. W. & Rubin, D. B. (2015), *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*, Cambridge University Press, New York, NY, USA.

Knuth, D. E. (1998), *Sorting and searching*, Vol. 3 of *The Art of Computer Programming*, 2$^{\text{th}}$ edn, Addison Wesley Longman, Redwood City, CA.

Kuhn, H. W. (1955), 'The hungarian method for the assignment problem', *Naval Research Logistics Quarterly* **2**(1-2), 83–97.

Li, S., Vlassis, N., Kawale, J. & Fu, Y. (2016), Matching via dimensionality reduction for estimation of treatment effects in digital marketing campaigns, *in* 'Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence', pp. 3768–3774.

Pimentel, S. D., Kelz, R. R., Silber, J. H. & Rosenbaum, P. R. (2015), 'Large, sparse optimal matching with refined covariate balance in an observational study of the health outcomes produced by new surgeons', *Journal of the American Statistical Association* **110**(510), 515–527.

Pimentel, S. D., Yoon, F. & Keele, L. (2015), 'Variable-ratio matching with fine balance in a study of the peer health exchange', *Statistics in Medicine* **34**(30), 4070–4082.

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Rosenbaum, P. R. (1991), 'A characterization of optimal designs for observational studies', *Journal of the Royal Statistical Society. Series B (Methodological)* **53**(3), 597–610.

Rosenbaum, P. R. (2002), *Observational studies*, 2 edn, Springer, New York.

Rosenbaum, P. R. (2010), *Design of Observational Studies*, Springer, New York.

Rosenbaum, P. R., Ross, R. N. & Silber, J. H. (2007), 'Minimum distance matched sampling with fine balance in an observational study of treatment for ovarian cancer', *Journal of the American Statistical Association* **102**(477), 75–83.

Rosenbaum, P. R. & Rubin, D. B. (1983), 'The central role of the propensity score in observational studies for causal effects', *Biometrika* **70**(1), 41–55.

Sekhon, J. S. (2011), 'Multivariate and Propensity Score Matching Software with Automated Balance Optimization: The Matching Package for R', *Journal of Statistical Software* **42**(7), 1–52.

Stuart, E. A. (2010), 'Matching Methods for Causal Inference: A Review and a Look Forward', *Statistical Science* **25**(1), 1–21.

Titiunik, R. (2015), 'Can big data solve the fundamental problem of causal inference?', *PS: Political Science & Politics* **48**, 75–79.

Zubizarreta, J. R. (2012), 'Using Mixed Integer Programming for Matching in an Observational Study of Kidney Failure After Surgery', *Journal of the American Statistical Association* **107**(500), 1360–1371.

SUPPLEMENTARY MATERIALS

Generalized Full Matching

S1 Overview of graph theory

Graph A graph $G = (V, E)$ consists of a set of indices $V = \{a, b, \cdots\}$, called vertices, and a set of 2-element subsets of V, called edges. If an edge $\{i, j\} \in E$, we say that i and j are connected in the graph. In a directed graph (or digraph), the edges (which are then called arcs) are ordered sets (i, j). In other words, i can be connected to j without the reverse being true in a digraph.

Weighted graph A weighted graph assign a weight or cost to each edge or arcs. In our case, the weights are exclusively given by the distance of the connected vertices according to the distance metric used in the matching problem.

Adjacent Vertices i and j are adjacent in G if an edge (or an arc) connecting i and j exists in E.

Geodesic distance The geodesic distance between i and j in G is the number of edges or arcs (in our case, of any directionality) in the shortest path connecting i and j in G.

Subgraph $G_1 = (V_1, E_1)$ is a subgraph of $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. In that case, we also say that G_2 is a supergraph of G_1. G_1 is a spanning subgraph of G_2 if $V_1 = V_2$.

Complete graph G is complete if $\{i, j\} \in E$ for any two vertices $i, j \in V$. If G is directed, both (i, j) and (j, i) must be in E.

Union The union of $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

Graph difference The difference between two graphs, $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, spanning the same set of vertices is $G_1 - G_2 = (V, E_1 \setminus E_2)$.

Independent set A set of vertices $I \subseteq V$ is independent in G if no two vertices in the set are adjacent:

$$\forall\, i, j \in I,\ \{i, j\} \notin E.$$

Maximal independent set An independent set of vertices I in G is maximal if for any additional vertex $i \in V$ the set $\{i\} \cup I$ is not independent:

$$\forall\, i \in V \setminus I,\ \exists\, j \in I,\ \{i, j\} \in E.$$

Cluster graph The (directed) cluster graph induced by some partition of V is the graph where arcs exists between any pair of units in the same component of the partition and no other arcs exist.

Adjancency matrix The adjancency matrix $\mathbf{A}$ of a graph G with n vertices is a n-by-n binary matrix where the entry i, j is one if the edge $\{i, j\}$ or arc (i, j) is in E, and otherwise zero.

S2 Proofs

We here provide proofs for the propositions in the article. The relevant propositions from the article are restated with their original numbering for reference.

S2.1 Optimality

Recall the two objective functions:

$$L^{Max}(\mathbf{M}) = \max_{\mathbf{m} \in \mathbf{M}} \max\{\mathrm{d}(i, j) : i, j \in \mathbf{m}\}, \tag{1}$$

$$L^{Max}_{tc}(\mathbf{M}) = \max_{\mathbf{m} \in \mathbf{M}} \max\{\mathrm{d}(i, j) : i, j \in \mathbf{m} \wedge W_i \neq W_j\}. \tag{2}$$

Lemma 8. *The closed neighborhood of each vertex in the $\mathcal{C}$-compatible nearest neighbor digraph satisfies the matching constraints $\mathcal{C} = (c_1, \cdots, c_k, t)$:*

$$\forall i \in V, \forall j \in \{1, \cdots, k\}, |\,\mathrm{N}[i] \cap \mathbf{w}_j| \geq c_j, \qquad \textit{and} \qquad \forall i \in V, |\,\mathrm{N}[i]| \geq t. \tag{3}$$

Proof. This follows directly from the construction of $G_{\mathcal{C}}$. For each treatment-specific constraint, $c_1, \cdots, c_k$, the first step of the algorithm ensures that each vertex has that many arcs pointing to units assigned to the corresponding treatment condition. Similarly, if $t > c_1 + c_2 + \cdots + c_k$, the second step draws additional arcs so that each vertex has t outward-pointing arcs in total. □

Lemma S1. *Each vertex has at least one labeled vertex in its neighborhood in $G_{\mathcal{C}}$.*

Proof. By definition, all vertices in a closed neighborhood of a seed are labeled. That is, ℓ is a labeled vertex if and only if $\exists i \in \mathbf{S}, \ell \in \mathrm{N}[i]$. Suppose the lemma does not hold, i.e., that some vertex i does not have a labeled vertex in its neighborhood:

$$\exists i, \forall \ell \in \mathrm{N}[i], \nexists j \in \mathbf{S}, \ell \in \mathrm{N}[j]. \tag{4}$$

It follows directly that i cannot be a seed as all vertices in its neighborhood would otherwise be labeled by definition. Note that (4) entails that i's neighborhood does not have any overlap with any seed's neighborhood:

$$\forall j \in \mathbf{S}, \mathrm{N}[i] \cap \mathrm{N}[j] = \emptyset, \tag{5}$$

However, this violates the maximality condition in the definition of a valid set of seeds (see the third step of the algorithm) and, subsequently, a vertex such as i is not possible. □

Lemma S2. $\mathbf{M}_{alg}$ *is an admissible generalized full matching with respect to the matching constraint $\mathcal{C} = (c_1, \cdots, c_k, t)$:*

$$\mathbf{M}_{alg} \in \mathcal{M}_{\mathcal{C}}. \tag{6}$$

Proof. We must show that $\mathbf{M}_{alg}$ satisfies the four conditions of an admissible generalized full matching in Definition 2.

Step 5 of the algorithm ensures that $\mathbf{M}_{alg}$ is spanning. At this step, any vertex that lacks a label will be assigned the same label as one of the labeled vertices in its neighborhood. Lemma S1 ensures that at least one labeled vertex exists in the neighborhoods of the unassigned vertices.

No vertex is assigned more than one label; $\mathbf{M}_{alg}$ is disjoint. Vertices are only assigned labels in either Step 4 or 5, but never in both. Step 3 ensures that the neighborhoods of the seeds are non-overlapping. Thus vertices will be assigned at most one label in Step 4. In Step 5, vertices are explicitly assigned only one label even if several labels could be represented in a vertex's neighborhood.

The two last conditions in Definition 2 are ensured by Lemma 8. Step 4 of the algorithm ensures that each matched group is a superset of a seed's neighborhood. From Lemma 8, we have that this neighborhood will satisfy the matching constraints. □

Lemma S3. *If the arc weights in the $\mathcal{C}$-compatible nearest neighbor digraph are bounded by some λ, the maximum within-group distance in $\mathbf{M}_{alg}$ is bounded by 4λ:*

$$\forall (i,j) \in E_{\mathcal{C}},\, \mathrm{d}(i,j) \leq \lambda \Rightarrow \max_{\mathbf{m} \in \mathbf{M}_{alg}} \max\{\mathrm{d}(i,j) : i,j \in \mathbf{m}\} \leq 4\lambda. \tag{7}$$

Proof. First, consider the distance from any vertex i to the seed in its matched group, denoted j. If i is a seed, we have $i = j$ as each matched group contains exactly one seed by construction. By the self-similarity property of distance metrics, the distance is zero: $\mathrm{d}(i,j) = 0$. If i is a labeled vertex (i.e., assigned a label in Step 4 of the algorithm), we have $(j,i) \in E_{\mathcal{C}}$ by definition of labeled vertices. By assumption, $\mathrm{d}(j,i)$ is bounded by λ. Due to the symmetry property of distance metrics, this also bounds $\mathrm{d}(i,j)$. If i is not a labeled vertex (i.e., assigned a label in Step 5), it will be adjacent in $G_{\mathcal{C}}$ to a labeled vertex, ℓ, in its matched group as implied by Lemma S1. We have $(i,\ell) \in E_{\mathcal{C}}$ so the distance between i

and ℓ is bounded by λ. As ℓ is labeled, we have $(j, \ell) \in E_{\mathcal{C}}$ which implies that $\mathrm{d}(j, \ell) \leq \lambda$. From the triangle inequality property of metrics, we have that the distance between i and the seed, j, is at most 2λ.

Now consider any two vertices assigned to the same matched group. We have shown that the distance from each of these vertices to their (common) seed is at most 2λ. By applying the triangle inequality once more, we bound the distance between the two non-seed vertices by 4λ. □

Lemma S4. *The $\mathcal{C}$-compatible nearest neighbor digraph has the smallest maximum arc weight among all digraphs compatible with $\mathcal{C}$, i.e., all graph wherein each vertex's closed neighborhood contains $c_1, c_2, \cdots, c_k$ vertices of each treatment condition and t vertices in total.*

Proof. The definition of the directed neighborhoods is asymmetric in the sense that if i is in j's neighborhood, j is not necessarily in i's neighborhood. Thus, whether a vertex's neighborhood satisfies the constraints is independent of whether other vertices' neighborhoods do so. As a consequence, to minimize the maximum arc weight, we can simply minimize the maximum arc weight in each neighborhood separately. To minimize the arc weights in a single neighborhood, we draw arcs to the vertices closest to the vertex so that the matching constraints are fulfilled. This is exactly the procedure the algorithm follows. □

Lemma 9. *The distance between any two vertices connected by an arc in the $\mathcal{C}$-compatible nearest neighbor digraph, $G_{\mathcal{C}} = (V, E_{\mathcal{C}})$, is less or equal to the maximum within-group distance in an optimal matching:*

$$\forall (i, j) \in E_{\mathcal{C}},\ \mathrm{d}(i, j) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} L^{Max}(\mathbf{M}). \tag{8}$$

Proof. Let w^* be the maximum within-group distance in an optimal matching and let $w^+_{\mathcal{C}}$

be the maximum weight of an arc in $G_{\mathcal{C}}$:

$$w^* = \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} L^{Max}(\mathbf{M}),$$
$$w^+_{\mathcal{C}} = \max\{\mathrm{d}(i,j) : (i,j) \in E_{\mathcal{C}}\}.$$

Furthermore, let $B_{\mathcal{C}} = (\mathbf{U}, E^b_{\mathcal{C}})$ be the digraph that contains arcs between all units at a distance strictly closer than $w^+_{\mathcal{C}}$:

$$E^b_{\mathcal{C}} = \{(i,j) : \mathrm{d}(i,j) < w^+_{\mathcal{C}}\}. \tag{9}$$

$B_{\mathcal{C}}$ must contain a vertex whose neighborhood does not satisfy the size constraints. If no such vertex exists, a digraph compatible with $\mathcal{C}$ with a smaller maximum arc weight than in $G_{\mathcal{C}}$ exists as a subgraph of $B_{\mathcal{C}}$. This contradicts Lemma S4.

Let $B_{op} = (\mathbf{U}, E^b_{op})$ be the digraph that contains arcs between all units at a distance weakly closer than w^*:

$$E^b_{op} = \{(i,j) : \mathrm{d}(i,j) \leq w^*\}. \tag{10}$$

By construction, B_{op} is a supergraph of the cluster graph induced by the optimal matching. That is, arcs are drawn in B_{op} between all units assigned to the same matched group in the optimal matching. As the optimal matching is admissible, each vertex's neighborhood in B_{op} is compatible with $\mathcal{C}$.

Suppose the lemma does not hold: $w^+_{\mathcal{C}} > w^*$. It follows that $E^b_{op} \subset E^b_{\mathcal{C}}$. As at least one vertex's neighborhood does not satisfy the size constraint in $B_{\mathcal{C}}$, that must be the case in B_{op}. This, however, implies that the optimal matching is not admissible which, in turn, contradicts optimality. □

Theorem 10. $\mathbf{M}_{alg}$ *is a 4-approximate generalized full matching with respect to the matching constraint* $\mathcal{C} = (c_1, \cdots, c_k, t)$ *and matching objective* L^{Max}*:*

$$\mathbf{M}_{alg} \in \mathcal{M}_{\mathcal{C}}, \qquad \textit{and} \qquad L^{Max}(\mathbf{M}_{alg}) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} 4L^{Max}(\mathbf{M}). \tag{11}$$

Proof. Admissibility follows from Lemma S2. Approximate optimality follows from Lemma S3 and 9. □

Lemma S5. *When all treatment-specific constraints are less or equal to one and the overall size constraint is the sum of the treatment-specific constraints, the distance between any two vertices connected by an arc in $G_{\mathcal{C}} = (V, E_{\mathcal{C}})$ is less or equal to the maximum within-group distance in an optimal matching with L_{tc}^{Max} as objective:*

$$c_1, c_2, \cdots, c_k \leq 1 \wedge t = \textstyle\sum_{x=1}^{k} c_x \Rightarrow \forall (i,j) \in E_{\mathcal{C}}, \mathrm{d}(i,j) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} L_{tc}^{Max}(\mathbf{M}). \tag{12}$$

Proof. Let w_s^+ be the maximum weight of an arc connecting two units with the same treatment conditions in $G_{\mathcal{C}}$, let w_d^+ the maximum arc weight between units with different conditions:

$$w_s^+ = \max\{\mathrm{d}(i,j) : (i,j) \in E_{\mathcal{C}} \wedge W_i = W_j\},$$
$$w_d^+ = \max\{\mathrm{d}(i,j) : (i,j) \in E_{\mathcal{C}} \wedge W_i \neq W_j\}.$$

Note that:

$$\max\{\mathrm{d}(i,j) : (i,j) \in E_{\mathcal{C}}\} = \max\{w_s^+, w_d^+\}.$$

Consider w_s^+. Since $c_1, c_2, \cdots, c_k \leq 1$ and $t = \sum_{x=1}^{k} c_x$, each unit will have at most one arc pointing to a unit with the same treatment condition as its own:

$$\forall i,\ |\{(i,j) : (i,j) \in E_{\mathcal{C}} \wedge W_i = W_j\}| = c_{W_i} \leq 1. \tag{13}$$

From the self-similarity and non-negativity properties of distance metrics, we have:

$$\forall i, j,\ 0 = \mathrm{d}(i,i) \leq \mathrm{d}(i,j). \tag{14}$$

By construction of $G_{\mathcal{C}}$, all arcs in the set will be self-loops and, thus, at distance zero:

$$w_s^+ = \max\{\mathrm{d}(i,i) : (i,i) \in E_{\mathcal{C}}\} = 0. \tag{15}$$

From non-negativity, it follows that:

$$\max\{\mathrm{d}(i,j) : (i,j) \in E_{\mathcal{C}}\} = \max\{0, w_d^+\} = w_d^+.$$

Let w^* be the maximum within-group distance between units assigned to different treatment conditions when L_{tc}^{Max} is used as objective:

$$w^* = \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} L_{tc}^{Max}(\mathbf{M}).$$

Let $B_d = (\mathbf{U}, E_d^b)$ be the digraph that contains all arcs between units that either are strictly closer than w_d^+ or have the same treatment condition:

$$E_d^b = \{(i,j) : \mathrm{d}(i,j) < w_d^+ \vee W_i = W_j\}. \tag{16}$$

Following the same logic as in the proof of Lemma 9, B_d must contain a vertex whose neighborhood is not compatible with $\mathcal{C}$.

Let $B_{op} = (\mathbf{U}, E_{op}^b)$ be the digraph that contains all arcs between units that either are weakly closer than w^* or have the same treatment condition:

$$E_{op}^b = \{(i,j) : \mathrm{d}(i,j) \leq w^* \vee W_i = W_j\}. \tag{17}$$

By construction, B_{op} is a supergraph of the cluster graph induced by the optimal matching. That is, arcs are drawn in B_{op} between all units assigned to the same matched group in the optimal matching. As the optimal matching is admissible, each vertex's neighborhood in B_{op} is compatible with $\mathcal{C}$.

Assume $w_d^+ > w^*$. It follows that $E_{op}^b \subset E_d^b$. As at least one vertex's neighborhood does not satisfy the size constraint in B_d, that must be the case in B_{op}. This, however, implies that the optimal matching is not admissible which, in turn, contradicts optimality. We conclude that $w_d^+ \leq w^*$. □

Theorem 11. *For matching constraints $\mathcal{C} = (1, \cdots, 1, k)$, $\mathbf{M}_{alg}$ is a 4-approximate traditional full matching with respect to matching objective L_{tc}^{Max}:*

$$\mathbf{M}_{alg} \in \mathcal{M}_{\mathcal{C}}, \qquad \textit{and} \qquad L_{tc}^{Max}(\mathbf{M}_{alg}) \leq \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} 4L_{tc}^{Max}(\mathbf{M}). \tag{18}$$

Proof. Admissibility follows from Lemma S2. Note that all distances considered by L_{tc}^{Max} are considered by L^{Max} as well. As a result, the latter acts as a bound for the former:

$$\forall \mathbf{M} \in \mathcal{M}_{\mathcal{C}}, L_{tc}^{Max}(\mathbf{M}) \leq L^{Max}(\mathbf{M}). \tag{19}$$

Approximate optimality follows from Lemma S3 and S5:

$$L_{tc}^{Max}(\mathbf{M}_{alg}) \leq L^{Max}(\mathbf{M}_{alg}) \leq 4 \min_{\mathbf{M} \in \mathcal{M}_{\mathcal{C}}} L_{tc}^{Max}(\mathbf{M}).$$

□

S2.2 Complexity

Lemma S6. *A $\mathcal{C}$-compatible nearest neighbor digraph can be constructed in polynomial time using linear memory.*

Proof. In the first step of the algorithm, we construct G_w as the union of $\mathrm{NN}(c_x, G(\mathbf{U} \to \mathbf{w}_x))$ for each treatment condition x. The operands of this union can be constructed by with nearest neighbor searches for each treatment condition. With a naive implementation, such searches can be done sequentially for each $i \in \mathbf{U}$ by sorting the set $\{\mathrm{d}(i,j) : j \in \mathbf{w}_x\}$ and drawing an arc from i to the first c_x elements in the sorted set. When using standard sorting algorithms, this has a time complexity of $O(n|\mathbf{w}_x| \log |\mathbf{w}_x|)$ and a space complexity of $O(c_x n)$ (Knuth 1998). Note that $|\mathbf{w}_x| < n$ for all treatments, so $O(n^2 \log n)$ also provides a bound. The union can be performed in linear time in the total number of arcs, $O[(c_1 + c_2 + \cdots + c_k)n]$. As each $\mathrm{NN}(c_x, G(\mathbf{U} \to \mathbf{w}_x))$ can be derived sequentially and the size constraints are fixed, the G_w digraph can be constructed in $O(n^2 \log n)$ time.

In the second step, G_r can be constructed in a similar fashion. For each $i \in \mathbf{U}$, sort the set $\{\mathrm{d}(i,j) : j \in \mathbf{U} \wedge (i,j) \notin E_w\}$ and draw an arc from i to the first $r = t - c_1 - \cdots - c_k$ elements in that set. Like above, this has a complexity of $O(n^2 \log n)$. Finally, the union between G_w and G_r can be constructed in linear time in the total number of arcs. As the

number of arcs per vertex is fixed at t, the union is completed in $O(n)$ time. The steps are sequential so the total complexity of both Step 1 and 2 is $O(n^2 \log n)$. □

Remark S7. For most common metrics, standard sorting algorithms are inefficient. Storing the data points in a structure made for the purpose, such as a kd- or bd-tree, typically leads to large improvements (Friedman et al. 1977). Each $\mathrm{NN}(c_x, G(\mathbf{U} \to \mathbf{w}_x))$ can then be constructed in $O(n \log n)$ average time, without changing the memory complexity. However, this approach typically requires a preprocessing step to build the search tree. In the proof of Lemma S6, the search set is unique for each vertex when G_r is constructed. We can, therefore, not use these specialized algorithms if we construct G_r in the way suggested there. However, the construction can easily be transformed into a problem with a fixed search set. Note that:

$$\mathrm{NN}(r, G(\mathbf{U} \to \mathbf{U}) - G_w) = \mathrm{NN}(r, \mathrm{NN}(t, G(\mathbf{U} \to \mathbf{U})) - G_w). \tag{20}$$

That is, finding the r nearest neighbors not already connected in G_w is the same as finding the r nearest neighbors not already connected in G_w among the t nearest neighbors in the complete graph. The first nearest neighbor search, $\mathrm{NN}(t, G(\mathbf{U} \to \mathbf{U}))$, has a fixed search set and can thus be completed in $O(n \log n)$. The second nearest neighbor search involves sorting at most t elements for each vertex, which is done in constant time as t is fixed.

Theorem 12. *In the worst case, the generalized full matching algorithm terminates in polynomial time using linear memory.*

Proof. The algorithm runs sequentially. The first and second step can be completed in $O(n^2 \log n)$ worst case time as shown in Lemma S6, or, in many cases, in $O(n \log n)$ average time as discussed in Remark S7.

Step 3 and 4 can be done by sequentially labeling seeds and their neighbors as they are selected. Any vertex whose neighborhood does not contain any labeled vertices can be

a valid seed, and any vertex that is adjacent to labeled vertices can never become a seed. Thus, iterating over the vertices in any order and greedily selecting unit will yield a valid set of seeds. As the size of each seed's neighborhood is fixed at t, this step is completed in $O(n)$ time.

Finally, assigning labels to unlabeled vertices in the last step can be done by iterating over their neighborhoods. Thus, Step 5 requires $O(n)$ time to complete. □

S3 Additional simulation results

Table S1: Aggregated distances for matching methods with samples of 1,000 and 10,000 units.

	1,000 units					10,000 units				
	L^{Max}	L^{Max}_{tc}	L^{Mean}	L^{Mean}_{tc}	L^{Sum}_{tc}	L^{Max}	L^{Max}_{tc}	L^{Mean}	L^{Mean}_{tc}	L^{Sum}_{tc}
Greedy 1:1	1.87	2.67	1.41	1.50	0.43	2.20	3.14	0.89	0.95	2.69
Optimal 1:1	1.29	1.85	1.20	1.27	0.36	1.87	2.68	0.80	0.85	2.41
Replacement 1:1	0.45	0.51	0.65	0.66	0.19	0.19	0.20	0.20	0.21	0.59
Greedy 1:2	3.66	5.23	3.21	4.31	2.46	3.99	5.71	2.51	3.69	20.97
Optimal 1:2	3.27	4.68	3.17	3.79	2.17	3.93	5.62	2.96	3.50	19.87
Full matching	1.00	1.00	1.00	1.00	1.00	0.39	0.38	0.31	0.31	3.10
GFM	1.00	1.00	0.99	0.98	1.05	0.39	0.38	0.31	0.30	3.25
Refined GFM	0.95	1.25	0.98	1.10	1.19	0.37	0.49	0.31	0.34	3.70

Notes: The measures are normalized by the result for traditional full matching in the sample with 1,000 units. Results are based on 10,000 simulation rounds; simulation errors are negligible.

Table S2: Covariate balance for matching methods with samples of 1,000 and 10,000 units.

	1,000 units					10,000 units				
	X_1	X_2	X_1^2	X_2^2	X_1X_2	X_1	X_2	X_1^2	X_2^2	X_1X_2
Unadjusted	52.48	52.73	10.72	10.91	13.11	52.528	52.735	10.707	10.874	12.588
Greedy 1:1	5.93	5.94	7.21	7.31	13.87	5.270	5.286	6.647	6.756	13.332
Optimal 1:1	5.94	5.95	7.08	7.19	14.09	5.260	5.279	6.594	6.704	13.396
Replacement 1:1	0.44	0.44	0.76	0.76	0.80	0.043	0.043	0.077	0.079	0.077
Greedy 1:2	26.23	26.39	15.48	15.76	35.59	25.662	25.741	15.410	15.667	36.914
Optimal 1:2	26.18	26.34	15.22	15.48	36.11	25.668	25.759	15.495	15.749	36.745
Full matching	1.00	1.00	1.00	1.00	1.00	0.105	0.105	0.106	0.108	0.096
GFM	0.74	0.75	0.77	0.77	0.80	0.074	0.075	0.075	0.077	0.073
Refined GFM	1.04	1.05	0.99	0.99	1.08	0.108	0.108	0.103	0.104	0.105

Notes: The measures are normalized by the result for traditional full matching in the sample with 1,000 units.

Table S3: Group composition for matching methods with samples of 1,000 and 10,000 units.

	1,000 units				10,000 units			
	Size	σ(Size)	% drop	σ(wgh)	Size	σ(Size)	% drop	σ(wgh)
Greedy 1:1	2.00	0.00	46.96	1.81	2.00	0.00	47.03	1.81
Optimal 1:1	2.00	0.00	46.96	1.81	2.00	0.00	47.03	1.81
Replacement 1:1	2.41	0.86	54.70	2.85	2.41	0.87	54.73	2.85
Greedy 1:2	3.00	0.00	20.44	0.84	3.00	0.00	20.54	0.85
Optimal 1:2	3.00	0.00	20.44	0.84	3.00	0.00	20.54	0.85
Full matching	4.24	3.50	0.00	1.93	4.24	3.51	0.00	1.97
GFM	4.74	3.54	0.00	2.13	4.73	3.55	0.00	2.15
Refined GFM	4.55	3.26	0.00	2.04	4.54	3.30	0.00	2.07

Notes: The columns report the average group size, the standard deviation of the size, share of units not assigned to a group and the standard deviation in the weights of the control units implied by the matchings. Results are based on 10,000 simulation rounds; simulation errors are negligible.

Table S4: Estimator performance for matching methods with samples of 1,000 and 10,000 units.

	1,000 units				10,000 units			
	Bias	SE	RMSE	$\frac{\text{Bias}}{\text{RMSE}}$	Bias	SE	RMSE	$\frac{\text{Bias}}{\text{RMSE}}$
Unadjusted	83.34	1.47	12.70	0.993	83.390	0.47	12.64	0.999
Greedy 1:1	4.86	1.04	1.26	0.583	4.118	0.33	0.71	0.884
Optimal 1:1	4.96	1.04	1.27	0.590	4.153	0.33	0.71	0.885
Replacement 1:1	0.11	1.17	1.16	0.015	0.024	0.38	0.37	0.010
Greedy 1:2	33.97	1.61	5.38	0.956	32.980	0.52	5.02	0.995
Optimal 1:2	34.08	1.59	5.39	0.957	32.939	0.51	5.01	0.995
Full matching	1.00	1.00	1.00	0.151	0.077	0.32	0.32	0.037
GFM	0.77	1.03	1.03	0.113	0.043	0.33	0.33	0.020
Refined GFM	1.20	1.02	1.02	0.178	0.091	0.32	0.32	0.043

Notes: The first three measures in each panel are normalized by the result for traditional full matching in the sample with 1,000 units. Results are based on 10,000 simulation rounds; simulation errors are negligible.

Table S5: Runtime and memory use by sample size for matching methods.

Panel A: Runtime (in minutes)

	100	500	1K	5K	10K	20K	50K	100K	200K	500K	1M	5M	10M	50M	100M
Greedy 1:1	0.01	0.01	0.01	0.01	0.03	0.09	0.62	2.71	12.85						
Optimal 1:1	0.06	0.06	0.08	1.08	5.03	19.40									
Replacement 1:1	0.01	0.01	0.01	0.01	0.03	0.09	0.58	2.60	12.15						
Greedy 1:2	0.01	0.01	0.01	0.01	0.03	0.09	0.63	2.76	13.20						
Optimal 1:2	0.06	0.06	0.10	5.02	26.19										
Full matching	0.06	0.06	0.08	0.30	0.87	2.84									
GFM	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.02	0.06	0.11	0.64	1.05	6.49	14.15
Refined GFM	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.02	0.03	0.08	0.16	0.96	1.52	9.31	20.07

Panel A: Memory use (in gigabytes)

	100	500	1K	5K	10K	20K	50K	100K	200K	500K	1M	5M	10M	50M	100M
Greedy 1:1	0.03	0.03	0.03	0.04	0.04	0.04	0.05	0.06	0.09						
Optimal 1:1	0.12	0.13	0.15	0.74	2.75	10.21									
Replacement 1:1	0.03	0.03	0.03	0.03	0.04	0.04	0.05	0.06	0.10						
Greedy 1:2	0.03	0.03	0.03	0.04	0.04	0.04	0.05	0.07	0.11						
Optimal 1:2	0.12	0.13	0.15	0.74	2.74										
Full matching	0.12	0.13	0.16	0.74	2.74	10.19									
GFM	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.05	0.06	0.12	0.21	0.88	1.73	8.57	17.11
Refined GFM	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.05	0.06	0.12	0.21	0.88	1.73	8.57	17.11

Notes: Each cell presents the runtime and memory of the matching implementations for different sample sizes. The first panel shows runtime in minutes, and the second panel shows memory use in gigabytes. Each column represents a different sample size where "K" denotes thousand and "M" denotes million. The rows indicate matching method. Greedy 1:1, Replacement 1:1 and Greedy 1:2 are implemented by the `Matching` package. Optimal 1:1, Optimal 1:2 and traditional full matching are implemented by the `optmatch` package. Generalized full matching (GFM) is implemented by the `scclust` package. Blank cells indicate that the corresponding matching method did not terminate successfully for the corresponding sample size within reasonable time and memory limits. Each measure is based on 1,000 simulation rounds.

References

Friedman, J. H., Bentley, J. L. & Finkel, R. A. (1977), 'An algorithm for finding best matches in logarithmic expected time', *ACM Transactions on Mathematical Software* **3**(3), 209–226.

Knuth, D. E. (1998), *Sorting and searching*, Vol. 3 of *The Art of Computer Programming*, 2$^{\text{th}}$ edn, Addison Wesley Longman, Redwood City, CA.