Package 'microsynth'

May 16, 2018

Title Synthetic Control Methods with Micro- And Meso-Level Data

Version 1.0.9

Description A generalization of the 'Synth' package that is designed for data at a more granular level (e.g., micro-level). Provides functions to construct weights (including propensity score-type weights) and run analyses for synthetic control methods with micro- and meso-level data; see Robbins, Saunders, and Kilmer (2017) <doi:10.1080/01621459.2016.1213634>.

Depends R (>= 3.3.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports boot, survey, MASS, stats, utils, kernlab, LowRankQP, nleqslv, pracma

Suggests knitr, xlsx, rmarkdown

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Michael Robbins [aut, cre], Steven Davenport [aut]

Maintainer Michael Robbins <mrobbins@rand.org>

URL https://github.com/ssdavenport/microsynth

Repository CRAN

Date/Publication 2018-05-16 15:33:56 UTC

R topics documented:

`microsynth` *Synthetic control methods for micro- and meso-level data.*

Description

Implements the synthetic control method for micro-level data as outlined in Robbins, Saunders, and Kilmer (2017). `microsynth` is designed for use in assessment of the effect of an intervention using longitudinal data. However, it may also be used to calculate propensity score-type weights in cross-sectional data. `microsynth` is a generalization of `Synth` (see Abadie and Gardeazabal (2003) and Abadie, Diamond, Hainmueller (2010, 2011, 2014)) that is designed for data at a more granular level (e.g., micro-level). For more details see the help vignette: `vignette('microsynth', package = 'microsynth')`.

`microsynth` develops a synthetic control group by searching for weights that exactly match a treatment group to a synthetic control group across a number of variables while also minimizing the discrepancy between the synthetic control group and the treatment group across a set second set of variables. `microsynth` works in three primary steps: 1) calculation of weights, 2) plotting of treatment vs. synthetic control for pertinent outcomes, and 3) calculation of results.

The time range over which data are observed is segmented into pre- and post-intervention periods. Treatment is matched to synthetic control across the pre-intervention period, and the effect of the intervention is assessed across the post-intervention (or evaluation) period. The input `end.pre` (which gives the last pre-intervention time period) is used to delineate between pre- and post-intervention. Note that if the intervention is not believed to have an instantaneous effect, `end.pre` should indicate the time of the intervention.

Variables are categorized as outcomes (which are time-variant) and covariates (which are time-invariant). Using the respective inputs `match.covar` and `match.out`, the user specifies across which covariates and outcomes (and which pre-intervention time points of the outcomes) treatment is to be exactly matched to synthetic control. The inputs `match.covar.min` and `match.out.min` are similar but instead specify variables across which treatment is to be matched to synthetic control as closely as possible. If there are no variables specified in `match.covar.min` and `match.out.min`, the function `calibrate()` from the `survey` package is used to calculate weights. Otherwise, the function `LowRankQP()` from the package of the same name is used. In the event that the model specified by `match.covar` and `match.out` is not feasible (i.e., weights do not exist that exactly match treatment and synthetic control subject to the given constraints), a less restrictive backup model is used.

`microsynth` has the capability to perform statistical inference using Taylor series linearization, a jackknife and permutation methods. Several sets of weights are calculated. A set of main weights is calculated that is used to determine a point estimate of the intervention effect. The main weights can also be used to perform inferences on the point estimator via Taylor series linearization. If a jackknife is to be used, one set of weights is calculated for each jackknife replication group, and if permutation methods are to be used, one set of weights is calculated for each permutation group. If treatment and synthetic control are not easily matched based upon the model outlined in `match.covar` and `match.out` (i.e., an exact solution is infeasible or nearly infeasible), it is recommended that the jackknife not be used for inference.

The software provides the user the option to create time series plots of outcomes for both the treatment and outcome groups during the pre- and post-intervention time periods and to output overall

findings in an Excel file. For each outcome variable, the results list the estimated treatment effect, as well as confidence intervals of the effect and p-values of a hypothesis test that assesses whether the effect is zero. Such results are produced as needed for each of the three methods of statistical inference noted above. `microsynth` can also apply an omnibus test that examines the presence of a treatment effect jointly across several outcomes.

Usage

```
microsynth(data, idvar, intvar, timevar = NULL, start.pre = NULL,
  end.pre = NULL, end.post = NULL, match.out = TRUE, match.covar = TRUE,
  match.out.min = NULL, match.covar.min = NULL, result.var = TRUE,
  omnibus.var = result.var, plot.var = TRUE, period = 1,
  scale.var = "Intercept", confidence = 0.9, test = "twosided",
  perm = 0, jack = 0, use.survey = TRUE, cut.mse = Inf,
  check.feas = FALSE, use.backup = FALSE, w = NULL, max.mse = 0.01,
  maxit = 250, cal.epsilon = 1e-04, calfun = "linear", bounds = c(0,
  Inf), result.file = NULL, plot.file = NULL, sep = FALSE,
  legend.spot = "bottomleft")
```

Arguments

`data`	A data frame. If longitudinal, the data must be entered in tall format (e.g., at the case/time-level with one row for each time period for each case). Missingness is not allowed. All individuals must have non-NA values of all variables at all time points.
`idvar`	A character string that gives the variable in `data` that identifies multiple records from the same case.
`intvar`	A character string that gives the variable in `data` that corresponds to the intervention variable. The intervention variable indicates which cases (and times) have received the intervention. The variable should be binary, with a 1 indicating treated and 0 indicating untreated. If `end.pre` is specified, a case is considered treated if there is 1 or more non-zero entries in the column indicated by `intvar` for that case (at any time point). If `end.pre` is not specified, an attempt will be made to use `intvar` to determine which time periods will be considered post-intervention (i.e., the times contained in the evaluation period). In this case, the evaluation period is considered to begin at the time of the first non-zero entry in `intvar`).
`timevar`	A character string that gives the variable in `data` that differentiate multiple records of the same case. Can be set to `NULL` only when used with cross-sectional data (i.e., with one observation per entry in `idvar`).
`start.pre`	An integer indicating the time point that corresponds to the beginning of the pre-intervention period used for matching. When `start.pre = NULL` (default), it is reset to the minimum time appearing in the column given by `timevar`. If `match.out` (and `match.out.min`) are given in list format, `start.pre` is ignored except for plotting.
`end.pre`	An integer that gives the final time point of the pre-intervention period. That is, `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. Setting end.pre = NULL will begin the post-intervention period at the time that corresponds to the first non-zero entry in the column indicated by intvar.

end.post — An integer that gives the maximum post-intervention time that is taken into when compiling results. That is, the treatment and synthetic control groups are compared across the outcomes listed in result.var from the first time following the intervention up to end.post. Can be a vector (ordered, increasing) giving multiple values of end.post. In this case, the results will be compiled for each entry in end.post. When end.post = NULL (the default), it is reset to the maximum time that appears in the column given by timevar.

match.out — Either A) logical, B) a vector of variable names that indicates across which time-varying variables treatment is to be exactly matched to synthetic control pre-intervention, or C) a list consisting of variable names and timespans over which variables should be aggregated before matching. Note that outcome variables and time-varying covariates should be included in match.out.

If match.out = TRUE (the default), it is set equal to result.var; if match.out = NULL or match.out = FALSE, no outcome variables are factored into the calculation of weights. If match.out is passed a vector of variable names, then weights are calculated to match treatment and synthetic control for the value of each variable that appears in match.out at each time point from start.pre to end.pre. Otherwise, to allow more flexibility, match.out may also be a list that gives an outcome-based model outlining more specific constraints that are to be exactly satisfied within calibration weighting. In this case, each entry of match.out is a vector of integers, and the names of entries of match.out are the outcome variables to which the vectors correspond. Each element of the vectors gives a number of time points that are to be aggregated for the respective outcome, with the first element indicating time points immediately prior the beginning of the post-intervention period. The sum of the elements in each vector should not exceed the number of pre-intervention time periods in the data.

The following examples show the proper formatting of match.out as a list. Assume that there are two outcomes, Y1 and Y2 (across which treatment is to be matched to synthetic control), and end.pre = 10 (i.e., the post-intervention period begins at time 11). Let match.out = list('Y1' = c(1, 3, 3), 'Y2'= c(2,5,1)). According to this specification, treatment is to be matched to synthetic control across: a) The value of Y1 at time 10; b) the sum of Y1 across times 7, 8 and 9; c) the sum of Y1 across times 4, 5 and 6; e) The sum of Y2 across times time 9 and 10; e) the sum of Y2 across times 4, 5, 6, 7, and 8; f) the value of Y2 at time 3. Likewise, if match.out = list('Y1' = 10, 'Y2'= rep(1,10)), Y1 is matched to synthetic control the entire aggregated pre-intervention time range, and Y2 is matched at each pre-intervention time point individually.

match.covar — Either a logical or a vector of variable names that indicates which time invariant covariates are to be used for weighting. Weights are calculated so that treatment and synthetic control exactly match across these variables. If match.covar = TRUE, it is set equal to a vector of variable names corresponding to the time invariant variables that appear in data. If match.covar = FALSE, it is set to NULL (in

which case no time-invariant variables are used for matching when calculating weights).

`match.out.min` A vector or list of the same format as `match.out` that is used to specify additional time-varying variables to match on, but which need not be matched exactly. Weights are calculated so the distance is minimized between treatment and synthetic control across these variables. If `match.out.min = NULL`, no outcome-based constraints beyond those indicated by `match.out` are imposed (i.e., all outcome variables will be matched on exactly).

`match.covar.min`
A vector of variable names that indicates supplemental time invariant variables that are to be used for weighting, for which exact matches are not required. Weights are calculated so the distance is minimized between treatment and synthetic control across these variables.

`result.var` A vector of variable names giving the outcome variables for which results will be reported. Time-varying covariates should be excluded from `result.var`. If `result.var = TRUE` (the default), `result.var` is set as being equal to all time-varying variables that appear in `data`. If `result.var = NULL` or `result.var = FALSE`, results are not tabulated.

`omnibus.var` A vector of variable names that indicates the outcome variables that are to be used within the calculation of the omnibus statistic. Can also be a logical indicator. When `omnibus.var = TRUE`, it is reset as being equal to `result.var`. When `omnibus.var = NULL` or `omnibus = FALSE`, no omnibus statistic is calculated.

`plot.var` A vector of variable names giving the outcome variables that are shown in plots. If `plot.var = NULL` or `plot.var = FALSE`, no plots are generated. If `plot.var = TRUE`, it is reset as equaling all time variant variables in `data`.

`period` An integer that gives the granularity of the data that will be used for plotting and compiling results; if `match.out` and `match.out.min` are provided a vector of variable names, it will also affect the calculation of weights used for matching. In this case, matching of treatment and synthetic control is performed at a temporal granularity defined by `period`. For instance, if monthly data are provided and `period = 3`, data are aggregated to quarters for plots and results (and weighting unless otherwise specified). If `match.out` and `match.out.min` are provided a list, `period` only affects plots and how results are displayed.

`scale.var` A variable name. When comparing the treatment group to all cases, the latter is scaled to the size of the former with respect to the variable indicated by `scale.var`. Defaults to the number of units receiving treatment (i.e., the intercept).

`confidence` The level of confidence for confidence intervals.

`test` The type of hypothesis test (one-sided lower, one-sided upper, or two-sided) that is used when calculating p-values. Entries of `'lower'`, `'upper'`, and `'twosided'` are recognized.

`perm` An integer giving the number of permutation groups that are used. If `perm = 0`, no permutation groups are generated, permutation weights are not calculated, and permutations do not factor into the reported results. `perm` is set to the number of possible permutation groups if the former exceeds the latter.

jack — An integer giving the number of replication groups that are used for the jackknife. jack can also be a logical indicator. If jack = 0 or jack = FALSE, no jackknife replication groups are generated, jackknife weights are not calculated, and the jackknife is not considered when reporting results. If jack = TRUE, it is reset to being equal to the minimum between the number of total cases in the treatment group and the total number of cases in the control group. jack is also reset to that minimum if it, as entered, exceeds that minimum.

use.survey — If use.survey = TRUE, Taylor series linearization is applied to the estimated treatment effect within each permutation group. Setting use.survey = TRUE makes for better inference but increases computation time substantially. Confidence intervals for permutation groups are calculated only when use.survey = TRUE.

cut.mse — The maximum error (given as mean-squared error) permissible for permutation groups. Permutation groups with a larger than permissible error are dropped when calculating results. The mean-squared error is only calculated over constraints that are to be exactly satisfied.

check.feas — A logical indicator of whether or not the feasibility of the model specified by match.out is evaluated prior to calculation of weights. If check.feas = TRUE, feasibility is assessed. If match.out is found to not specify a feasible model, a less restrictive feasible backup model will be applied to calculate the main weights and for jackknife and permutation methods.

use.backup — A logical variable that, when true, indicates whether a backup model should be used whenever the model specified by match.out yields unsatisfactory weights. Weights are deemed to be unsatisfactory if they do not sufficiently satisfy the constraints imposed by match.out and match.covar. Different backup models may be used for each of the main, jackknife or permutation weights as needed.

w — A list of the form as returned by a prior application of microsynth. If w = NULL, weights are calculated from scratch. Entering a non-NULL value affords the user the ability to use previously calculated weights.

max.mse — The maximum error (given as mean-squared error) permissible for constraints that are to be exactly satisfied. If max.mse is not satisfied by these constraints, and either check.feas = TRUE or use.backup = TRUE, then back-up models are used.

maxit — The maximum number of iterations used within the calibration routine (calibrate() from the survey package) for calculating weights.

cal.epsilon — The tolerance used within the calibration routine (calibrate() from the survey package) for calculating weights.

calfun — The calibration function used within the calibration routine (calibrate() from the survey package) for calculating weights.

bounds — Bounds for calibration weighting (fed into the calibrate() from the survey package).

result.file — A character string giving the name of a file that will be created in the home directory containing results. If result.file = NULL (the default), no file is created. If end.post has length 1, a .csv file is created. If end.post has length greater than one, a formatted .xlsx file is created with one tab for each element of end.post. If result.file has a .xlsx (or .xls) extension (e.g., the last

five characters of result.file are '.xlsx'), an .xlsx file is created regardless of the length of end.post.

plot.file — A character string giving the name of file that will be created in the home directory containing plots (if plot.var is non-NULL). The name should have a .pdf extension.

sep — If sep = TRUE, separate plots will be generated for each outcome. Applicable only if plotting variables are specified (plot.var is non-NULL) and plots are saved to file (plot.file is non-NULL). To change display of plots produced as output, use par.

legend.spot — The location of the legend in the plots.

Details

microsynth requires specification of the following inputs: data, idvar, intvar. data is a longitudinal data frame; idvar and intvar are character strings that specific pertinent columns of data. In longitudinal data, timevar should be specified. Furthermore, specification of match.out and match.covar is recommended.

microsynth can also be used to calculate propensity score-type weights in cross sectional data (in which case timevar does not need to be specified) as proposed by Hainmueller (2012).

microsynth calculates weights using survey::calibrate() from the survey package in circumstances where a feasible solution exists for all constraints, whereas LowRankQP::LowRankQP() is used to assess feasibility and to calculate weights in the event that a feasible solution to all constraints does not exist. The LowRankQP routine is memory-intensive and can run quite slowly in data that have a large number of cases. To prevent LowRankQP from being used, set match.out.min = NULL, match.covar.min= NULL, check.feas = FALSE, and use.backup = FALSE.

Value

microsynth returns a list with up to four elements: a) w, b) Results, c) svyglm.stats, and c) Plot.Stats. The fourth element is returned only if plot.var is not NULL or FALSE.

w is a list with five elements: a) Weights, b) Intervention, c) MSE, d) Model, and e) Summary. Assume there are C total sets of weights calculated, where C = 1 + jack + perm, and there are N total cases across the treatment and control groups. w$Weights is an N x C matrix, where each column provides a set of weights. w$Intervention is an N x C matrix made of logical indicators that indicate whether or not the case in the respective row is considered treated (at any point in time) for the respective column. Entries of NA are to be dropped for the respective jackknife replication group (NAs only appear in jackknife weights). w$MSE is a 6 x C matrix that give the MSEs for each set of weights. MSEs are listed for the primary and secondary constraints for the first, second, and third models. w$Model is a length-C vector that indicates whether backup models were used in the calculation of each set of weights. w$Summary is a three-column matrix that (for treatment, synthetic control, and the full dataset), shows aggregate values of the variables across which treatment and synthetic control are matched. The summary, which is tabulated only for the primary weights, is also printed by microsynth while weights are being calculated.

Further, Results is a list where each element gives the final results for each value of end.post. Each element of Results is itself a matrix with each row corresponding to an outcome variable (and a row for the omnibus test, if used) and each column denotes estimates of the intervention

effects and p-values, upper, and lower bounds of confidence intervals as found using Taylor series linearization (Linear), jackknife (jack), and permutation (perm) methods where needed.

In addition, `svyglm.stats` is a list where each element is a matrix that includes the output from the regression models run using the `svyglm()` function to estimate the treatment effect. The list has one element for each value of `end.post`, and the matrices each have one row per variable in `result.var`.

Lastly, `Plot.Stats` contains the data that are displayed in the plots. `Plot.Stats` is a list with four elements (Treatment, Control, All, Difference). The first three elements are matrices with one row per outcome variable and one column per time point. The last element (which gives the treatment minus control values) is an array that contains data for each permutation group in addition to the true treatment area. Specifically, `Plot.Stats$Difference[,,1]` contains the time series of treatment minus control for the true intervention group; `Plot.Stats$Difference[,,i+1]` contains the time series of treatment minus control for the i^th permutation group.

A summary of weighted matching variables and of results can be viewed using `summary`

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.

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.

Abadie A, Diamond A, Hainmueller J (2015). "Comparative politics and the synthetic control method. *American Journal of Political Science*, 59(2), 495-510.

Abadie A, Gardeazabal J (2003). "The economic costs of conflict: A case study of the Basque Country." *American Economic Review*, pp. 113-132.

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

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.

Examples

```
# Use seattledmi, block-level panel data, to evaluate a crime intervention.
# Declare time-variant (outcome) and time-invariant variables for matching
cov.var <- c('TotalPop', 'BLACK', 'HISPANIC', 'Males_1521',
       'HOUSEHOLDS', 'FAMILYHOUS', 'FEMALE_HOU', 'RENTER_HOU', 'VACANT_HOU')


match.out <- c('i_felony', 'i_misdemea', 'i_drugs', 'any_crime')

set.seed(99199) # for reproducibility

# Perform matching and estimation, without permutations or jackknife
# runtime: < 1 min
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,
       omnibus.var=match.out, plot.var=match.out, test='lower')

# View results
summary(sea1)

# Repeat matching and estimation, with permutations and jackknife
# Set permutations and jack-knife to very few groups (2) for
# quick demonstration only.
# runtime: ~30 min
sea2 <- microsynth(seattledmi, idvar='ID', timevar='time',
        intvar='Intervention', start.pre=1, end.pre=12, end.post=c(14, 16),
        match.out=match.out, match.covar=cov.var, result.var=match.out,
        omnibus.var=match.out, plot.var=match.out, test='lower', perm=250,
        jack=TRUE, plot.file=NULL, sep = TRUE,
        result.file=file.path(tempdir(), 'ExResults2.xlsx'))

# View results
summary(sea2)

# Specify additional outcome variables for matching, which makes
# matching harder.
match.out <- c('i_robbery','i_aggassau','i_burglary','i_larceny',
       'i_felony','i_misdemea','i_drugsale','i_drugposs','any_crime')

# Perform matching, setting check.feas = T and use.backup = T
# to ensure model feasibility
# runtime: ~40 minutes
sea3 <- microsynth(seattledmi, idvar='ID', timevar='time',
        intvar='Intervention', match.out=match.out, match.covar=cov.var,
        start.pre=1, end.pre=12, end.post=16,
        result.var=match.out, plot.var=match.out, perm=250, jack=0,
        test='lower', check.feas=TRUE, use.backup = TRUE,
        plot.file=NULL, result.file=file.path(tempdir(), 'ExResults3.xlsx'))

# Aggregate outcome variables before matching, to boost model feasibility
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))

# After aggregation, use.backup and cheack.feas no longer needed
# runtime: ~40 minutes
sea4 <- microsynth(seattledmi, idvar='ID', timevar='time',
         intvar='Intervention', match.out=match.out, match.covar=cov.var,
         start.pre=1, end.pre=12, end.post=16,
         result.var=names(match.out), omnibus.var=names(match.out),
         plot.var=names(match.out), perm=250, jack = TRUE, test='lower',
         plot.file='ExPlots4.pdf', result.file=file.path(tempdir(), 'ExResults4.xlsx'))
```

```
# Generate weights only (for four variables)
match.out <- c('i_felony', 'i_misdemea', 'i_drugs', 'any_crime')


# runtime: ~ 20 minutes
sea5 <- microsynth(seattledmi,  idvar='ID', timevar='time',
         intvar='Intervention', match.out=match.out, match.covar=cov.var,
         start.pre=1, end.pre=12, end.post=16,
         result.var=FALSE, plot.var=FALSE, perm=250, jack=TRUE)


# View weights
summary(sea5)

# Generate plots only using previous weights
# runtime: ~ 1 min

# Generate results only
# runtime: < 5 minutes
sea7 <- microsynth(seattledmi, idvar='ID', timevar='time',
          intvar='Intervention',
          start.pre=1, end.pre=12, end.post=c(14, 16),
          result.var=match.out, plot.var=FALSE, test='lower',
          w=sea5$w, result.file=file.path(tempdir(), 'ExResults7.xlsx'))

# View results (including previously-found weights)
summary(sea7)


# Apply microsynth in the traditional setting of Synth
# Create macro-level (small n) data, with 1 treatment unit
set.seed(86872)
ids.t <- names(table(seattledmi$ID[seattledmi$Intervention==1]))
ids.c <- names(table(seattledmi$ID[seattledmi$Intervention==0]))
ids.synth <- c(base::sample(ids.t, 1), base::sample(ids.c, 100))
seattledmi.one <- seattledmi[is.element(seattledmi$ID,
           as.numeric(ids.synth)), ]


# Apply microsynth to the new macro-level data
# runtime: < 5 minutes
sea8 <- microsynth(seattledmi.one, idvar='ID', timevar='time',
           intvar='Intervention',
           start.pre=1, end.pre=12, end.post=16,
           match.out=match.out[4],
           match.covar=cov.var, result.var=match.out[4],
           plot.var=match.out[4], test='lower', perm=250, jack=FALSE,
           check.feas=TRUE, use.backup=TRUE)

## Not run:
# Use microsynth to calculate propensity score-type weights
# Prepare cross-sectional data at time of intervention
seattledmi.cross <- seattledmi[seattledmi$time==16, colnames(seattledmi)!="time"]#'
```

```
# Apply microsynth to find propensity score-type weights
# runtime: ~5 minutes
sea9 <- microsynth(seattledmi.cross, idvar='ID', intvar='Intervention',
             match.out=FALSE, match.covar=cov.var, result.var=match.out,
             test='lower', perm=250, jack=TRUE)

# View results
summary(sea9)

## End(Not run)
```

`print.microsynth` *Displaying microsynth Fits and Results*

Description

Print method for class "microsynth".

Usage

```
## S3 method for class 'microsynth'
print(x, ...)
```

Arguments

`x`	A `microsynth` object produced by `microsynth()`
`...`	further arguments passed to or from other methods.

Value

The functions `print.microsynth` and `summary.microsynth` display information about the microsynth fit and estimation results, if available.

The output 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., once for each specified post-intervention evaluation time.

`seattledmi` *Data for a crime intervention in Seattle, Washington*

Description

The dataset contains information used to evaluate a Drug Market Intervention (DMI) occurring in parts of Seattle, Washington in 2013. The data include 2010 block-level Census data and counts of crime reported by the Seattle Police, by crime type. Crime data are available for one year prior to the intervention and two years after. DMIs are an intervention intended to disrupt drug markets by targeting enforcement priorities at specific market participants. The intervention was applied to 39 blocks in Seattle's International District.

Usage

```
seattledmi
```

Format

A data frame with 154,272 rows and 22 columns, consisting of 9,642 unique blocks with 16 (quarterly) observations each. It contains the following variables:

ID unique Census block ID

time time unit (in quarters)

Intervention time-variant binary indicator; all treated units receive 0 pre-intervention and 1 from the start of the intervention onward, while untreated cases receive 0s throughout

i_robbery number of robberies reported in that block-quarter (time-variant)

i_aggassau number of aggravated assaults reported

i_burglary number of burglaries reported

i_larceny number of larcenies reported

i_felony number of felony crimes reported

i_misdemea number of misdemeanor crimes reported

i_drugsale number of drug sales reported

i_drugposs number of drug possession incidents reported

i_drugs number of drug sale or possession incidents reported

any_crime number of all crimes reported

TotalPop number of residents

BLACK number of African American residents

HISPANIC number of Hispanic residents

Males_1521 number of male residents aged 15-21

HOUSEHOLDS number of households

FAMILYHOUS number of family households

FEMALE_HOU number of female-headed households

RENTER_HOU number of households occupied by renters

VACANT_HOU number of vacant housing units

Source

Demographic data obtained from the 2010 Census, and administrative crime data from the Seattle Police Department.

`summary.microsynth` *Summarizing microsynth Fits and Results*

Description

Summary method for class "microsynth".

Usage

```
## S3 method for class 'microsynth'
summary(object, ...)
```

Arguments

`object`	A `microsynth` object produced by `microsynth()`
`...`	further arguments passed to or from other methods.

Value

The functions `print.microsynth` and `summary.microsynth` display information about the microsynth fit and estimation results, if available.

The output 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., once for each specified post-intervention evaluation time.

Index