Journal of Statistical Software

January 2021, Volume 97, Issue 2. doi: 10.18637/jss.v097.i02

# microsynth: Synthetic Control Methods for Disaggregated and Micro-Level Data in R

**Michael W. Robbins**
RAND Corporation

**Steven Davenport**
Pardee RAND Graduate School

**Abstract**

The R package **microsynth** has been developed for implementation of the synthetic control methodology for comparative case studies involving micro- or meso-level data. The methodology implemented within **microsynth** is designed to assess the efficacy of a treatment or intervention within a well-defined geographic region that is itself a composite of several smaller regions (where data are available at the more granular level for comparison regions as well). The effect of the intervention on one or more time-varying outcomes is evaluated by determining a synthetic control region that resembles the treatment region across pre-intervention values of the outcome(s) and time-invariant covariates and that is a weighted composite of many untreated comparison regions. The **microsynth** procedure includes functionality that enables its user to (1) calculate weights for synthetic control, (2) tabulate results for statistical inferences, and (3) create time series plots of outcomes for treatment and synthetic control. In this article, **microsynth** is described in detail and its application is illustrated using data from a drug market intervention in Seattle, WA.

*Keywords*: synthetic control methods, micro-level, causal inference, **Synth**, program evaluation.

## 1. Introduction

Synthetic controls are a generalization of the difference-in-differences approach (Abadie and Gardeazabal 2003; Abadie, Diamond, and Hainmueller 2010, 2015). Difference-in-differences methods often require identification of one or more control cases (against which the treatment will be compared) and rely upon the plausibility that parallel trends affect the treatment and control after the intervention. The synthetic control method (SCM) offers a rigorous alternative wherein a comparison case is identified by constructing a "synthetic" control unit that represents a weighted combination of many untreated cases. Weights are calculated in order to maximize the similarity between the synthetic control and the treatment unit in terms of specified "matching" variables.

Generally, synthetic controls have been applied in the context of a single treatment case with a limited number (e.g., several dozens) of untreated cases for comparison. Such circumstances typically arise with data at a highly aggregated (e.g., macro-) level. The **Synth** package (Abadie, Diamond, and Hainmueller 2011) was developed for R and designed for this type of application. However, the relative dearth of treatment and comparison cases in such settings complicates efforts to (a) develop a synthetic control that exactly matches the treatment case, (b) precisely estimate the effect of treatment, (c) gauge the significance of that effect, and (d) jointly incorporate multiple outcome variables.

To address those limitations, the synthetic controls framework has been extended to settings involving disaggregated, micro-level data (Robbins, Saunders, and Kilmer 2017; Saunders, Robbins, and Ober 2017). This package was developed for implementation of SCM machinery in those settings, as **Synth** cannot be applied with data that have more than one treated case. Therefore, **microsynth** offers several new advantages and functionalities:

1. With the advantage of a large number of smaller-scale observations, **microsynth** is often better able to calculate weights that provide exact matches between treatment and synthetic control units across constraints based on time-invariant covariates and pre-intervention values of time-variant outcome variables—calibration (Särndal 2007) is used to calculate weights in such circumstances. This bolsters the conceptual framework behind the SCM. To facilitate the possibility that an exact match between the treatment and synthetic control groups may not be feasible across specified constraints, **microsynth** includes checks for feasibility and additional machinery that searches for the closest possible match when an exact match is not feasible.

2. The **microsynth** package also includes multiple modes for making statistical inferences with regards to the effect of the intervention. The presence of multiple treated areas enables the analyst to assess uncertainty in the estimate of the intervention effect through survey methods including linearization and a jackknife. Furthermore, granular data permit inference through permutations, which is a generalization the placebo method described in Abadie and Gardeazabal (2003); Abadie *et al.* (2010, 2015). Specifically, **microsynth** can generate a myriad of permuted treatment units using random permutations of the data units. This allows estimated effects from the actual treatment unit to be compared to effects for the permuted treatment units, which facilitates inferences.

3. For each of the three methods of inference, **microsynth** gives confidence intervals for the effect of treatment, which is enumerated as a percent reduction in outcome, and $p$ values are provided that assess the statistical significance of the estimated treatment effect. Since the jackknife and permutation methods can be computationally intensive, **microsynth** enables the user to specify whether or not each should be run. Since granular data afford the analyst the capability to study multiple outcomes simultaneously, **microsynth** can also calculate an omnibus statistic for each inference method to jointly assess the statistical significance across multiple variables.

4. For any time variant outcome of interest, **microsynth** can be used to create time series plots that compare outcome levels of treatment to synthetic control both before and after the intervention. The sampling distribution of the effects from permuted treatment region is included in the plots (if permutations are used) to help the user visualize the statistical significance of the intervention effect. The user also has the option of creating

an **Excel** (or CSV) file that contains results in the form of estimated intervention effects for each outcome along with $p$ values and confidence intervals.

5. The **microsynth** package includes parameters to assist users in finding feasible models when a plethora of matching variables and a scarcity of data make the calculation of satisfactory weights difficult. Users may call `check.feas` or `use.backup` to call on more computationally-intensive methods to calculate weights. Alternately, difficult-to-match variables may be passed to `match.out.min`/`match.covar.min` so as to seek weights that deliver the best-possible but not necessarily exact match on those variables.

6. The **microsynth** package can also be deployed on the **Synth**-like case of a single treatment with a limited number of untreated cases, although the relative dearth of data should be expected to decrease matching performance and limit the usefulness of the features discussed above.

7. Lastly, **microsynth** can be used to calculate propensity score-type weights in the vein of Hainmueller (2012) for cross-sectional data. In this case, plots should not be generated, but inferences can be tabulated using any of the three methods described earlier.

Most of the capabilities of **microsynth** can be executed through the function `microsynth()`, which returns an object of the class '`microsynth`', which contains weights, results, and the data used for plotting. A separate function, `plot_microsynth()` (which uses a '`microsynth`' object as input) is used to create plots. Package **microsynth** is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=microsynth`.

The article continues as follows. Section 2 reviews the theory underpinning synthetic controls as it pertains to this work, Section 3 describes the implementation of **microsynth**, and Section 4 provides several examples outlining the application of **microsynth** to data from a drug market intervention in Seattle, Washington. We conclude with discussion points in Section 5.

## 2. SCM with micro- and meso-level data

Here, we outline SCMs for micro-level data as described within Robbins *et al.* (2017), and we borrow the notation therein. Note that a bare-bones description of the theory that underpins the methods is provided here – see Robbins *et al.* (2017) for more details, including an outline of the relevant potential outcomes framework.

We assume there are data on $J$ total cases (or units, regions, etc.), which are indexed so that the first $J_0$ are untreated and the final $J - J_0$ are treated. In addition, there are $I$ time-varying outcomes measured across $T$ total time periods (the first $T_0$ of which are pre-intervention and the last $T - T_0$ of which are post-intervention), where $Y_{itj}$ is used to represent the value of outcome $i$ at time $t$ for unit $j$. A general data generating mechanism is used for $Y_{itj}$ (see Formula 8 in Robbins *et al.* 2017). Furthermore, there are $L$ total time-invariant (i.e., baseline) covariates where $R_{\ell j}$ denotes the value of covariate $\ell$ for unit $j$. (We recommend that any time-varying covariates be included in the procedure as outcomes.)

A synthetic control group is calculated by assigning a weight to each non-treated case. These weights are denoted $w_j$ for $j \in (1, \ldots, J_0)$. The **microsynth** algorithm matches treatment and synthetic control by calculating weights that satisfy three classes of constraints. First, the

sum of the weights equals the number of cases in the treatment area. That is,

$$\sum_{j=1}^{J_0} w_j = J - J_0. \tag{1}$$

Second, the aggregated (weighted) synthetic control matches the aggregated treatment area across the covariates. Specifically,

$$\sum_{j=1}^{J_0} w_j R_{\ell j} = \sum_{j=J_0+1}^{J} R_{\ell j} \qquad \text{for } \ell \in (1, \ldots, L). \tag{2}$$

Lastly, the synthetic control and treatment also match across all pre-intervention time points of each outcome in that

$$\sum_{j=1}^{J_0} w_j Y_{itj} = \sum_{j=J_0+1}^{J} Y_{itj} \qquad \text{for all } i \in (1, \ldots, I) \text{ and } t \in (1, \ldots, T_0). \tag{3}$$

Note that there are $1 + L + IT_0$ constraints that must be satisfied according to (1)–(3). If $w_j$ for $j \in (1, \ldots, J_0)$ meet these constraints, the cumulative treatment effect $\widehat{\alpha}_i$ for outcome $i$, is determined by first calculating the cumulative outcome value across all post-intervention time periods and all treated units. The result is then subtracted from the corresponding value for the synthetic control. That is, we calculate

$$\widehat{\alpha}_i = \sum_{t=T_0+1}^{T} \left( \sum_{j=J_0+1}^{J} Y_{itj} - \sum_{j=1}^{J_0} w_j Y_{itj} \right). \tag{4}$$

Note that this is an aggregate analogue of an average treatment effect for the treated (ATT) estimator (in the vein of Imbens 2004) of the treatment effect parameter outlined in Robbins *et al.* (2017).

Techniques for statistical inference, as described in Section 3.3, are used to determine whether or not $\widehat{\alpha}_i$ is statistically different from zero for each outcome $i$. Section 3.3 also includes discussion of an omnibus test which jointly assesses whether the estimated treatment effects for all outcomes are non-zero. The raw value of $\widehat{\alpha}_i$ is not helpful for interpreting the efficacy of the intervention. Therefore, we examine the estimated intervention effect as a relative change in percentage terms:

$$\widehat{\Delta}_i = 100 \sum_{t=T_0+1}^{T} \left( \sum_{j=J_0+1}^{J} Y_{itj} - \sum_{j=1}^{J_0} w_j Y_{itj} \right) / \sum_{t=T_0+1}^{T} \sum_{j=1}^{J_0} w_j Y_{itj}. \tag{5}$$

The **microsynth** package can be applied within cross-sectional data (in which case $T = 1$) to evaluate an intervention. In this case, only constraints (1) and (2) must be satisfied and the treatment effect is estimated using

$$\widehat{\alpha}_i = \sum_{j=J_0+1}^{J} Y_{ij} - \sum_{j=1}^{J_0} w_j Y_{ij},$$

where the time index has been suppressed. This effectively implements the entropy balancing procedure of Hainmueller (2012). See Section 4.5 for an illustration of the use of `microsynth` for entropy balancing in cross-sectional data.

# 3. Implementing microsynth

The bulk the functionality of **microsynth** is implemented within the `microsynth()` function. This function performs each of the subroutines of the **microsynth** algorithm, which are divided into two primary phases: (1) calculation of weights, and (2) plotting of treatment effects. `microsynth()` is used to call each of these subroutines (though they can also be performed separately by a modified `microsynth()` call); ergo, the function has a large number of inputs. Time series plots are created with the function `plot_microsynth()`.

## 3.1. Data statements

The `microsynth()` algorithm requires specification of the following inputs: `data`, `idvar`, `intvar`, `timevar`. At the minimum, `data` is a $TJ \times (L + I + 3)$ data array, i.e., a longitudinal dataset in the tall format (one row per case per time period). Missing values are not permitted in `data` (see Section 5 for discussion on this issue). Furthermore, `idvar`, `intvar`, and `timevar` indicate column names of `data` that distinguish the case IDs, indicators of intervention cases, and time period, respectively. The column specified by `intvar` should be binary in that any data unit that has an entry of one for `intvar` at any time is considered to be in the treatment group. If `data` is cross sectional (one row per ID), in which case **microsynth** will calculate propensity score-type weights, `timevar` should not be specified.

The `microsynth()` function also has three inputs that are used to distinguish time points that are relevant to intervention assessment: `start.pre`, `end.pre`, and `end.post`. Specifically, `start.pre` is the earliest time point at which pre-intervention outcomes will be examined, `end.pre` is the final time point of the pre-intervention period, and `end.post` is the last time point when post-intervention outcomes will be considered. Note that `end.pre` is the last time at which treatment and synthetic control will be matched to one another. All time points following `end.pre` are considered to be post-intervention and the behavior of outcomes will be compared between the treatment and synthetic control groups across those time periods. If `end.pre` is not specified by the user, **microsynth** will begin the post-intervention period at the time that corresponds to the first non-zero entry in the column indicated by `intvar`. In order to evaluate outcomes across multiple follow-up periods, `end.post` can be an (ordered increasing) vector of post-intervention time periods.

## 3.2. Creating weights

Fundamental to the calculation of weights within `microsynth()` is the specification of which variables to include as constraints given by (2) and (3). The input `match.covar`, a vector of variable names that indicate time-invariant columns of `data` (or a subset of such columns), is used to specify the constraints in (2). Likewise, `match.out` is a vector of variables names that indicate time-variant columns of `data` that define the constraints in (3). Specifically, each pre-intervention time point of each variable in `match.out` is used to build a constraint of the form in (3) (where pre-intervention time points are defined as those greater than or equal to `start.pre` and less than or equal to `end.pre`).

*Feasibility*

Note that **Synth** is built on the presumption that it is not feasible to exactly satisfy the constraints that are used to match treatment and synthetic control. Although settings involving micro-level data are presumed to involve a substantial number of untreated cases and are consequentially more conducive to there existing a set of weights that satisfies all constraints, **microsynth** is designed to handle circumstances in which there is no feasible solution to the constraints. If the user has set `check.feas = TRUE`, `microsynth()` will check for the existence of a feasible solution prior to calculating weights. If a solution does not exist, `microsynth()` partitions constraints into two classes:

1. *Exact*: Constraints that are to be exactly satisfied.

2. *Proximate*: Constraints that are to be approximately satisfied.

Weights are calculated so as to satisfy the first class of constraints while minimizing the degree to which the second class is not satisfied.

The set of constraints specified by `match.covar` and `match.out` defines the first (or primary) model for weighting, wherein they are treated as exact constraints. When this model is not feasible, `microsynth()` uses backup models that reclassify constraints as follows. In the second model (first backup), all constraints of the form in (1) and (2) are maintained as exact constraints, whereas all outcomes-based constraints of the form in (3) are categorized as proximate constraints. Furthermore, new constraints are considered wherein each outcome must align for treatment and synthetic control when aggregated across all pre-intervention time periods. Specifically,

$$\sum_{t=1}^{T_0}\sum_{j=1}^{J_0} w_j Y_{itj} = \sum_{t=1}^{T_0}\sum_{j=J_0+1}^{J} Y_{itj}, \tag{6}$$

for $i \in (1, \ldots, I)$; these are categorized as exact constraints. That is, outcome variables are aggregated across the pre-intervention time period and then treatment is matched to synthetic control across the aggregated versions.

If there is no solution to the exact constraints imposed by the first backup, a second backup is considered. Therein, (1) is categorized as an exact constraint, whereas (2) and (3) are deemed proximate constraints. This model will always be feasible (e.g., setting $w_j = (J - J_0)/J_0$ for all $j \in (1, \ldots, J_0)$ will satisfy (1)).

When there are no proximate constraints involved, the function `calibrate()` from the R package **survey** (Lumley 2011) is used to find weights. This function employs calibration techniques (Deville and Särndal 1992; Särndal 2007). The calibration algorithm assumes that weights that satisfy all constraints exist. If proximate constraints are required, `microsynth()` will use the R package **LowRankQP** (Ormerod and Wand 2020) to find weights.

To elaborate on the technicalities, let $\mathbf{x}_j$ denote a vector that embodies all variables defining the constraints given by (1)–(3) for region $j$. That is,

$$\mathbf{x}_j = (1, Y_{1j1}, \ldots, Y_{1jT_0}, Y_{2j1}, \ldots, Y_{2jT_0}, \ldots, Y_{Ij1}, \ldots, Y_{IjT_0}, \mathbf{R}_j^\top)^\top,$$

which is a $(1 + L + IT_0)$-length vector, where $^\top$ indicates a matrix transpose. Additionally, the target totals for the treatment region are defined by $\mathbf{t}_x = \sum_{j=J_0+1}^{J} \mathbf{x}_j$. If we further

define $\mathbf{w} = (w_1, \ldots, w_{J_0})^\top$ and $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{J_0})^\top$, all constraints given by (1)–(3) are encapsulated in the expression $\mathbf{X}^\top \mathbf{w} = \mathbf{t}_x$. If a solution to this expression is feasible, the calibration technique is used and will return (calibrated) weights that satisfy all constraints while minimizing the discrepancy between the calibrated weights and initial weights (which are set as equaling $(J - J_0)/J_0$ for all cases) subject to a distance metric. We recommend setting `calfun = "linear"` (which specifies that a linear distance metric of the form $G(x) = (1/2)(x-1)^2$ is used) and `bounds = c(0, Inf)`, in which case the calibrated weights will also minimize the Kish approximation of design effect (Kish 1965) to the degree possible. (The design effect is defined as the inflation in variance due to weighting of the treatment effect estimators.) When the inputs are specified in this manner, the calibrated weights are the solution to the following quadratic program:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \mathbf{w}^\top \mathbf{w} \\ \text{subject to} \quad & \mathbf{X}^\top \mathbf{w} = \mathbf{t}_x, \\ & \mathbf{w} \geq \mathbf{0}. \end{aligned}$$

If a solution to $\mathbf{X}^\top \mathbf{w} = \mathbf{t}_x$ with $\mathbf{w} \geq \mathbf{0}$ does not exist, proximate constraints must be invoked. In this case, weights are found (using **LowRankQP**) that minimize the degree to which proximate constraints are not satisfied while ensuring that the exact constraints are satisfied. To elaborate, define the matrix $\widetilde{\mathbf{X}}$ and vector $\tilde{\mathbf{t}}_x$ so as to contain the columns of $\mathbf{X}$ and elements of $\mathbf{t}_x$, respectively, that correspond to the proximate constraints. Likewise, let $\mathbf{X}^*$ and $\mathbf{t}_x^*$ denote analogues of $\widetilde{\mathbf{X}}$ and $\tilde{\mathbf{t}}_x$ that correspond to the exact constraints. Therefore, **LowRankQP** is used to find the solution to the following quadratic program:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & (\widetilde{\mathbf{X}}^\top \mathbf{w} - \tilde{\mathbf{t}}_x)^\top (\widetilde{\mathbf{X}}^\top \mathbf{w} - \tilde{\mathbf{t}}_x) \\ \text{subject to} \quad & (\mathbf{X}^*)^\top \mathbf{w} = \mathbf{t}_x^*, \\ & \mathbf{w} \geq \mathbf{0}. \end{aligned}$$

The `LowRankQP()` function is processor- and memory-intensive, so the user should be cautious when applying `microsynth()` in a manner that invokes proximate constraints with larger data sets. Note that the user may prevent `LowRankQP()` from being executed by setting `check.feas = FALSE` and `use.backup = FALSE`; however, this is inadvisable in the event that no feasible solution exists for the constraints in the primary model.

For added flexibility, the **microsynth** user may directly specify constraints that are to be categorized as proximate within the primary model. As noted previously, `match.covar` and `match.out` specify exact constraints. The `match.covar.min` and `match.out.min` inputs are of an analogous format that are used to specify proximate constraints.

As an alternative to re-categorizing constraints as proximate for the purpose of obtaining feasibility, the **microsynth** user may choose to aggregate some exact constraints in order to reduce the sparsity of the linear program. For instance, if data are observed quarterly, the user may decide to match treatment and synthetic control on values that have been aggregated to annual terms for a given outcome. Specifically, for an outcome $i$ and a subset of pre-intervention time periods (i.e., $\mathbf{T}^* = (t_1, t_2, , \ldots t_\tau) \subseteq (1, \ldots, T_0)$), the constraints

$$\sum_{j=1}^{J_0} w_j Y_{itj} = \sum_{j=J_0+1}^{J} Y_{itj},$$

for each $t \in \mathbf{T}^*$, are replaced with a single exact constraint of the form

$$\sum_{t \in \mathbf{T}^*} \sum_{j=1}^{J_0} w_j Y_{itj} = \sum_{t \in \mathbf{T}^*} \sum_{j=J_0+1}^{J} Y_{itj}.$$

By specifying `match.out` (and/or `match.out.min`) in a list format, the user may aggregate constraints in this manner. See the **microsynth** vignettes and the documentation for the `microsynth()` function for details.

Note that our procedure for using back-up models and/or aggregation to address infeasible primary models is akin to approaches invoked with **Synth** (wherein minimization across a linear combination of pre-intervention outcome values is a fundamental facet of the algorithm).

*Jackknife and permutation weights*

Recall that **microsynth** is designed to facilitate variance estimation in multiple manners, including two separate replications methods: jackknife and permutation. Both methods require calculation of several sets of additional weights. Within a jackknife, data units are randomly segmented into $G$ equal size and disjoint sub-samples. Let $N^{(g)}$ and $C^{(g)}$ denote the set of cases from the intervention and control groups, respectively, that are in the $g$-th sub-sample. Furthermore, the $g$-th replication group contains $N^{(-g)}$ and $C^{(-g)}$, which denote analogues of $N^{(g)}$ and $C^{(g)}$ that contain all cases not in the $g$-th sub-sample. A separate set of weights is calculated for each replication group. To determine the weights assigned to replication group $g$, we calculate $w_j^{(g)}$ for $j \in C^{(-g)}$ so that the following constraints are satisfied:

$$\sum_{j \in C^{(-g)}} w_j^{(g)} = \sum_{j \in N^{(-g)}} 1, \tag{7}$$

with

$$\sum_{j \in C^{(-g)}} w_j^{(g)} R_{\ell j} = \sum_{j \in N^{(-g)}} R_{\ell j} \tag{8}$$

for $\ell \in (1, \ldots, L)$, and

$$\sum_{j \in C^{(-g)}} w_j^{(g)} Y_{itj} = \sum_{j \in N^{(-g)}} Y_{itj} \tag{9}$$

for all $i \in (1, \ldots, I)$ and $t \in (1, \ldots, T_0)$. The above are jackknife analogues of (1)–(3). For each $j \in R^{(-g)}$, we set $w_j^{(g)} = (J - J_0)/(\sum_{j \in R^{(-g)}} 1)$. The `microsynth()` input `jack` is used to set the value of $G$. If `jack = 0`, then the jackknife is not run; otherwise $G =$ `jack`. The parameter `jack` cannot exceed the minimum of the number of treated units or the number of control units (i.e., we enforce $G \leq \min\{J_0, J - J_0\}$). Note that setting `jack = TRUE` results in $G = \min\{J_0, J - J_0\}$, which is our recommended choice for $G$.

The permutation method, which is an analogue of the placebo method of **Synth**, involves selecting a permuted treatment group of size $J - J_0$ (the size of the true treatment group) randomly from the $J$ total units. Assume that $K$ total permuted treatment groupings will be selected. Let $\tilde{N}^{(k)}$ and $\tilde{C}^{(k)}$ represent the treatment and control cases for the $k$-th permutation, respectively. As with the jackknife, weights are recalculated for each permutation grouping. Letting $\tilde{w}_j^{(k)}$ for $j \in \tilde{C}^{(k)}$ represent weights for the $k$-th permutation grouping, the

$\tilde{w}_j^{(k)}$ are selected to satisfy

$$\sum_{j \in \tilde{C}^{(k)}} \tilde{w}_j^{(k)} = \sum_{j \in \tilde{N}^{(k)}} 1, \quad \text{with} \quad \sum_{j \in \tilde{C}^{(k)}} \tilde{w}_j^{(k)} R_{\ell j} = \sum_{j \in \tilde{N}^{(k)}} R_{\ell j}$$

for $\ell \in (1, \ldots, L)$, and

$$\sum_{j \in \tilde{C}^{(k)}} \tilde{w}_j^{(k)} Y_{itj} = \sum_{j \in \tilde{N}^{(k)}} Y_{itj}.$$

The `microsynth()` input `perm` sets the value of $K$. If `perm = 0`, permutation methods are not applied; otherwise, $K =$ `perm`. Furthermore, `perm` cannot exceed the number of possible permutation groupings, i.e., $\binom{J}{J_0}$. If $\binom{J}{J_0} \leq 1000000$, steps are taken to ensure that no two permutation groupings are the same.

The processes used for calculating jackknife and permutation weights can be parallelized within **microsynth**, which is controlled via the `microsynth()` input `n.cores`; this invokes the package **parallel** (R Core Team 2020).

*Weighting output*

In all, $G + K + 1$ sets of weights are created within `microsynth()`. Following completion of a run of `microsynth()`, a '`microsynth`' object (generically labeled `ms` here) is returned. Its first element is a list (`w`), with its first element named `Weights`, which is itself a $J \times (G + K + 1)$ matrix. The first column of `ms$w$Weights` contains the main weights; the next $G$ columns contain the jackknife weights (if calculated), and the final $K$ columns contain the permutation weights (if calculated). The second component of `w` is named `Intervention`, which is a $J \times (G + K + 1)$ matrix of logical elements. Entries in `ms$w$Intervention` indicate whether the case that corresponds to a given row is considered treated for the set of weights in a given column. That is, the entries in `ms$w$Intervention` are `TRUE` for all elements corresponding to the actual treatment group in the first $G + 1$ columns and for elements that correspond to permuted treatment cases in the final $K$ columns. Columns corresponding to the jackknife in `ms$w$Intervention` may have entries of `NA`, indicating cases that have been dropped from the corresponding jackknife replication group. A balance table that illustrates the degree to which the weights satisfy the specified constraints is given in the element `ms$w$Summary`.

### 3.3. Statistical inferences

Once weights have been created, the next step in `microsynth()` involves the tabulation of results including point estimators of the treatment effect for each outcome as well as inferences regarding the statistical significance of the estimators. Results are tabulated for the variables listed in `result.var` (which by default includes all time variant variables in `data`). Note that `result.var` can include variables external to those listed in `match.out` and `match.out.min`. If `result.var` is `NULL`, no results are returned. An omnibus statistic is created if `omnibus.var` is non-`NULL`, in which case; this statistic jointly assesses the statistical significance of all variables listed in `omnibus.var`. Results may be returned in the form of an **Excel** file or CSV when `result.file`, which indicates the name of the output file, is non-`NULL`. As noted previously, statistical inference is performed by up to three methods: (1) linearization, (2) jackknife, and (3) permutation. Linearization is performed so long as `result.var` is non-`NULL`; results are tabulated for the jackknife and permutations so long

as `jack` $> 0$ and `perm` $> 0$, respectively. In addition to being returned within an **Excel** file (when `result.file` is non-`NULL`), any results that are tabulated are included as part of the '`microsynth`' object that is returned when `microsynth()` completes, and can be viewed by calling `microsynth.tab(ms)`.

*Calculation of results*

Recall that our treatment effect estimator, $\hat{\alpha}_i$, was defined in (4). However, in order to evaluate uncertainty in this quantity and perform appropriate inferences, we formulate (and fit) the outcome as a function of treatment status using a weighted linear model. Specifically, consider

$$E[Y_{ijt}] = \beta_{it} + a_i D_{jt}, \tag{10}$$

where $\beta_{it}$ is a fixed effect for time $t$ with $T_0 < t \leq T$. Further, $D_{jt}$ is a binary indicator of treatment status that is unity only if case $j$ is in the intervention group and $t$ is a post-intervention time. When this model is computed using weighted least squares (where we use the synthetic control weights $w_j$ for cases in the control group and set $w_j = 1$ for cases in the treatment group), the estimate of $a_i$ in the above expression equals the treatment effect estimate $\widehat{\alpha}_i$ from (4) (Robbins *et al.* 2017). As such, the use of the synthetic control weights in this model ensures that discrepancies (in terms of covariates and pre-intervention outcome values) between treatment and control regions are taken into account. Software for survey analysis will return an estimate of the standard error of $\widehat{\alpha}_i$ when (10) is run.

The function `svyglm()` with a design object computed with `svydesign()` in the `survey` package is used to approximate standard errors via linearization. Within this approach, estimators from generalized linear models are written as a function of weighted totals (the variance of which may be estimated through simple expressions). Taylor series approximations are then used extract the variance of the estimator. See Binder (1983) and Lumley (2011) for further details.

Similarly, the function `svyglm()` with a design object computed with `svrepdesign()` in the `survey` package is used to calculate standard errors with a jackknife. The jackknife replication weights that satisfy (7)–(9) are set as the object `repweights` within `svrepdesign()`. Let $\widehat{\alpha}_i^{(k)}$ represent the version of $\widehat{\alpha}_i$ calculated using the $k$-th set of replication weights (where again $w_j = 1$ for cases in the treatment group) with only cases in the $k$-th replication group. The jackknife variance estimator is

$$\widehat{\mathrm{Var}}(\widehat{\alpha}_i) = \frac{G-1}{G} \sum_{g=1}^{G} (\widehat{\alpha}_i^{(g)} - \widehat{\alpha}_i)^2.$$

Once $\widehat{\alpha}_i$ and its standard error have been produced for each outcome in `result.var` (using either linearization or the jackknife), basic statistical theory can be used to a determine $p$ value of a test that assesses the statistical significance of $\widehat{\alpha}_i$. That is,

$$\widehat{Z}_i = \frac{\widehat{\alpha}_i}{\sqrt{\widehat{\mathrm{Var}}(\widehat{\alpha}_i)}} \tag{11}$$

is used as a test statistic (which is assumed to have a $t$-distribution). Following application of additional Taylor series approximations, a confidence interval for $\widehat{\Delta}_i$ from (5), which is

the treatment effect written as percent change, is calculated (see Robbins *et al.* 2017, for details). The `microsynth()` inputs `test` and `confidence` specify the type of test (lower-tailed, upper-tailed, two-sided) and the confidence level (e.g., 95%), respectively. Furthermore, an omnibus test that jointly assesses the statistical significance for the outcomes in `omnibus.var` is implemented as follows. Let $\hat{\mathbf{a}} = (\widehat{\alpha}_1, \ldots, \widehat{\alpha}_I)$ indicate the vector of treatment effects for pertinent outcomes. Further, the estimated variance/covariance matrix of this vector is given by $\mathbf{C} = \text{Var}(\hat{\mathbf{a}})$. If a two-sided test is applied, the omnibus test statistic is represented $\widehat{Z}_{\text{omni}} = \hat{\mathbf{a}}^\top \mathbf{C}^{-1} \hat{\mathbf{a}}$, which is assumed to have a $\chi^2$ distribution with $I$ degrees of freedom. For details on the omnibus statistic for upper- or lower-tailed tests, see Robbins *et al.* (2017).

Likewise, permutation methods can be used to calculate $p$ values and confidence intervals. We calculate $\widehat{\alpha}_i^{(k)}$, which is a treatment effect estimator in the vein of (4) as found using the $k$-th permutation group with corresponding permutation weights, for all $k \in (1, \ldots, K)$ and each $i \in (1, \ldots, I)$. This term is standardized in the vein of (11) to yield $\widehat{Z}_i^{(k)}$. To determine $\widehat{\alpha}_i^{(k)}$ and its standard error, we estimate the parameter $a_i$ from (10) while using the $k$-th set of permutation weights and while replacing $D_{jt}$ with a corresponding treatment status indicator for the respective permutation group. The permutation-based $p$ value for outcome $i$ is set as

$$p_i = \frac{\{\#k : \widehat{Z}_i^{(k)} < \widehat{Z}_i\}}{K}$$

for lower-tailed tests,

$$p_i = \frac{\{\#k : \widehat{Z}_i^{(k)} > \widehat{Z}_i\}}{K}$$

for upper-tailed tests, and

$$p_i = 2 \min \left\{ \frac{\{\#k : \widehat{Z}_i^{(k)} < \widehat{Z}_i\}}{K}, \frac{\{\#k : \widehat{Z}_i^{(k)} > \widehat{Z}_i\}}{K} \right\} \tag{12}$$

for two-sided tests. We use (12) in lieu of an expression that involves squaring $\widehat{Z}_i$ since the former does not mandate symmetry of the statistic's sampling distribution. For upper- and lower-tailed omnibus tests, $p$ values are created by replacing $\widehat{Z}_i$ and $\widehat{Z}_i^{(k)}$ with $\widehat{Z}_{\text{omni}}$ and $\widehat{Z}_{\text{omni}}^{(k)}$ in the respective formulas above. For a two-tailed omnibus test, we use $p_{\text{omni}} = K^{-1}\{\#k : \widehat{Z}_{\text{omni}}^{(k)} > \widehat{Z}_{\text{omni}}\}$, since the omnibus test rejects only for large values. See Robbins *et al.* (2017) for a description of the manner in which a confidence interval is calculated for the treatment effect (as a percent change) using permutation methods. The calculation of $\widehat{Z}_i^{(k)}$ for all $k$ can be time-intensive, so those processes are parallelized (which is controlled using the parameter `n.cores`).

If `use.survey = FALSE`, the permutation-based $p$ values are calculated with $\widehat{\alpha}_i$ and $\widehat{\alpha}_i^{(k)}$ in place of $\widehat{Z}_i$ and $\widehat{Z}_i^{(k)}$ in the above expressions. This circumvents the need to calculate standard errors of $\widehat{\alpha}_i^{(k)}$ for each permutation group and therefore saves substantial computation time. No confidence intervals are calculated for permutation methods in this case. Note that, however, best practice is to use `use.survey = TRUE`.

### *Output of results for inference*

A '`microsynth`' object includes an element named `Results` which is a data frame that contains the results described above. The element `ms$Results` contains one row for each outcome

that is included in `out.result` and a final row that corresponds to the omnibus tests (if `omnibus.var` is non-`NULL`). The first column of `ms$Results`, which is named `Trt`, gives the aggregated outcome values across all cases in the treatment group. The second column, named `Con`, gives corresponding values for the synthetic control group. The third column, named `Pct.Chng`, gives the percent change between treatment and synthetic control (notated as $\widehat{\Delta}_i$ above and calculated as 100(`Trt - Con`)/`Con`). The next three columns give the $p$ value for a test of the statistical significance of the treatment effect as well as the upper and lower bounds of a confidence interval for $\widehat{\Delta}_i$ when founding using linearization. If `jack > 0`, the following three columns give the $p$ value and confidence interval when found using jackknife, and if `perm > 0`, the final three columns will give the corresponding information when found using permutation methods. Note that the row named omnibus will be empty for all columns other than those containing $p$ values. The user can write the contents of `ms$Results` directly to an XLSX (**Excel**) or CSV (comma-separated values) file by specifying a file name via the `microsynth()` input `result.file`.

Recall that the `microsynth()` input `end.post` can be an ordered (increasing) vector of post-intervention time points. If `end.post` has length greater than one, `ms$Results` is a list with one data frame for each element of `end.post`. Further, if an Excel file is created, it will have one tab for each value in `end.post`.

The **microsynth** package also includes `summary()` and `print()` functions. The output of the `summary()` function includes two parts: (1) a matching summary that compares characteristics of the treatment to the synthetic control and the population; and (2) estimated results, in a similar format as they appear when saved to CSV or XLSX. The `print()` function also includes the estimated results (but not the matching summary) as well as other basic information such as the number of treatment/control units, the number of constraints, and so forth.

### 3.4. Plotting

The function `plot_microsynth()` can be used to produce time series plots comparing treatment and synthetic control before and after the intervention. The primary input in this function is a parameter labeled `ms`, which is a '`microsynth`' object. Plots may only be created for variables that were included in the `microsynth()` parameters `result.var` or `omnibus.var` during creation of `ms`. Plots can be produced as output (if `plot.file = NULL`) or written directly to a PDF file, the name of which is provided in the input `plot.file`. A single PDF file will be created with three pairs of figures per page unless `sep = TRUE`, in which case a separate file is produced for each figure.

For each outcome in the `plot_microsynth()` parameter `plot.var`, two figures are created. The figures are in the same format as those provided in Robbins *et al.* (2017). The first figure charts the observed values for each outcome variable for the aggregated treatment group as well as the corresponding value for the (weighted) synthetic control group for all time points from `start.pre` to `end.post`. A horizontal red line is provided at `end.pre` to indicate the end of the pre-intervention period. This figure also shows the level of the outcome across all cases in the `microsynth()` parameter `data`, scaled by default to the number of treatment cases, or to the value specified by the parameter `scale.var` in `microsynth()`. The second plot is of a similar format but instead shows the difference between the treatment and (weighted) synthetic control groups. To help the user visualize the uncertainty in the post -intervention

treatment/control difference, the corresponding treatment/control difference is shown for the permutation groups. Note that the values of `start.pre`, `end.pre`, and `end.post` may be specified as inputs to `plot_microsynth()`; otherwise, they are taken from `ms`.

*Output for plotting*

A '`microsynth`' object includes output that is used for plotting during application of function `plot_microsynth()`. Specifically, the output of `microsynth()` includes an object named `Plot.Stats`, which contains the data that are plotted. Specifically, `ms$Plot.Stats` is a list with six elements, the first three of which (`Treatment`, `Control`, `All`) are matrices and the fourth (`Difference`) is three-dimensional array that also contains data for the permutation groups (when `perm > 0`). The final two (`end.pre` and `scale.by`)

# 4. Examples

To demonstrate functionality, we will use the `seattledmi` dataset provided with the microsynth package to evaluate a Drug Market Intervention (DMI) in Seattle, WA. The data are measured at the level of the census block with quarterly time measurements and are given in quarter-block panel format. The `seattledmi` dataset contains data on nine time-invariant demographic-type covariates (measured using Census data) and ten time-varying outcomes, each of which represents a type of crime (reported by the Seattle Police Department). There are data measured across 16 quarters, the first 12 of which are considered pre-intervention.

```
R> library("microsynth")
R> data("seattledmi", package = "microsynth")
```

The DMI, which is an established crime-reduction mechanism (see Saunders *et al.* 2017, for details), was applied to 39 blocks in the International District in Seattle. The remaining 9,603 Seattle blocks are potential comparison units from which the synthetic control may be constructed. This data include variables `"ID"` to indicate observation units, `"time"` to indicate the respective time period, and `"Intervention"`, which is a binary variable with 0 for all untreated groups and the treated groups during the pre-intervention period and a 1 for treated groups at the time of intervention and later.

## 4.1. Example 1: A feasible first model with four outcomes

First, we will estimate the effect of the DMI on the incidences of four types of crime: felony arrests, misdemeanor arrests, drug arrests, and any criminal arrest. That is, we set

```
R> match.out <- c("i_felony", "i_misdemea", "i_drugs", "any_crime")
```

We also use all available time invariant covariates:

```
R> cov.var <- c("TotalPop", "BLACK", "HISPANIC", "Males_1521", "HOUSEHOLDS",
+    "FAMILYHOUS", "FEMALE_HOU", "RENTER_HOU", "VACANT_HOU")
```

Passing these vectors to the inputs `match.out` and `match.covar`, respectively, instructs microsynth to calculate weights that provide exact matches on these variables. Setting

`result.var = match.out` and `plot.var = match.out` ensures that time series plots and results will be provided for each of the four outcomes noted above. Lower-tailed tests (`test = "lower"`) are used since the DMI will, in theory, cause a reduction in crime levels. , since by default `match.out = result.var`, these variables may be passed directly to `result.var` so long as they are the only time-variant outcome variables for which we would like to compute estimates and plots. Setting `start.pre = 1`, `end.pre = 12`, and `end.post = 16` ensures that the first twelve time periods will constitute the pre-intervention period and time periods 13–16 will constitute the post-intervention period. We set `jack = TRUE`, which runs the jackknife with 39 groups (which equals the number of blocks in the treatment region), and `perm = 250` to create 250 placebo groups via permutation.

A '`microsynth`' object named `sea1` is created by executing the following code:

```
R> sea1 <- microsynth(seattledmi, idvar = "ID", timevar = "time",
+    intvar = "Intervention", start.pre = 1, end.pre = 12, end.post = 16,
+    match.out = match.out, match.covar = cov.var, result.var = match.out,
+    jack = TRUE, perm = 250, test = "lower", n.cores = 1)
```

Upon completion of the algorithm, one can use the `summary()` function to assess the results.

```
R> summary(sea1)
```

First, a balance table is given (with several rows omitted here) that is used to evaluate the matching of treatment and control.

```
Weight Balance Table:
              Targets Weighted.Control All.scaled
Intercept          39               39    39.0000
TotalPop         2994             2994  2384.7477
BLACK             173              173   190.5224
HISPANIC          149              149   159.2682
Males_1521         49               49    97.3746
HOUSEHOLDS       1968             1968  1113.5588
FAMILYHOUS        519              519   475.1876
FEMALE_HOU        101              101    81.1549
RENTER_HOU       1868             1868   581.9340
VACANT_HOU        160              160    98.4222
i_felony.12        14               14     4.9023
i_felony.11        11               11     4.6313
i_felony.10         9                9     3.0741
i_felony.9          5                5     3.2642
i_felony.8         20               20     4.4331
...               ...              ...        ...
any_crime.5       270              270    50.6531
any_crime.4       250              250    57.2946
any_crime.3       236              236    58.8681
any_crime.2       250              250    51.5429
any_crime.1       242              242    55.1145
```

From the table, it is confirmed that all constraints are satisfied (as the entries in the column labeled `Targets` equal those in the column labeled `Weighted.Control`). Note that the first row (`Intercept`) indicates that there are 39 blocks in the treatment area and that the weights for the synthetic control sum to that value. Note also that the final column gives the corresponding values when calculated across the entire dataset (scaled so to match the treatment region on the basis of the intercept). Note that row names that correspond to time varying outcomes are appended with the corresponding time period (e.g., `i_felony.12` is the number of felonies observed in the 12-th time period).

In addition, the summary function provides post-intervention inferences in tabulated form, as shown below. (These may be also be viewed by calling `sea1$Results`.)

```
Results:

end.post = 16
           Trt    Con Pct.Chng Linear.pVal Linear.Lower Linear.Upper
i_felony    46  68.22   -32.6%      0.0109       -50.3%        -8.4%
i_misdemea  45  71.80   -37.3%      0.0019       -52.8%       -16.7%
i_drugs     20  23.76   -15.8%      0.2559       -46.4%        32.1%
any_crime  788 986.43   -20.1%      0.0146       -32.9%        -4.9%
Omnibus     --     --       --      0.0010           --           --
           Jack.pVal Jack.Lower Jack.Upper Perm.pVal Perm.Lower Perm.Upper
i_felony      0.0650     -54.6%       0.1%    0.0200     -50.9%     -12.9%
i_misdemea    0.0725     -61.0%       0.7%    0.0080     -53.8%     -17.3%
i_drugs       0.3190     -53.0%      50.9%    0.3040     -46.8%      22.2%
any_crime     0.0440     -35.3%      -1.3%    0.0200     -30.8%      -7.7%
Omnibus       0.0495         --         --    0.0080         --         --
```

The results table gives the levels of the outcome when aggregated across the entire post-intervention period for the treatment (`Trt`) and synthetic control (`Con`) groups. The estimated treatment effect is given by the percent change which is calculated as `Pct.Chng` = 100(`Trt`-`Con`)/`Con`. In subsequent columns, $p$ values for the statistical significance of the estimated treatment effect and a confidence interval (where the magnitude of confidence is given by the parameter `confidence`) are provided for each outcome for all three methods of variance estimation. $p$ values of the omnibus statistic are given in the final row of the table. In this Example, we see evidence that the DMI had a statistically significant effect on crime levels, which is manifested in the number of felonies, misdemeanors, and total crimes (but not drug crimes) that were reported in the post-intervention period.

Note that that the estimated variance is smaller when linearization is used than when the jackknife is used. This observation is in line with prior research involving calibration (Kott 2006; Robbins, Ghosh-Dastidar, and Ramchand 2020). See Section 5 for more discussion.

The `print()` function provides additional summary information:

```
R> print(sea1)

Scope:
        Units:               Total: 9642     Treated: 39     Untreated: 9603
        Study Period(s):     Pre-period: 1 - 12      Post-period: 13 - 16
        Constraints:         Exact Match: 58         Minimized Distance: 0
```

```
Time-variant outcomes:
        Exact Match: i_felony, i_misdemea, i_drugs, any_crime (4)
        Minimized Distance: (0)
Time-invariant covariates:
        Exact Match: TotalPop, BLACK, HISPANIC, Males_1521, HOUSEHOLDS,
            FAMILYHOUS, FEMALE_HOU, RENTER_HOU, VACANT_HOU (9)
        Minimized Distance: (0)
```

Note that the balance table shown above can be extracted using the element `sea1$w$Summary`. Via the **xtable** package (Dahl, Scott, Roosen, and Magnusson 2019), for example, one may output this table into a LaTeX document:

```
R> library("xtable")
R> print(xtable(sea1$w$Summary, type = "latex",
+    tabular.environment = "longtable"), file = "Ex1wSummary.tex")
```

Furthermore, the `plot_microsynth()` function may be used to produce time series plots of the outcomes.

```
R> plot_microsynth(sea1)
```

The plots are shown in Figure 1 for this example. The plots indicate that treatment is exactly matched to synthetic control during pre-intervention time periods. Further, the observed levels of crime are lower than they would have been in the absence of treatment for all outcomes except for drug crimes (`i_drugs`). Since the difference in treatment and synthetic control extends beyond that seen within the placebo regions, there are indications that the effect is statistically significant.

### 4.2. Example 2: Infeasible first model

Next, we augment the number of outcomes that will be considered to include counts for a total of nine types of crimes (i.e., robberies, aggravated assaults, burglaries, larcenies, felonies, misdemeanors, drug sales, drug possessions, and total crimes).

```
R> match.out <- c("i_robbery", "i_aggassau", "i_burglary", "i_larceny",
+    "i_felony", "i_misdemea", "i_drugsale", "i_drugposs", "any_crime")
```

However, the extra outcome variables implies that there are 60 additional constraints for matching. As such, the added sparsity decreases the likelihood that a set of weights exists that exactly match treatment and synthetic control. To allow the possibility that a feasible solution to all constraints does not exist, we set `check.feas = TRUE` and `use.backup = TRUE`.

```
R> sea2 <- microsynth(seattledmi, idvar = "ID", timevar = "time",
+    intvar = "Intervention", start.pre = 1, end.pre = 12, end.post = 16,
+    match.out = match.out, match.covar = cov.var, result.var = match.out,
+    jack = TRUE, perm = 250, check.feas = TRUE, use.backup = TRUE,
+    test = "lower", n.cores = 1)
```

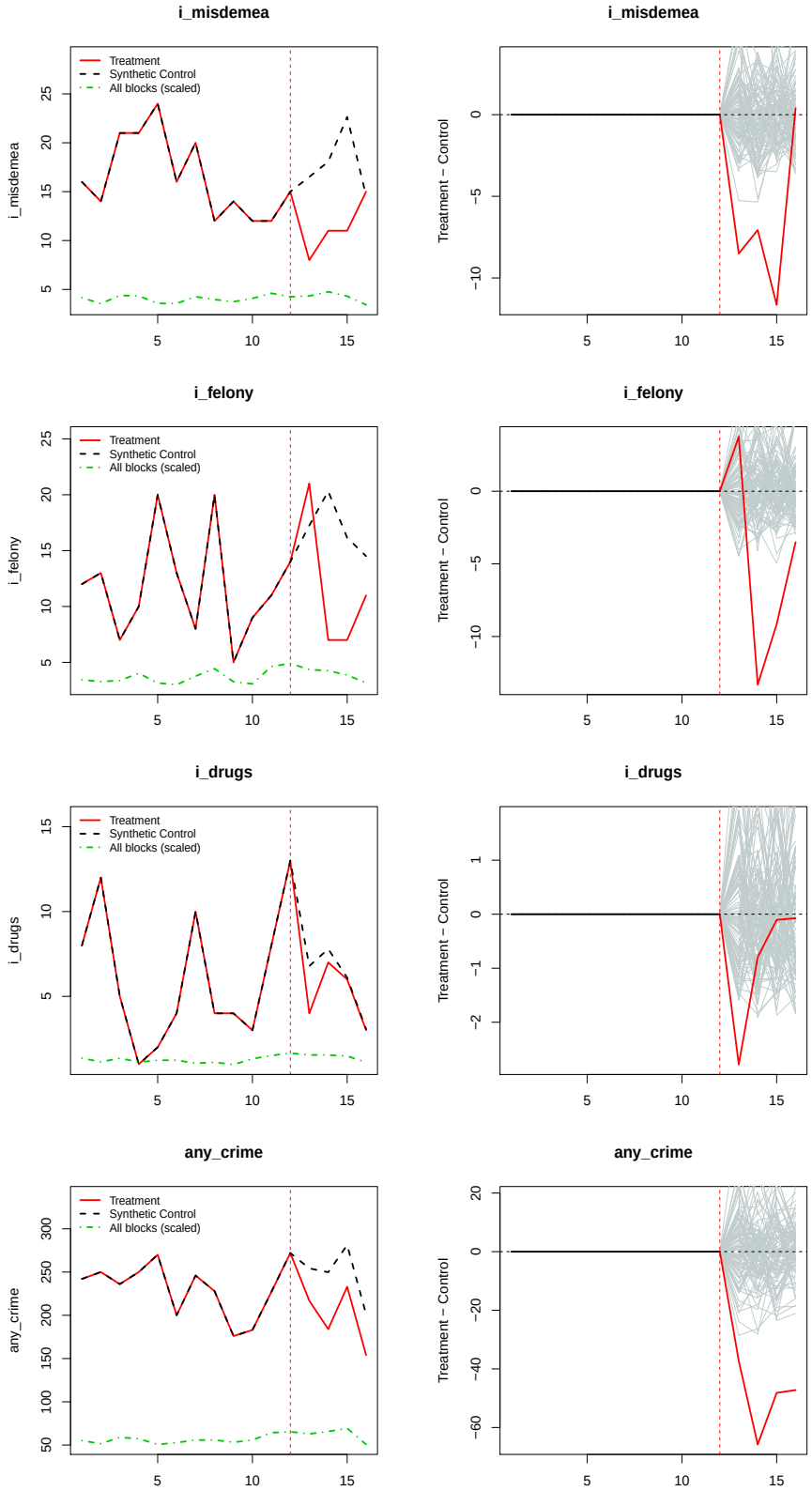


Figure 1: Time series plots for Example 1. Plots on the right give the difference between treatment and synthetic control with permutation groups plotted in gray. The $x$-axis indicates the time period (1–16).

In this example, the constraints indicated by the specified model do not yield a feasible solution. As such, the algorithm uses the second model (first backup) for matching as described in Section 3.2, for which a feasible solution does exist. The **microsynth** package provides this information to the user by printing the following output as the algorithm is being run:

```
Checking feasibility of first model...
First model is infeasible.
Checking feasibility of second model...
Second model is feasible.
```

The summary balance table for this example is shown below. Each outcome was aggregated across the 12 pre-intervention time periods (see `i_robbery.1.12`, for example) and matched exactly on the resulting aggregated outcomes. Furthermore, the constraints given by intercept and covariates are matched exactly. However, the remaining constraints (i.e., the outcomes at each specific time period) are matched as closely as possible.

```
R> summary(sea2)

Weight Balance Table:
               Targets Final.Weighted.Control All.scaled
Intercept           39                39.0000    39.0000
TotalPop          2994              2994.0000  2384.7477
...                ...                   ...        ...
RENTER_HOU        1868              1868.0000   581.9340
VACANT_HOU         160               160.0000    98.4222
i_robbery.1.12      68                68.0000    15.6938
i_aggassau.1.12     43                43.0000    12.0737
i_burglary.1.12    805               805.0000   193.3740
i_larceny.1.12     486               486.0000   121.4250
i_felony.1.12      142               142.0000    44.3351
i_misdemea.1.12    197               197.0000    48.4284
i_drugsale.1.12     25                25.0000     5.2057
i_drugposs.1.12     49                49.0000     9.8693
any_crime.1.12    2780              2780.0000   676.4328
i_robbery.12        12                11.3780     1.7352
i_robbery.11         9                 8.1200     1.6058
i_robbery.10         4                 3.8790     1.1892
i_robbery.9          1                 1.4904     1.1325
i_robbery.8          7                 6.3736     1.2620
...                ...                   ...        ...
any_crime.5        270               261.3992    50.6531
any_crime.4        250               243.6447    57.2946
any_crime.3        236               237.5359    58.8681
any_crime.2        250               253.0278    51.5429
any_crime.1        242               239.5277    55.1145
```

The above observations are confirmed by the results summary listed below. Many outcomes (though not all) observe a statistically significant effect of the intervention (except for when

the jackknife is used, which is a consequence of the increased variance that results from the jackknife). Further, the inclusion of additional outcomes causes the omnibus test to no longer result in a significant finding.

```
Results:

end.post = 16
           Trt    Con Pct.Chng Linear.pVal Linear.Lower Linear.Upper
i_robbery   11  21.49   -48.8%      0.0119       -70.2%       -12.0%
i_aggassau  12  16.46   -27.1%      0.1620       -58.5%        28.2%
i_burglary 245 294.10   -16.7%      0.0857       -33.2%         3.9%
i_larceny  145 165.45   -12.4%      0.1754       -30.5%        10.5%
i_felony    46  51.74   -11.1%      0.2788       -36.4%        24.2%
i_misdemea  45  62.75   -28.3%      0.0387       -47.5%        -2.0%
i_drugsale  11   4.22   160.9%      0.9482        23.6%       450.7%
i_drugposs   9  17.15   -47.5%      0.0225       -71.1%        -4.7%
any_crime  788 921.55   -14.5%      0.0963       -29.8%         4.1%
Omnibus     --     --       --      0.0149           --           --
           Jack.pVal Jack.Lower Jack.Upper Perm.pVal Perm.Lower Perm.Upper
i_robbery     0.1708     -81.5%      41.9%    0.0320     -71.7%     -20.4%
i_aggassau    0.2427     -63.6%      46.2%    0.1720     -61.4%      13.3%
i_burglary    0.1575     -37.6%      11.3%    0.0520     -33.0%      -0.5%
i_larceny     0.2283     -34.1%      16.5%    0.1160     -30.9%       5.5%
i_felony      0.4081     -59.9%      97.1%    0.2280     -39.4%      16.6%
i_misdemea    0.2011     -60.7%      30.9%    0.0480     -49.4%      -3.9%
i_drugsale    0.7891     -86.5%    4923.2%    0.9840       2.0%     296.3%
i_drugposs    0.1178     -76.5%      16.9%    0.0440     -73.6%     -17.0%
any_crime     0.1879     -35.8%      13.8%    0.0520     -29.9%       0.1%
Omnibus       0.1225         --         --    0.0480         --         --
```

Also, the example above illustrates that confidence intervals and $p$ values may yield contradicting findings when permutation is used for variance estimation. For instance, with the outcome `i_drugposs`, the $p$ value does not indicate statistical significance, but the confidence interval does not contain zero. This issue is a consequence of the confidence intervals being determined using approximation techniques and is exacerbated when considering outcomes that have low prevalence.

Time series plots for four of the outcomes in this example are shown in Figure 2.

```
R> plot_microsynth(sea2)
```

From the plots, we see an indication that the intervention may have increased the number of reported drug sales but decreased the number of drug possessions. This may be a consequence of the DMI resulting in a crackdown on drug sales and may also explain the lack of an effect of the intervention on total drug crimes as seen in Example 1. The plots also indicate the presence of an intervention effect on other outcomes, but the effect does not appear as marked as in the prior example.

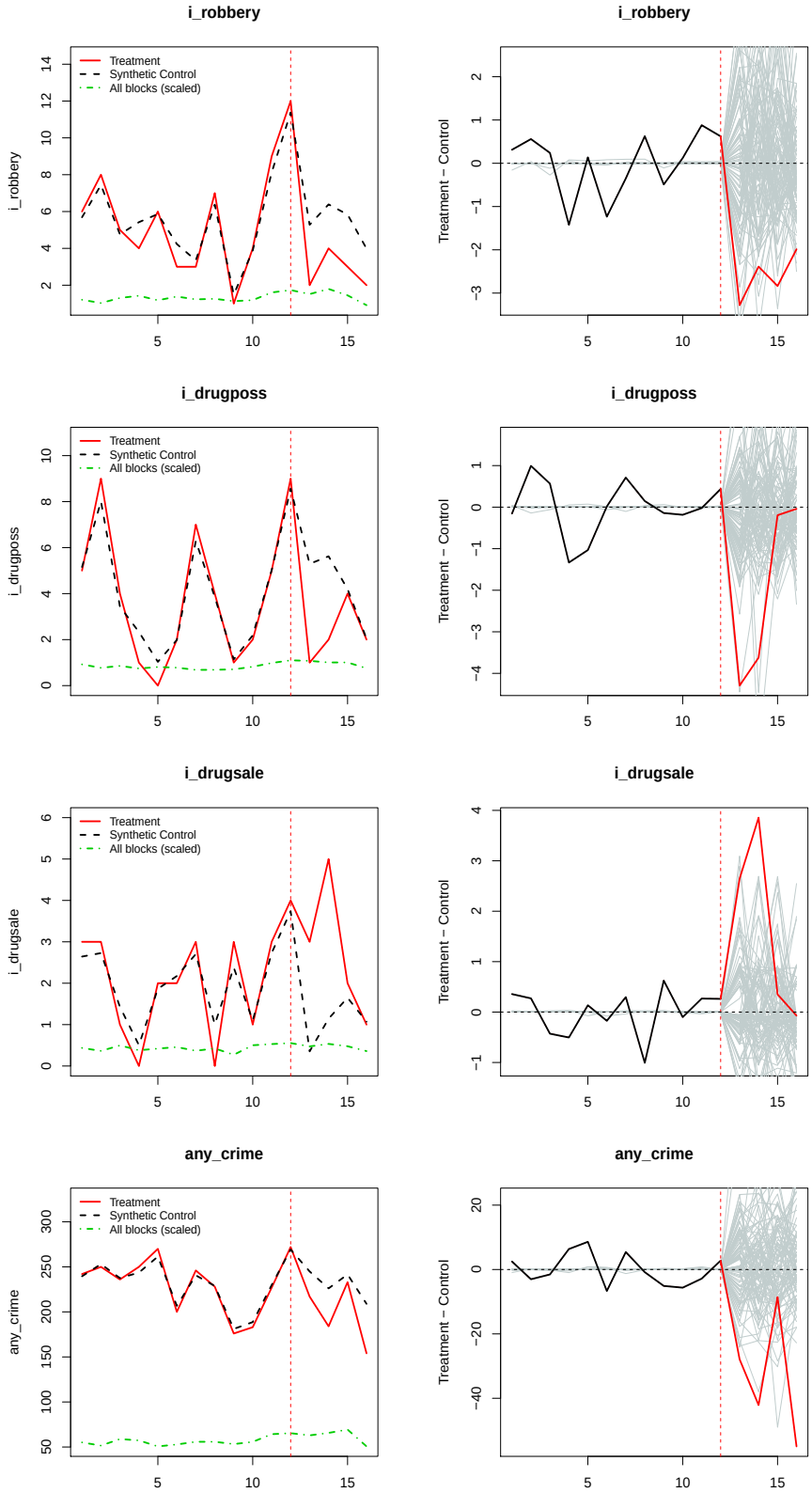


Figure 2: Time series plots for Example 2. Plots on the right give the difference between treatment and synthetic control with permutation groups plotted in gray. The $x$-axis indicates the time period (1–16).

### 4.3. Example 3: `match.out` as a list

As an alternative to using a built-in backup model to obtain feasibility when the initial set of constraints is infeasible, one may specify `match.out` in a list format in order to aggregate specific outcomes across specific time periods and thereby reduce the number of constraints needed.

```
R> match.out <- list("i_robbery" = rep(2, 6), "i_aggassau" = rep(2, 6),
+    "i_burglary" = rep(1, 12), "i_larceny" = rep(1, 12),
+    "i_felony" = rep(2, 6), "i_misdemea" = rep(2, 6),
+    "i_drugsale" = rep(4, 3), "i_drugposs" = rep(4, 3),
+    "any_crime" = rep(1, 12))
```

This list tells `microsynth()` which variables to use for matching and which time periods to aggregate for each variable. Variables with higher prevalence are kept as quarterly counts whereas more sparse variables are aggregated to biannual or annual frequency. Specifically, quarterly versions of `i_burglary`, `i_larceny`, and `any_crime` are used, whereas biannual versions of `i_robbery`, `i_aggassau`, `i_felony`, and `i_misdemea` and annual versions of `i_drugsale`, and `i_drugposs` are used.

Note that when running `microsynth()` with `match.out` in list format, one should not feed `match.out` as defined above into parameters such as `result.var` (`names(match.out)` could be used instead).

The call to `microsynth()` in this example is:

```
R> sea3 <- microsynth(seattledmi, idvar = "ID", timevar = "time",
+    intvar = "Intervention", start.pre = 1, end.pre = 12, end.post = 16,
+    match.out = match.out, match.covar = cov.var,
+    result.var = names(match.out), jack = TRUE, perm = 250,
+    test = "lower", n.cores = 1)
```

In the summary balance table below, we can see that all constraints are satisfied.

```
R> summary(sea3)

Weight Balance Table:
                Targets Weighted.Control All.scaled
Intercept            39               39    39.0000
TotalPop           2994             2994  2384.7477
...                 ...              ...       ...
RENTER_HOU         1868             1868   581.9340
VACANT_HOU          160              160    98.4222
i_robbery.11.12      21               21     3.3410
i_robbery.9.10        5                5     2.3217
i_robbery.7.8        10               10     2.4916
i_robbery.5.6         9                9     2.5644
...                 ...              ...       ...
i_drugposs.9.12      17               17     3.6120
```

```
i_drugposs.5.8        13              13     2.9608
i_drugposs.1.4        19              19     3.2965
any_crime.12         272             272    65.3398
any_crime.11         227             227    64.2396
...                  ...             ...        ...
any_crime.2          250             250    51.5429
any_crime.1          242             242    55.1145
```

In the results for Example 3, shown below, we see indications of a statistically significant treatment effect for a handful of outcomes when linearization or permutation is used for variance estimation.

```
Results:

end.post = 16
           Trt    Con Pct.Chng Linear.pVal Linear.Lower Linear.Upper
i_robbery   11  18.60   -40.9%      0.0426       -65.8%         2.3%
i_aggassau  12  16.64   -27.9%      0.1593       -59.3%        27.9%
i_burglary 245 314.59   -22.1%      0.0217       -36.7%        -4.1%
i_larceny  145 168.76   -14.1%      0.1303       -31.2%         7.3%
i_felony    46  52.83   -12.9%      0.2555       -38.6%        23.5%
i_misdemea  45  56.78   -20.7%      0.1134       -42.4%         9.0%
i_drugsale  11   6.13    79.5%      0.8643       -18.1%       293.3%
i_drugposs   9  15.46   -41.8%      0.0619       -68.8%         8.5%
any_crime  788 961.63   -18.1%      0.0405       -32.1%        -1.1%
Omnibus     --     --       --      0.0730           --           --
           Jack.pVal Jack.Lower Jack.Upper Perm.pVal Perm.Lower Perm.Upper
i_robbery     0.2735     -82.1%      95.3%    0.0800     -65.4%      -6.3%
i_aggassau    0.3714     -83.0%     206.4%    0.2280     -59.7%      12.5%
i_burglary    0.2307     -52.8%      28.6%    0.0080     -34.7%      -9.1%
i_larceny     0.3387     -50.9%      50.3%    0.1440     -30.1%       7.0%
i_felony      0.3835     -57.9%      80.0%    0.3040     -37.9%      23.6%
i_misdemea    0.3124     -61.3%      62.3%    0.1440     -41.7%       5.0%
i_drugsale    0.6988     -81.7%    1659.9%    0.9400     -20.3%     159.3%
i_drugposs    0.2653     -81.6%      83.9%    0.1560     -65.2%      -8.0%
any_crime     0.2658     -49.2%      32.3%    0.0240     -28.6%      -3.6%
Omnibus       0.2709         --         --    0.2400         --         --
```

The jackknife yields results, however, that are markedly different from the other methods of variance estimation in this example. Specifically, $p$ values founding using the jackknife are never less than 0.25, whereas several $p$ values are less than 0.05 when other methods are used. This discrepancy is explained by the fact that constraints used for matching did not have a feasible solution for several of the jackknife replication groups in this example. As a consequence, variance is artificially inflated. The **microsynth** package prints output during the running of the algorithm that indicates this problem:

```
First model was infeasible for jackknife group 4.
First model was infeasible for jackknife group 6.
```

```
...
First model was infeasible for jackknife group 37.
First model was infeasible for jackknife group 38.
```

The time series plots (see Figure 3 for plots of four outcomes) indicate that treatment and synthetic control do not match across all time points for all outcomes, as expected. Further, the plots confirm the appearance of a treatment effect for some of the outcomes.

```
R> plot_microsynth(sea3)
```

### 4.4. Example 4: One treated case

Although designed for granular data, **microsynth** is applicable with macro-level data, wherein one may have only a single treated case. To illustrate this functionality, we reduce `seattledmi` to a truncated dataset that has a single treated case and 100 non-treated cases as follows:

```
R> ids.t <- names(table(seattledmi$ID[seattledmi$Intervention == 1]))
R> ids.c <- names(table(seattledmi$ID[seattledmi$Intervention == 0]))
R> ids.synth <- c(base::sample(ids.t, 1), base::sample(ids.c, 100))
R> seattledmi.one <- seattledmi[is.element(seattledmi$ID,
+    as.numeric(ids.synth)), ]
```

Given the small number of treated and untreated cases in this example, it is unlikely that it will be possible to exactly match treatment and synthetic control. Therefore, we set `check.feas = TRUE` and `use.backup = TRUE`. Further, the jackknife should not be run with small numbers of cases, so we set `jack = FALSE`. Additionally, we only consider a single outcome (`"any_crime"`).

```
R> sea4 <- microsynth(seattledmi.one, idvar = "ID", timevar = "time",
+    intvar = "Intervention", start.pre = 1, end.pre = 12, end.post = 16,
+    match.out = "any_crime", match.covar = cov.var, result.var = "any_crime",
+    test = "lower", perm = 250, jack = FALSE,
+    check.feas = TRUE, use.backup = TRUE, n.cores = 1)
```

When there is a single treated case, the number of possible placebo groups for permutation equals the number of untreated cases. Therefore, **microsynth** will reset the number of permutation groups to equal 100 in this example, as indicated by the following printed output:

```
Resetting perm = 100
```

In this example, neither the first nor second models had feasible solutions. Therefore, the third model (wherein only the intercept is matched exactly and all other constraints are matched as closely as possible thereafter) is used, as shown by the following:

```
Checking feasibility of first model...
First model is infeasible.
Checking feasibility of second model...
Second model is infeasible.
Will use third model.
```

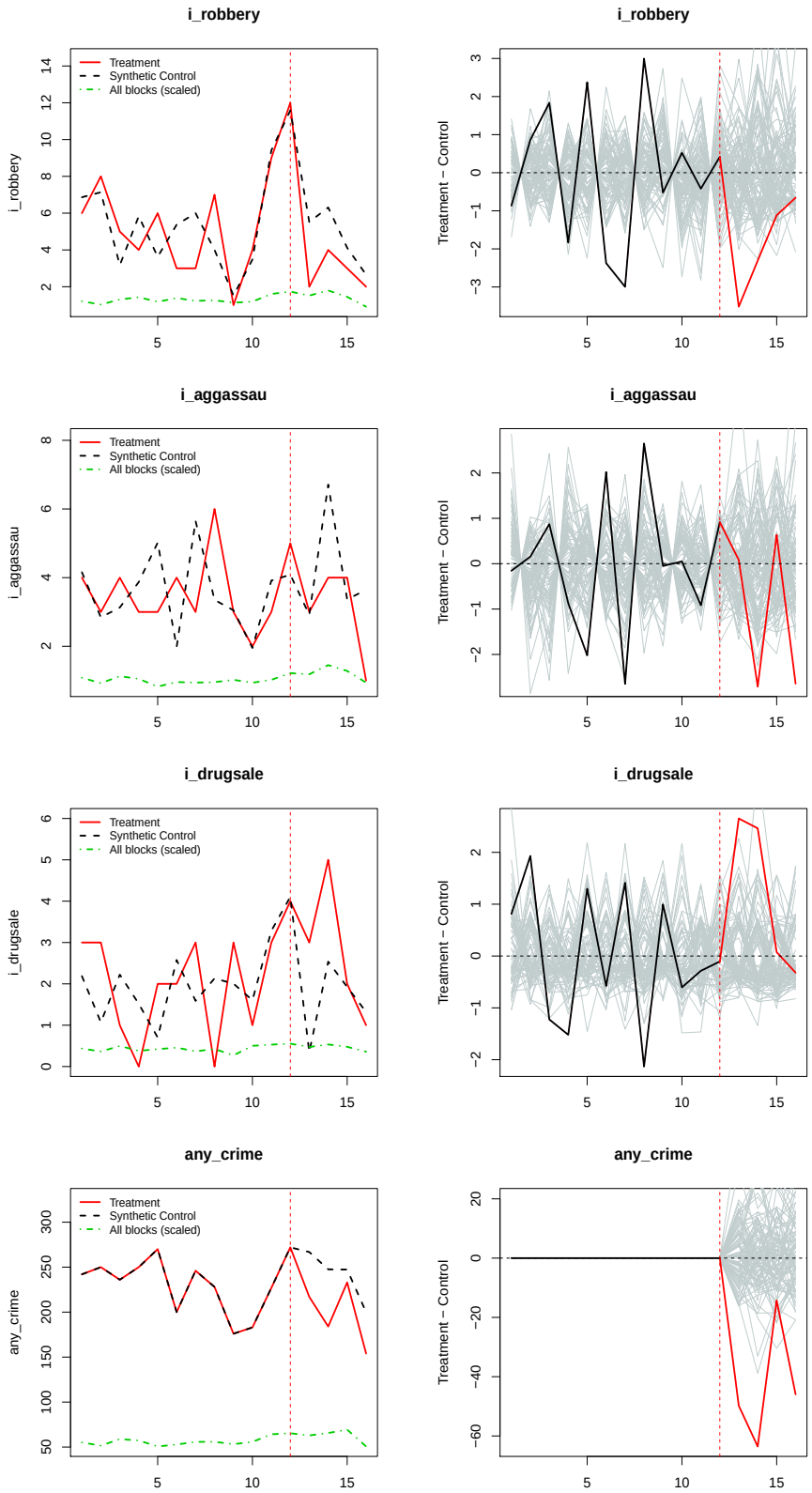


Figure 3: Time series plots for Example 3. Plots on the right give the difference between treatment and synthetic control with permutation groups plotted in gray. The $x$-axis indicates the time period (1–16).

The resulting discordance between treatment and synthetic control may be seen in the balance table below; we see that treatment and synthetic control are not matched.

```
R> summary(sea4)

Weight Balance Table:
             Targets Final.Weighted.Control All.scaled
Intercept          1                 1.0000     1.0000
TotalPop          43                45.0872    63.9208
...              ...                    ...        ...
RENTER_HOU        21                14.0944    16.1089
VACANT_HOU         0                 0.6826     2.2079
any_crime.12       3                 2.3876     1.4554
any_crime.11       1                 0.9742     1.2574
...              ...                    ...        ...
any_crime.2        1                 1.3649     0.9604
any_crime.1        1                 0.4959     1.0297
```

In line with the work of Abadie and Gardeazabal (2003) and Abadie *et al.* (2010), linearization should not be considered when there is a single treated unit. In addition, confidence intervals could yield misleading results with such data. Therefore, only $p$ values determined using permutation should be used for inference in this example, as seen below.

```
Results:

end.post = 16
           Trt  Con Pct.Chng Perm.pVal
any_crime   2 5.53   -63.8%    0.0800
```

The time series plots in Figure 4 also illustrate that treatment and synthetic control are not matched exactly in the pre-intervention time periods.

```
R> plot_microsynth(sea4)
```

### 4.5. Example 5: Cross-sectional data

Although designed for longitudinal data, **microsynth** can be used to create propensity score-type weights with cross-sectional data. In such settings, the method applied aligns with that Hainmueller (2012), wherein calibration as a means of obtaining exact balance between treatment and control groups is proposed. To illustrate, we first isolate to the cross section of `seattledmi` at time 16:

```
R> seattledmi.cross <- seattledmi[seattledmi$time == 16,
+    colnames(seattledmi) != "time"]
```

Since the data are cross-sectional in this example, we set `timevar = NULL`. Likewise, we may only match on time-invariant covariates, so we use `match.out = FALSE`. (We cannot match

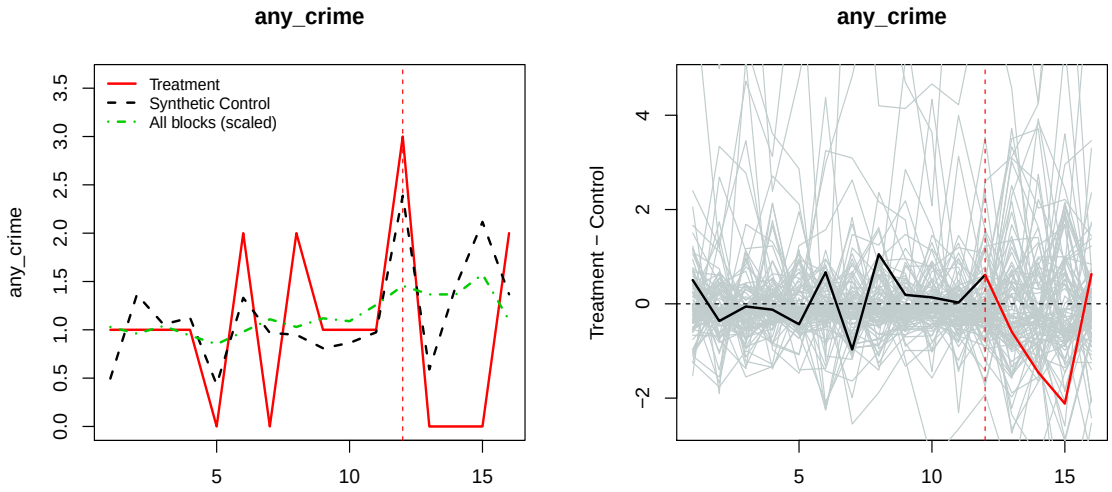


Figure 4: Time series plots for Example 4. Plots on the right give the difference between treatment and synthetic control with permutation groups plotted in gray. The $x$-axis indicates the time period (1–16).

on outcome variables since there are no pre-intervention values of the outcomes in the cross-sectional setting.) However, we set `result.var = match.out` to tabulate results across the desired outcomes. All three methods of variance estimation may be used here.

```
R> sea5 <- microsynth(seattledmi.cross, idvar = "ID",
+    intvar = "Intervention", timevar = NULL, match.out = FALSE,
+    match.covar = cov.var, result.var = match.out, test = "lower",
+    perm = 250, jack = TRUE, cal.epsilon = 1e-06, n.cores = 1)
```

The summary balance table shows that treatment and synthetic control are exactly matched all constraints, which include the intercept and time invariant covariates (with no outcomes).

```
R> summary(sea5)

Weight Balance Table:
           Targets Weighted.Control All.scaled
Intercept       39               39   39.00000
TotalPop      2994             2994 2384.74767
BLACK          173              173  190.52240
HISPANIC       149              149  159.26820
Males_1521      49               49   97.37461
HOUSEHOLDS    1968             1968 1113.55881
FAMILYHOUS     519              519  475.18762
FEMALE_HOU     101              101   81.15495
RENTER_HOU    1868             1868  581.93404
VACANT_HOU     160              160   98.42222
```

Results are shown in the summary output below. In this case, all methods of variance estimation indicate that the intervention was associated with a marked increase in crime levels

(even after controlling for covariates). However, this illustrates the issues inherent with causal inference using cross sectional observational data. The treatment area observes drastically higher crime levels than other areas of Seattle both before and after the intervention. We see that controlling for covariates does not explain this difference. Therefore, the results below are driven by confounding and not by a causal signal.

```
Results:

end.post = 1
           Trt   Con Pct.Chng Linear.pVal Linear.Lower Linear.Upper
i_felony    11  5.58    97.1%      0.9380        12.9%       243.8%
i_misdemea  15  6.21   141.6%      0.9865        50.8%       287.3%
i_drugs      3  2.01    49.4%      0.7211       -42.0%       284.9%
any_crime  154 82.99    85.6%      0.9912        32.8%       159.2%
Omnibus     --    --       --      0.9983           --           --
           Jack.pVal Jack.Lower Jack.Upper Perm.pVal Perm.Lower Perm.Upper
i_felony      0.9375      12.1%     246.4%    0.9840      10.6%     200.6%
i_misdemea    0.9886      52.0%     284.2%    1.0000      32.6%     243.7%
i_drugs       0.7234     -42.6%     288.8%    0.8360     -44.5%      84.4%
any_crime     0.9926      33.2%     158.5%    1.0000      44.5%     219.5%
Omnibus       0.9957         --         --    1.0000         --         --
```

Time series plots are not generated in this example because the cross-sectional data are used.

Note that entropy balancing as proposed by Hainmueller (2012) involves matching treatment and control groups across higher moments of covariates. However, **microsynth** as written only matches across the first moment of the covariates $R_{\ell j}$. To balance entropy across higher moments of the covariates, variables defined by $R^{\nu}_{\ell j}$ for some values of $\nu$ so that $\nu > 1$ should be appended to `data` and included in the `match.covar` statement. Note also that the entropy balancing method of Hainmueller (2012) uses a log-based distance metric in the calibration of weights – this metric can be employed within **microsynth** by setting `calfun = "raking"`.

### 4.6. Example 6: Separate steps for weights and results

Although `microsynth()` is designed to preform weight calculations and result tabulations simultaneously, the function can be used to perform the requisite computations in single steps. To elaborate, we return the setup of Example 1.

First, in order to use `microsynth()` to only calculate weights and not tabulate results, set `result.var = NULL`. Specifically:

```
R> sea6a <- microsynth(seattledmi, idvar = "ID", timevar = "time",
+    intvar = "Intervention", start.pre = 1, end.pre = 12,
+    match.out = match.out, match.covar = cov.var, result.var = NULL,
+    jack = TRUE, perm = 250, cal.epsilon = 1e-05)
```

Next, in order to tabulate results without recalculating weights, we input the resulting '`microsynth`' object (`sea6a`) into a second application of `microsynth()` that uses a non-null value of `result.var`. Specifically, we set `w = sea6a` (alternatively, we could use `w = sea6a$w`). That is:

```
R> sea6b <- microsynth(seattledmi, idvar = "ID", timevar = "time",
+    intvar = "Intervention", end.pre = 12, end.post = 16,
+    result.var = match.out, w = sea6a, test = "lower")
```

Note that any 'microsynth' object (including one that contains tabulated results) can be input into `microsynth()` via the parameter `w` in order to avoid recalculating weights.

## 5. Discussion

We conclude with some points of discussion. First, we provide guidance for the reader as to which methods of inference may be preferable, and we then compare and contrast **microsynth** with **Synth**.

Recall that **microsynth** will produce inferences found using up to three methods. Ideally, all three methods will yield the same conclusions, but in practice that may not be the case. Ergo, which method should form the basis of conclusions? As observed within Kott (2006), linearization is known to understate variance (meaning $p$ values will be too small and confidence intervals will be too narrow), whereas the jackknife tends to overstate variance (leading to inflated $p$ values and wide confidence intervals) in applications involving analysis of surveys where weights have been calibrated. The issues with the jackknife may be exacerbated in settings appropriate for **microsynth**. For instance, in applications of **microsynth**, the design effect (an indication of the inflation of variance due to weighting) tends to be higher for jackknife replication weights than for the main weights. Likewise, in circumstances where the primary model for weighting is infeasible and backup models are used, as in Example 2, jackknife replication weights have more pre-intervention error than do the main weights. In short, the jackknife should be used with caution in settings involving a low number of treated cases and/or an infeasible primary model.

Evidence suggests that permutation methods perform well for assessing the statistical significance of treatment effects. However, the approximations that yield the confidence interval for the permutation methods are more tenuous than those that yield confidence intervals based on standard errors (i.e., linearization and jackknife). Furthermore, we have observed that interval estimation is difficult with count outcomes that have a high prevalence of zeros across cases regardless of method.

Recall also that **microsynth** is an extension of the R package **Synth**. However, **Synth** requires data with a single treated unit, whereas **microsynth** relaxes that restriction. One should not apply **Synth** to data with more than one treated unit while aggregating the data in the treatment region and leaving untreated areas disaggregated (in this case, **Synth** will create weights that sum to unity whereas the weights should sum to $J - J_0$ as in (1)). Furthermore, **microsynth** uses different algorithms for weighting, is designed to handle multiple outcomes simultaneously, and provides a wider array of procedures for producing statistical inferences. Although **microsynth** can be applied when there is exactly one treated case, the procedure used in such settings differs from that of **Synth**. For example, **Synth** employs a nested optimization model, wherein iteration is used to determine an optimal manner of emphasizing certain (linear combinations of) constraints while minimizing the discrepancy between treatment and synthetic control over all constraints. This functionality is not employed in **microsynth** (and is not relevant if exact matches between treatment and synthetic control exist), which helps improve computational efficiency.

Furthermore, there are marked differences among inputs and outputs between the **Synth** and **microsynth** packages: The two packages require similarly formatted data as input (a data frame in tall format), although **Synth** uses a separate function, `dataprep`, for data processing, which is not required by **microsynth**. Furthermore, `microsynth()` includes parameters to allow for matching across multiple outcome variables, the possibility than an exact match may or may not exist, aggregation of pre-intervention time periods, jackknife weights, permutation weights, omnibus statistics, calibration weighting, etc. However, **Synth** also includes inputs to help determine how much emphasis to give certain constraints over others in weighting, which is not a facet of **microsynth**. The **Synth** package uses a separate function, `synth.tab()`, to generate weighting summaries, including a balance assessment for predictors. Analogous summaries are produced within `microsynth()`. Execution of placebo/permutation methods within **Synth** requires that iteratively re-assign treatment status, whereas **microsynth** can automatically create permutation weights within `microsynth()`, while the user need only specify the number of permutation groups to create. Furthermore, post-intervention inferences using **Synth** are provided only through plotting with no automated manner of performing permutation/placebo tests. In addition to having plotting capabilities, **microsynth** also tabulates post-intervention inferences using up to three separate methods for calculating $p$ values and confidence intervals of treatment effects.

Lastly, recall that **microsynth** does not allow for missing values in the dataset on which the methods are to be applied. We provide some brief guidance regarding manners in which missing data can be addressed. First, complete case analysis presents a basic option for handling data with missing values (Little and Rubin 2019), although imputation of missing covariate values is recommended in some cases (D'Agostino, Lang, Walkup, Morgan, and Karter 2001; Stuart and Rubin 2008). Furthermore, Amjad, Shah, and Shen (2018) introduces a framework for addressing missing data in synthetic control procedures. Note that several researchers (e.g., What Works Clearinghouse 2015) advise against the use of quasi-experimental methods in any data containing missing values. Regardless, we urge the user to keep in mind that the aforementioned methods for handling missing data make missing at random assumptions (Little and Rubin 2019).

## Acknowledgments

The authors would like to thank RAND's Center for Causal Inference for generously providing funds to support the development of **microsynth**.

## References

Abadie A, Diamond A, Hainmueller J (2010). "Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program." *Journal of the American Statistical Association*, **105**(490), 493–505. doi:10.1198/jasa.2009.ap08746.

Abadie A, Diamond A, Hainmueller J (2011). "**Synth**: An R Package for Synthetic Control Methods in Comparative Case Studies." *Journal of Statistical Software*, **42**(13), 1–17. doi:10.18637/jss.v042.i13.

Abadie A, Diamond A, Hainmueller J (2015). "Comparative Politics and the Synthetic Control Method." *American Journal of Political Science*, **59**(2), 495–510. doi:10.1111/ajps.12116.

Abadie A, Gardeazabal J (2003). "The Economic Costs of Conflict: A Case Study of the Basque Country." *American Economic Review*, **93**(1), 113–132. doi:10.1257/000282803321455188.

Amjad M, Shah D, Shen D (2018). "Robust Synthetic Control." *The Journal of Machine Learning Research*, **19**(1), 802–852. doi:10.1145/3376930.3376966.

Binder DA (1983). "On the Variances of Asymptotically Normal Estimators from Complex Surveys." *International Statistical Review*, **51**(3), 279–292. doi:10.2307/1402588.

D'Agostino R, Lang W, Walkup M, Morgan T, Karter A (2001). "Examining the Impact of Missing Data on Propensity Score Estimation in Determining the Effectiveness of Self-Monitoring of Blood Glucose (SMBG)." *Health Services and Outcomes Research Methodology*, **2**(3–4), 291–315. doi:10.1023/a:1020375413191.

Dahl DB, Scott D, Roosen C, Magnusson A (2019). ***xtable**: Export Tables to LaTeX or HTML*. R package version 1.8-4, URL https://CRAN.R-project.org/package=xtable.

Deville JC, Särndal CE (1992). "Calibration Estimators in Survey Sampling." *Journal of the American Statistical Association*, **87**(418), 376–382. doi:10.1080/01621459.1992.10475217.

Hainmueller J (2012). "Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies." *Political Analysis*, **20**(1), 25–46. doi:10.1093/pan/mpr025.

Imbens GW (2004). "Nonparametric Estimation of Average Treatment Effects under Exogeneity: A Review." *Review of Economics and Statistics*, **86**(1), 4–29. doi:10.1162/003465304323023651.

Kish L (1965). *Survey Sampling*. 1st edition. John Wiley & Sons.

Kott PS (2006). "Using Calibration Weighting to Adjust for Nonresponse and Coverage Errors." *Survey Methodology*, **32**(2), 133. doi:10.1093/biomet/asn022.

Little RJA, Rubin DB (2019). *Statistical Analysis with Missing Data*. Third edition. John Wiley & Sons. doi:10.1002/9781119482260.

Lumley T (2011). *Complex Surveys: A Guide to Analysis Using R*, volume 565. John Wiley & Sons.

Ormerod JT, Wand MP (2020). ***LowRankQP**: Low Rank Quadratic Programming*. R package version 1.0.4, URL http://CRAN.R-project.org/package=LowRankQP.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robbins MW, Ghosh-Dastidar B, Ramchand R (2020). "Blending Probability and Nonprobability Samples with Applications to a Survey of Military Caregivers." *Journal of Survey Statistics and Methodology*. doi:10.1093/jssam/smaa037. Smaa037.

Robbins MW, Saunders J, Kilmer B (2017). "A Framework for Synthetic Control Methods with High-Dimensional, Micro-Level Data: Evaluating a Neighborhood-Specific Crime Intervention." *Journal of the American Statistical Association*, **112**(517), 109–126. doi:10.1080/01621459.2016.1213634.

Särndal CE (2007). "The Calibration Approach in Survey Theory and Practice." *Survey Methodology*, **33**(2), 99–119. doi:10.1007/978-1-4612-4378-6_1.

Saunders J, Robbins M, Ober AJ (2017). "Moving from Efficacy to Effectiveness." *Criminology & Public Policy*, **16**(3), 787–814. doi:10.1111/1745-9133.12316.

Stuart EA, Rubin DB (2008). "Best Practices in Quasi-Experimental Designs." In *Best Practices in Quantitative Methods*, pp. 155–176. SAGE Publications, London. doi:10.4135/9781412995627.d14.

What Works Clearinghouse (2015). "Designing Quasi-Experiments: Meeting What Works Clearinghouse Standards without Random Assignment." *Technical report*, U.S. Department of Education, Washington, DC. URL https://ies.ed.gov/ncee/wwc/Docs/multimedia/qedwebinar/wwc_webinar_qed_030315.pdf.

## Affiliation:

Michael W. Robbins
Economics, Statistics, and Sociology Department
RAND Corporation
4570 Fifth Avenue #600
Pittsburgh, PA 15136, United States of America
E-mail: mrobbins@rand.org
URL: https://www.rand.org/statistics/bios.html

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
January 2021, Volume 97, Issue 2 *Submitted:* 2018-07-02
doi:10.18637/jss.v097.i02 *Accepted:* 2019-11-10