



*The World's Largest Open Access Agricultural & Applied Economics Digital Library*

**This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.**

**Help ensure our sustainability.**

Give to AgEcon Search

AgEcon Search

<http://ageconsearch.umn.edu>

[aesearch@umn.edu](mailto:aesearch@umn.edu)

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

*No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.*

The Stata Journal (2017)  
17, Number 4, pp. 834–849

# The `synth_runner` package: Utilities to automate synthetic control estimation using `synth`

Sebastian Galiani  
University of Maryland  
College Park, MD  
galiani@econ.umd.edu

Brian Quistorff  
Microsoft AI and Research  
Redmond, WA  
Brian.Quistorff@microsoft.com

**Abstract.** The synthetic control methodology ([Abadie and Gardeazabal, 2003](#), *American Economic Review* 93: 113–132; Abadie, Diamond, and Hainmueller, 2010, *Journal of the American Statistical Association* 105: 493–505) allows for a data-driven approach to small-sample comparative studies. `synth_runner` automates the process of running multiple synthetic control estimations using `synth`. It conducts placebo estimates in space (estimations for the same treatment period but on all the control units). Inference ( $p$ -values) is provided by comparing the estimated main effect with the distribution of placebo effects. It also allows several units to receive treatment, possibly at different time periods. It allows automatic generation of the outcome predictors and diagnostics by splitting the pretreatment into training and validation portions. Additionally, it provides diagnostics to assess fit and generates visualizations of results.

**Keywords:** st0500, `synth_runner`, synthetic control methodology, randomization inference

## 1 Introduction

The synthetic control methodology (SCM) ([Abadie and Gardeazabal 2003](#); Abadie, Diamond, and Hainmueller 2010) is a data-driven approach to small-sample comparative case studies for estimating treatment effects. Similar to a difference-in-differences design, SCM exploits the differences in treated and untreated units across the event of interest. However, in contrast to a difference-in-differences design, SCM does not give all untreated units the same weight in the comparison. Instead, it generates a weighted average of the untreated units that closely matches the treated unit over the pretreatment period. Outcomes for this synthetic control are then projected into the posttreatment period using the weights identified from the pretreatment comparison. This projection is used as the counterfactual for the treated unit. Inference is conducted using placebo tests.

Along with their article, [Abadie, Diamond, and Hainmueller \(2010\)](#) released the `synth` Stata command for single estimations. The `synth_runner` package builds on `synth` to help conduct multiple estimations, inference, and diagnostics as well as to help generate visualizations of results. `synth_runner` is designed to accompany `synth` but not supersede it. For more details about single estimations (variable weights, ob-

ervation weights, covariate balance, and synthetic control outcomes when there are multiple time periods), use `synth` directly.

## 2 SCM

Abadie, Diamond, and Hainmueller (2010) posit the following data-generating process. Let  $D_{jt}$  be an indicator for treatment for unit  $j$  at time  $t$ . Next, let the observed outcome variable  $Y_{jt}$  be the sum of a time-varying treatment effect,  $\alpha_{jt}D_{jt}$ , and the no-treatment counterfactual  $Y_{jt}^N$ , which is specified using a factor model

$$\begin{aligned} Y_{jt} &= \alpha_{jt}D_{jt} + Y_{jt}^N \\ &= \alpha_{jt}D_{jt} + (\delta_t + \theta_t \mathbf{Z}_j + \lambda_t \boldsymbol{\mu}_j + \varepsilon_{jt}) \end{aligned} \quad (1)$$

where  $\delta_t$  is an unknown time factor,  $\mathbf{Z}_j$  is an  $(r \times 1)$  vector of observed covariates unaffected by treatment,  $\theta_t$  is a  $(1 \times r)$  vector of unknown parameters,  $\lambda_t$  is a  $(1 \times F)$  vector of unknown factors,  $\boldsymbol{\mu}_j$  is an  $(F \times 1)$  vector of unknown factor loadings, and the error  $\varepsilon_{jt}$  is independent across units and time with zero mean. Letting the first unit be the treated unit, we estimate the treatment effect by approximating the unknown  $Y_{1t}^N$  with a weighted average of untreated units

$$\hat{\alpha}_{1t} = Y_{1t} - \sum_{j \geq 2} w_j Y_{jt}$$

Equation (1) simplifies to the traditional fixed-effects equation if  $\lambda_t \boldsymbol{\mu}_j = \phi_j$ . The fixed-effects model allows for unobserved heterogeneity that is only time invariant. The factor model employed by SCM generalizes this to allow for the existence of nonparallel trends between treated and untreated units after controlling for observables.

### 2.1 Estimation

To begin with, let there be a single unit that receives treatment. Let  $T_0$  be the number of pretreatment periods of the  $T$  total periods. Index units  $\{1, \dots, J+1\}$  such that the first unit is the treated unit and the others are “donors”. Let  $\mathbf{Y}_j$  be  $(T \times 1)$  the vector of outcomes for unit  $j$  and  $\mathbf{Y}_0$  be the  $(T \times J)$  matrix of outcomes for all donors. Let  $\mathbf{W}$  be a  $(J \times 1)$  observation-weight matrix  $(w_2, w_3, \dots, w_{J+1})'$ , where  $\sum_{j=2}^{J+1} w_j = 1$  and  $w_j \geq 0 \forall j \in \{2, \dots, J+1\}$ . A weighted average of donors over the outcome is constructed as  $\mathbf{Y}_0 \mathbf{W}$ . Partition the outcome into pretreatment and posttreatment vectors  $\mathbf{Y}_j = (\overleftarrow{\mathbf{Y}}_j, \overrightarrow{\mathbf{Y}}_j)$ . Let  $\mathbf{X}$  represent a set of  $k$  pretreatment characteristics (“predictors”). This includes  $\mathbf{Z}$  (the observed covariates above) and  $M$  linear combinations of  $\overleftarrow{\mathbf{Y}}$  so that  $k = r + M$ . Analogously, let  $\mathbf{X}_0$  be the  $(k \times J)$  matrix of donor predictors. Let  $\mathbf{V}$  be a  $(k \times k)$  variable-weight matrix indicating the relative significance of the predictor variables.

Given  $\mathbf{Y}$  and  $\mathbf{X}$ , estimation of SCM consists of finding the optimal weighting matrices  $\mathbf{W}$  and  $\mathbf{V}$ . For a given  $\mathbf{V}$ ,  $\mathbf{W}$  is picked to minimize the root mean squared

prediction error (RMSPE) of the predictor variables,  $\|\mathbf{X}_1 - \mathbf{X}_0 \mathbf{W}\|_{\mathbf{V}}$ . In this way, the treated unit and its synthetic control look similar along dimensions that matter for predicting pretreatment outcomes. The inferential procedure is valid for any  $\mathbf{V}$ , but [Abadie, Diamond, and Hainmueller \(2010\)](#) suggest that  $\mathbf{V}$  be picked to minimize the prediction error of the pretreatment outcome between the treated unit and the synthetic control. Define distance measures  $\|\mathbf{A}\|_{\mathbf{B}} = \sqrt{\mathbf{A}' \mathbf{B} \mathbf{A}}$  and  $\|\mathbf{A}\| = \sqrt{\mathbf{A}' \text{cols}(\mathbf{A})^{-1} \mathbf{A}}$ .  $\|\bar{\mathbf{Y}}_1 - \bar{\mathbf{Y}}_0 \mathbf{W}\|$  is then the pretreatment RMSPE with a given weighted average of the control units. Define this pretreatment RMSPE as  $\bar{s}_1$ , and define the posttreatment RMSPE as  $\vec{s}_1$ .  $\mathbf{V}$  is then picked to minimize  $\bar{s}_1$  (note that  $\mathbf{W}$  is a function of  $\mathbf{V}$ ).

If weights can be found such that the synthetic control matches the treated unit in the pretreatment period

$$\|\bar{\mathbf{Y}}_1 - \bar{\mathbf{Y}}_0 \mathbf{W}\| = 0 = \|\mathbf{Z}_1 - \mathbf{Z}_0 \mathbf{W}\|$$

and  $\sum_{t=1}^{T_0} \lambda'_t \lambda_t$  is nonsingular, then  $\hat{\alpha}_1$  will have a bias that goes to zero as the number of preintervention periods grows large relative to the scale of the  $\varepsilon_{jt}$ .

## 2.2 Inference

After estimating the effect, determine statistical significance by running placebo tests. Estimate the same model on each untreated unit, assuming it was treated at the same time, to get a distribution of “in-place” placebo effects. Disallow the actual treated unit from being considered for the synthetic controls of these other units. If the distribution of placebo effects yields many effects as large as the main estimate, then it is likely that the estimated effect was observed by chance. This nonparametric, exact test has the advantage of not imposing any distribution on the errors.

Suppose that the estimated effect for a particular posttreatment period is  $\hat{\alpha}_{1t}$  and that the distribution of corresponding in-place placebos is  $\hat{\alpha}_{1t}^{PL} = \{\hat{\alpha}_{jt} : j \neq 1\}$ . The two-sided  $p$ -value is then

$$\begin{aligned} p\text{-value} &= \Pr(|\hat{\alpha}_{1t}^{PL}| \geq |\hat{\alpha}_{1t}|) \\ &= \frac{\sum_{j \neq 1} 1(|\hat{\alpha}_{jt}| \geq |\hat{\alpha}_{1t}|)}{J} \end{aligned}$$

and the one-sided  $p$ -values (for positive effects) are

$$p\text{-value} = \Pr(\hat{\alpha}_{1t}^{PL} \geq \hat{\alpha}_{1t})$$

When treatment is randomized, this becomes classical randomization inference.<sup>1</sup> If treatment is not randomly assigned, the  $p$ -value still has the interpretation of being the

1. One may want to include  $\hat{\alpha}_{1t}$  in the comparison distribution as is common in randomization inference. This adds a 1 to the numerator and denominator of the  $p$ -value fraction. [Abadie, Diamond, and Hainmueller \(2015\)](#) and [Cavallo et al. \(2013\)](#), however, do not take this approach. With multiple treatments, there would be several approaches to adding the effects on the treated to the comparison distribution, so they are not dealt with here.

proportion of control units that have an estimated effect at least as large as that of the treated unit. Confidence intervals can be constructed by inverting the  $p$ -values for  $\hat{\alpha}_{1t}$ . However, one should take care with these. As noted by Abadie, Diamond, and Hainmueller (2015), they do not have the standard interpretation when treatment is not considered randomly assigned.

To gauge the joint effect across all posttreatment periods, Abadie, Diamond, and Hainmueller (2010) suggest using posttreatment RMSPE  $\vec{s}_1$ . In this case,  $\vec{s}_1$  would be compared with the corresponding  $\vec{s}_1^{PL}$ .

The placebo effects may be quite large if those units were not matched well in the pretreatment period. This would cause  $p$ -values to be too conservative. To control for this, one may want to adjust  $\hat{\alpha}_{jt}$  and  $\vec{s}_j$  for the quality of the pretreatment matches. Adjustment can be achieved by two mechanisms:

- Restricting the comparison set of control effects to include only those that match well. This is done by setting a multiple  $m$  and removing all placebos  $j$  with  $\overleftarrow{s}_j > m \overleftarrow{s}_1$ .
- Dividing all effects by the corresponding pretreatment match quality  $\overleftarrow{s}$  to get standardized (studentized) measures:  $\hat{\alpha}_{jt}/\overleftarrow{s}_j$  and  $\vec{s}_j/\overleftarrow{s}_j$ .

Inference can then be conducted over four quantities  $(\hat{\alpha}_{jt}, \vec{s}_j, \hat{\alpha}_{jt}/\overleftarrow{s}_j, \vec{s}_j/\overleftarrow{s}_j)$ , and the comparison set can also be limited by the choice of  $m$ .

## 2.3 Multiple events

The extension by Cavallo et al. (2013) allows for more than one unit to experience treatment and at possibly different times. Index treated units  $g \in \{1, \dots, G\}$ . Let  $J$  be those units that never undergo treatment. For a particular treatment  $g$ , one can estimate an effect, say, the first posttreatment period effect  $\hat{\alpha}_g$  (one could use any of the four types discussed above). We omit the  $t$  subscript because treatment dates may differ across events. Over all the treatments, the average effect is  $\bar{\alpha} = G^{-1} \sum_{g=1}^G \hat{\alpha}_g$ .

For each treatment  $g$ , one generates a corresponding set of placebo effects  $\hat{\alpha}_g^{PL}$ , where each untreated unit is thought of as entering treatment at the same time as unit  $g$ . If two treated units have the same treatment period, then their placebo sets will be the same.

Averaging over the treatments to obtain  $\bar{\alpha}$  smooths out noise in the estimate. The same should be done in constructing  $\bar{\alpha}^{PL}$ , the set of placebos with which the average treatment estimate is compared for inference. It should be constructed from all possible averages where a single placebo is taken from each  $\hat{\alpha}_g^{PL}$ . There are  $N_{\overline{PL}} = \prod_{g=1}^G J_g$  such possible averages.<sup>2</sup> Let  $i$  index be a selection where a single placebo effect is chosen from

2. The pool may be restricted by match quality. If  $J_g^m$  is the number of controls that match as well as treated unit  $g$  for the same time period, then  $N_{\overline{PL}}^m = \prod_{g=1}^G J_g^m$ .

each treatment placebo set. Let  $\bar{\alpha}^{PL(i)}$  represent the average of that placebo selection. Inference is now

$$\begin{aligned} p\text{-value} &= \Pr(|\bar{\alpha}^{PL}| \geq |\bar{\alpha}|) \\ &= \frac{\sum_{i=1}^{N_{PL}} 1(|\bar{\alpha}^{PL(i)}| \geq |\bar{\alpha}|)}{N_{PL}} \end{aligned}$$

## 2.4 Diagnostics

Cavallo et al. (2013) perform two basic checks to see whether the synthetic control serves as a valid counterfactual. The first is to check whether a weighted average of donors can approximate the treated unit in the pretreatment. This should be satisfied if the treated unit lies within the convex hull of the control units. One can visually compare the difference in pretreatment outcomes between a unit and its synthetic control. Additionally, one could look at the distribution of pretreatment RMSPEs and see what proportion of control units have values at least as high as that of the treated unit. Cavallo et al. (2013) discard several events from their study because they cannot be matched appropriately.

Secondly, one can exclude some pretreatment outcomes from the list of predictors and see whether the synthetic control matches well the treated unit in these periods.<sup>3</sup> Because this is still pretreatment, the synthetic control should match well. The initial section of the pretreatment period is often designated the “training” period, with the latter part being the “validation” period. Cavallo et al. (2013) set aside the first half of the pretreatment period as the training period.

## 3 The *synth\_runner* package

The *synth\_runner* package contains several tools to help conduct the SCM estimation. It requires the *synth* package, which can be obtained from the Statistical Software Components archive. The main program is *synth\_runner*, which is outlined here. There are also simple graphing utilities (*effect\_graphs*, *pval\_graphs*, and *single\_treatment\_graphs*) that show basic graphs. These are explained in the following code examples and can be modified easily.

---

3. Note also that unless some pretreatment outcome variables are dropped from the set of predictors, all other covariate predictors are rendered redundant. The optimization of  $V$  will put no weight on those additional predictors in terms of predicting pretreatment outcomes.

### 3.1 Syntax

```
synth_runner depvar predictorvars, {trunit(#) trperiod(#) | d(varname)}
  [ trends pre_limit_mult(real) training_propr(real) gen_vars
  noenforce_const_pre_length ci max_lead(#) n_pl_avgs(string)
  pred_prog(string) deterministicoutput parallel pvals1s
  drop_units_prog(string) xperiod_prog(string) mspeperiod_prog(string)
  synthsettings ]
```

Postestimation graphing commands are shown in the examples below. The syntax is similar to the `synth` command. New options include `d()`, `trends`, `pre_limit_mult()`, `training_propr()`, `ci`, `pvals1s`, `max_lead()`, `n_pl_avgs()`, `parallel`, `deterministicoutput`, `pred_prog()`, `drop_units_prog()`, `xperiod_prog()`, and `mspeperiod_prog()`. Options not explicitly matched will be passed to `synth` as *synthsettings*.

Required settings:

- *depvar* specifies the outcome variable.
- *predictorvars* specifies the list of predictor variables. See `help synth` help for more details.

### 3.2 Options

There are two methods for specifying the unit and time period of treatment: either `trunit()` and `trperiod()` or `d()`. Exactly one of these is required.

`trunit(#)` and `trperiod(#)`, used by `synth`, can be used when there is a single unit entering treatment. Because synthetic control methods split time into pretreatment and treated periods, `trperiod()` is the first of the treated periods and, slightly confusingly, also called posttreatment.

`d(varname)` specifies a binary variable, which is 1 for treated units in treated periods and 0 everywhere else. This allows for multiple units to undergo treatment, possibly at different times.

`trends` will force `synth` to match on the trends in the outcome variable. It does this by scaling each unit's outcome variable so that it is 1 in the last pretreatment period.

`pre_limit_mult(real)` will not include placebo effects in the pool for inference if the match quality of that control, namely, the pretreatment RMSPE, is greater than `pre_limit_mult()` times the match quality of the treated unit. *real* must be greater than or equal to 1.

**training\_propr**(*real*) instructs **synth\_runner** to automatically generate the outcome predictors. The default is **training\_propr**(0), which is to not generate any (the user then includes the desired ones in *predictorvars*). If the value is set to a number greater than 0, then that initial proportion of the pretreatment period is used as a training period, with the rest being the validation period. Outcome predictors for every time in the training period will be added to the **synth** commands. Diagnostics of the fit for the validation period will be outputted. If the value is between 0 and 1, there will be at least one training period and at least one validation period. If it is set to 1, then all the pretreatment period outcome variables will be used as predictors. This will make other covariate predictors redundant. *real* must be greater than or equal to 0 and less than or equal to 1.

**gen\_vars** generates variables in the dataset from estimation. This is allowed only if there is a single period in which units enter treatment. These variables are required for the following: **single\_treatment\_graphs** and **effect\_graphs**. If **gen\_vars** is specified, it will generate the following variables:

**lead** contains the respective time period relative to treatment. **lead** = 1 specifies the first period of treatment. This is to match Cavallo et al. (2013) and is effectively the offset from  $T_0$ .

**depvar\_synth** contains the unit's synthetic control outcome for that time period.

**effect** contains the difference between the unit's outcome and its synthetic control for that time period.

**pre\_rmspe** contains the pretreatment match quality in terms of RMSPE. It is constant for a unit.

**post\_rmspe** contains a measure of the posttreatment effect (jointly over all post-treatment time periods) in terms of RMSPE. It is constant for a unit.

**depvar\_scaled** (if the match was done on trends) is the unit's outcome variable normalized so that its last pretreatment period outcome is 1.

**depvar\_scaled\_synth** (if the match was done on trends) is the unit's synthetic control (scaled) outcome variable.

**effect\_scaled** (if the match was done on trends) is the difference between the unit's scaled outcome and its synthetic control's (scaled) outcome for that time period.

**noenforce\_const\_pre\_length** specifies that maximal histories are desired at each estimation stage. When there are multiple periods, estimations at later treatment dates will have more pretreatment history available. By default, these histories are trimmed on the early side so that all estimations have the same amount of history.

**ci** outputs confidence intervals from randomization inference for raw effect estimates. These should be used only if the treatment is randomly assigned (conditional on covariates and interactive fixed effects). If treatment is not randomly assigned, then these confidence intervals do not have the standard interpretation (see above).



`max_lead(int)` will limit the number of posttreatment periods analyzed. The default is the maximum number of leads that is available for all treatment periods.

`n_pl_avgs(string)` controls the number of placebo averages to compute for inference. The possible total grows exponentially with the number of treated events. The default behavior is to cap the number of averages computed at 1,000,000 and, if the total is more than that, to sample (with replacement) the full distribution. The option `n_pl_avgs(all)` can be used to override this behavior and compute all the possible averages. The option `n_pl_avgs(#)` can be used to specify a specific number less than the total number of averages possible.

`pred_prog(string)` allows for time-contingent predictor sets. The user writes a program that takes as input a time period and outputs via `r(predictors)` a `synth`-style predictor string. If one is not using `training_propr()`, then `pred_prog()` could be used to dynamically include outcome predictors. See example 3 for usage details.

`deterministicoutput`, when used with `parallel`, will eliminate displayed output that would vary depending on the machine (for example, timers and number of parallel clusters) so that log files can be easily compared across runs.

`parallel` will enable parallel processing if the `parallel` command is installed and configured. Version 1.18.2 is needed at a minimum.<sup>4</sup>

`pvals1s` outputs one-sided *p*-values in addition to the two-sided *p*-values.

`drop_units_prog(string)` specifies the name of a program that, when passed the unit to be considered treated, will drop other units that should not be considered when forming the synthetic control. This is usually because they are neighboring or interfering units. See example 3 for usage details.

`xperiod_prog(string)` allows for setting `synth`'s `xperiod()` option, which varies with the treatment period. The user-written program is passed the treatment period and should return, via `r(xperiod)`, a *numlist* suitable for `synth`'s `xperiod()` (the period over which generic predictor variables are averaged). See `synth` for more details on the `xperiod()` option. See example 3 for usage details.

`mspeperiod_prog(string)` allows for setting `synth`'s `mspeperiod()` option, which varies with the treatment period. The user-written program is passed the treatment period and should return, via `r(mspeperiod)`, a *numlist* suitable for `synth`'s `mspeperiod()` (the period over which the prediction outcome is evaluated). See `synth` for more details on the `mspeperiod()` option. See example 3 for usage details.

`synthsettings` specifies pass-through options sent to `synth`. See `help synth` for more information. The following are disallowed: `counit()`, `figure`, `resultsperiod()`.

4. At the time of writing, Statistical Software Components does not contain a new enough version. Newer versions are available via the development website <https://github.com/gvegayon/parallel/>.

### 3.3 Stored results

`synth_runner` stores the following in `e()`:

#### Scalars

<code>e(n_pl)</code>	number of placebo averages used for comparison
<code>e(pval_joint_post)</code>	proportion of placebos that have a posttreatment RMSPE at least as large as the average for the treated units
<code>e(pval_joint_post_t)</code>	proportion of placebos that have a ratio of posttreatment RMSPE over pretreatment RMSPE at least as large as the average ratio for the treated units
<code>e(avg_pre_rmspe_p)</code>	proportion of placebos that have a pretreatment RMSPE at least as large as the average of the treated units; the farther this measure is from 0 toward 1, the better the relative fit of the treated units
<code>e(avg_val_rmspe_p)</code>	when one specifies <code>training_propr()</code> , this is the proportion of placebos that have an RMSPE for the validation period at least as large as the average of the treated units; the farther this measure is from 0 toward 1, the better the relative fit of the treated units

#### Matrices

<code>e(treat_control)</code>	average treatment outcome (centered around treatment) and the average of the outcome of those units' synthetic controls for the pretreatment and posttreatment periods
<code>e(b)</code>	a vector with the per-period effects (unit's actual outcome minus the outcome of its synthetic control) for posttreatment periods
<code>e(pvals)</code>	a vector of the proportions of placebo effects that are at least as large as the main effect for each posttreatment period
<code>e(pvals_std)</code>	a vector of the proportions of placebo standardized effects that are at least as large as the main standardized effect for each posttreatment period
<code>e(failed_opt_targets)</code>	errors when constructing the synthetic controls for nontreated units are handled gracefully; if any are detected, they will be listed in this matrix (errors when constructing the synthetic control for treated units will abort the method)

### 3.4 Example usage

The following examples use a dataset from the `synth` package. Ensure that `synth` was installed with ancillary files (for example, `ssc install synth, all`). This panel dataset contains information for 39 U.S. States for the years 1970–2000 (see Abadie, Diamond, and Hainmueller [2010] for details).

```
. sysuse smoking
(Tobacco Sales in 39 US States)

. tsset state year
    panel variable:  state (strongly balanced)
    time variable:  year, 1970 to 2000
                delta:  1 unit
```

### ► Example 1

Reconstruct example 1 from the `synth` help file (note this is not the exact estimation strategy used in [Abadie, Diamond, and Hainmueller \[2010\]](#)):

```
. synth_runner cigsale beer(1984(1)1988) lnincome(1972(1)1988)
>   retprice age15to24 cigsale(1988) cigsale(1980) cigsale(1975),
>   trunit(3) trperiod(1989) gen_vars
Estimating the treatment effects
Estimating the possible placebo effects (one set for each of the 1 treatment
> periods)
|                                     | Total: 38
.....| 11.00s elapsed.

Conducting inference: 5 steps, and 38 placebo averages
Step 1... Finished
Step 2... Finished
Step 3... Finished
Step 4... Finished
Step 5... Finished

Post-treatment results: Effects, p-values, standardized p-values
```

	estimates	pvals	pvals_std
c1	-7.887098	.1315789	0
c2	-9.693599	.1842105	0
c3	-13.8027	.2105263	0
c4	-13.344	.1315789	0
c5	-17.0624	.1052632	0
c6	-20.8943	.0789474	0
c7	-19.8568	.1315789	.0263158
c8	-21.0405	.1578947	0
c9	-21.4914	.1052632	.0263158
c10	-19.1642	.1842105	.0263158
c11	-24.554	.1052632	0
c12	-24.2687	.1052632	.0263158

The program notes progress toward estimating prediction errors and conducting inference. Results for posttreatment periods are shown by default. In this example, they are negative and significant by the standardized effect measure indicating that California's Proposition 99 studied by [Abadie, Diamond, and Hainmueller \(2010\)](#) likely had a negative effect on cigarette sales.

The following are returned by `synth_runner`:

```
. ereturn list
scalars:
      e(n_pl) = 38
      e(n_pl_used) = 38
      e(pval_joint_post) = .131578947368421
      e(pval_joint_post_std) = 0
      e(avg_pre_rmspe_p) = .9210526315789473
macros:
      e(trperiod) : "1989"
      e(trunit) : "3"
      e(treat_type) : "single unit"
      e(depvar) : "cigsale"
      e(cmd) : "synth_runner"
      e(properties) : "b"
matrices:
      e(b) : 1 x 12
      e(pvals_std) : 1 x 12
      e(pvals) : 1 x 12
      e(treat_control) : 31 x 2
. // If truly random, can modify the p-value
. display (e(pval_joint_post_std)*e(n_pl)+1)/(e(n_pl)+1)
.02564103
```

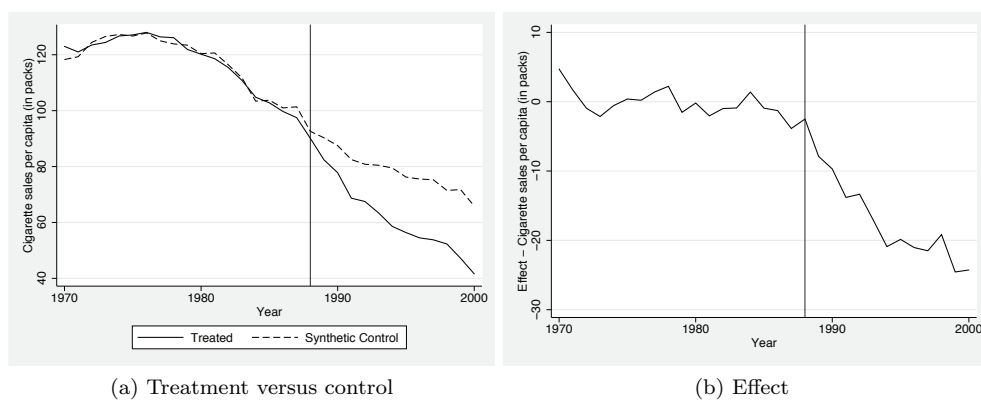
The `e(pval_joint_post)` lists the proportion of effects from control units that have posttreatment RMSPE at least as great as the treated unit. While it is not quite significant at the 10% level, it does not account for pretreatment match quality. The `e(pval_joint_post_std)` value lists the same proportion but scales all values by the relevant pretreatment RMSPE. This measure does show significance. The final measure, `e(avg_pre_rmspe_p)`, is a diagnostic measure noting that the treated unit was matched better than the majority of the control units. If the treatment is considered truly at random, then the true  $p$ -value is a modification that adds 1 to the numerator and denominator (in cases with a single treatment). This is shown for the case of the ratio of posttreatment to pretreatment RMSPE.

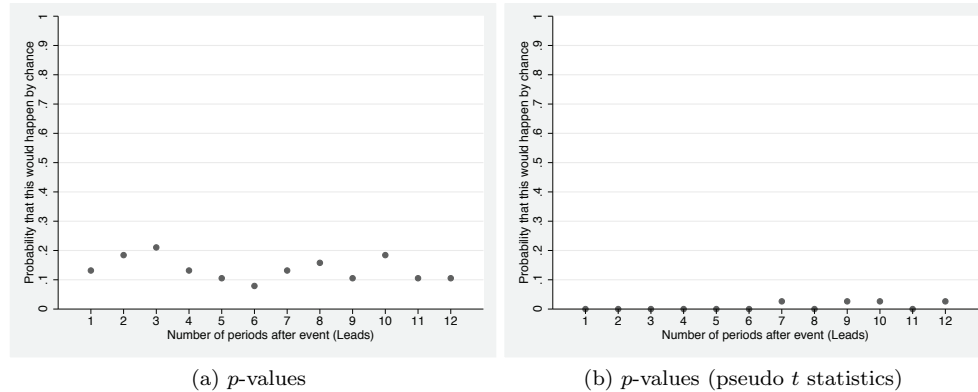
Next, we create common synthetic control graphs. Note that both `effect_graphs` and `single_treatment_graphs` require variables generated from the `gen_vars` option above. The `single_treatment_graphs` command creates the graphs in figure 1 (which are easy to do when there is a single treatment). The first graphs the outcome path of all units, while the second graphs the prediction differences for all units. The `effect_graphs` command creates the graphs in figure 2. One plots the outcome for the unit and its synthetic control, while the other plots the difference between the two (which for posttreatment is the “effect”). The two previous graphing commands allow the option `trlinediff(real)`, which allows the user to offset a vertical treatment from the first treatment period. Likely options include values in the range from (first treatment period–last posttreatment period) to 0, and the default value is  $-1$  (to match [Abadie, Diamond, and Hainmueller \[2010\]](#)). The `pval_graphs` command creates the graphs in figure 3. These plot the  $p$ -values per period for posttreatment periods for both raw and standardized effects.

```

. single_treatment_graphs, trlinediff(-1) effects_ylabels(-30(10)30)
> effects_ymax(35) effects_ymin(-35)
. effect_graphs, trlinediff(-1)
. pval_graphs

```

Figure 1. Graphs from `single_treatment_graphs`Figure 2. Graphs from `effect_graphs`

Figure 3. Graphs from `pval_graphs`

◀

► **Example 2**

In this example, we analyze the same treatment but use some of the more advanced options:

```
. sysuse smoking, clear
(Tobacco Sales in 39 US States)

. tsset state year
(output omitted)

. generate byte D = (state==3 & year>=1989)

. synth_runner cigsale beer(1984(1)1988) lnincome(1972(1)1988)
>   retprice age15to24, trunit(3) trperiod(1989) trends
>   training_propr(`=13/19`) pre_limit_mult(10) gen_vars
(output omitted)

. // Proportion of control units that have a higher RMSPE than the
. // treated unit in the validation period:"
. display round(`e(avg_val_rmspe_p)', 0.001)
.842

. single_treatment_graphs, scaled
. effect_graphs, scaled
. pval_graphs
```

Again, there is a single treatment period, so synthetic control data can be merged into our main dataset. In this setting, we i) specify the treated units or periods with a binary variable; ii) generate the outcome predictors automatically using the initial 13 periods of the pretreatment era (the rest is the “validation” period); iii) match on trends; and iv) limit the control units during inference to those with pretreatment match quality no more than 10 times worse than the match quality of the corresponding treatment units. Now that we have a training or validation period split, there is a new diagnostic. It shows that 84% of the control units have a worse match (a higher RMSPE) during the

validation period. The graphing commands are equivalent. The ones showing the range of effects and raw data are shown in figure 4. One can see that all lines converge on the last pretreatment period because that is the unit by which all are standardized (and all the synthetic controls then match their real units and have zero prediction error).

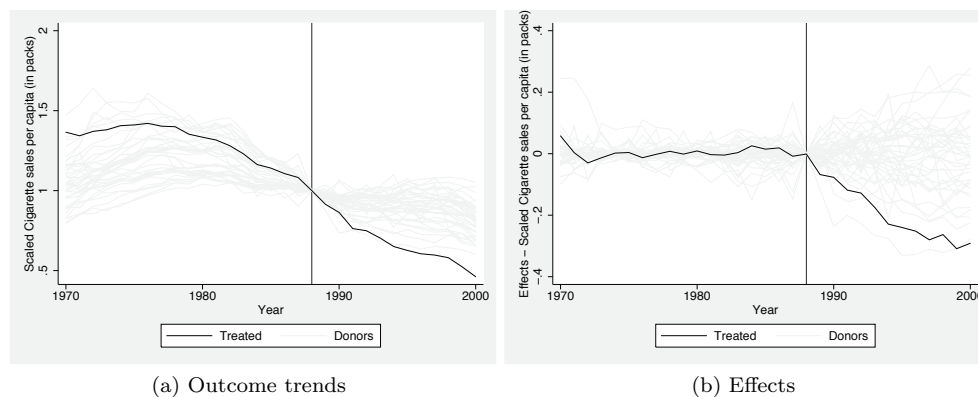


Figure 4. Graphs from `single_treatment_graphs`

◀

### ► Example 3

The final example involves multiple treatments at different time periods and shows usage of user-generated programs to customize the individual `synth` calls based on treatment year and the unit considered as treated. These programs allow, for instance, that the predictors include the last four periods of beer sales and income as predictors and shows how units can be dropped from the comparison set because of concerns about treatment spillovers.

```
. sysuse smoking, clear
(Tobacco Sales in 39 US States)

. tsset state year
(output omitted)

. program my_pred, rclass
1.     args tyear
2.     local beerv "beer(`= `tyear'-4'(1)`=' `tyear'-1`)"
3.     return local predictors "`beerv' lnincome(`= `tyear'-4'(1)`=' `tyear'-1`)"
4. end

. program my_drop_units
1.     args tunit
2.     if `tunit'==39 qui drop if inlist(state,21,38)
3.     if `tunit'==3 qui drop if state==21
4. end
```

```

. program my_xperiod, rclass
1.     args tyear
2.     return local xperiod "`='tyear'-12'(1)`='$tyear'-1'"
3. end

. program my_mspeperiod, rclass
1.     args tyear
2.     return local mspeperiod "`='tyear'-12'(1)`='$tyear'-1'"
3. end

. generate byte D = (state==3 & year>=1989) | (state==7 & year>=1988)
. synth_runner cigsale retprice age15to24, d(D) pred_prog(my_pred) trends
>     training_propr(`=13/18`) drop_units_prog(my_drop_units)
>     xperiod_prog(my_xperiod) mspeperiod_prog(my_mspeperiod)
(output omitted)
. effect_graphs
. pval_graphs

```

We extend example 2 by considering a control state now to be treated (Georgia in addition to California). Note that no treatment actually happened in Georgia in 1987. This is just to illustrate additional usage options. With several treatment periods, we cannot automatically include all the synthetic control outputs back into the main dataset. Some graphs (of `single_treatment_graphs`) can also no longer be made. The `effect_graphs` are shown in figure 5. In addition to showing how predictors and unit dropping can be dynamically generated for the underlying `synth` calls, we also show how this can be done for the `xperiod()` and `mspeperiod()` options to `synth`.

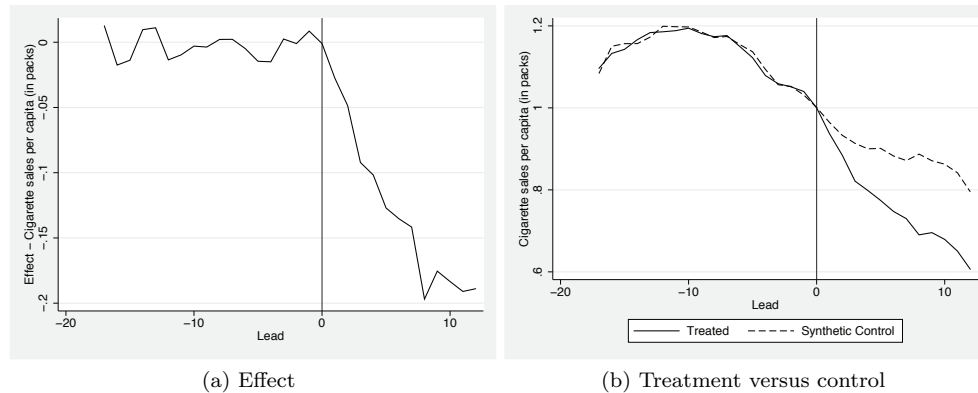


Figure 5. Graphs from `effect_graphs`

◀

## 4 Discussion

The SCM (Abadie and Gardeazabal 2003; Abadie, Diamond, and Hainmueller 2010) allows researchers to quantitatively estimate effects in many small-sample settings in a



manner grounded by theory. In this article, we provided an overview of the theory of SCM and the `synth_runner` package, which builds on the `synth` package of Abadie, Diamond, and Hainmueller (2010). `synth_runner` provides tools to help with the common tasks of fitting a synthetic control model. It automates the process of conducting in-place placebos and calculating inference on the various possible measures. Following Cavallo et al. (2013), it i) extends the initial estimation strategy to allow for multiple units that receive treatment (at potentially different times); ii) allows for matching on trends in the outcome variable rather than on the level; and iii) automates the process of splitting pretreatment periods into “training” and “validation” sections. It also provides graphs of diagnostics, inference, and estimate effects.

## 5 Acknowledgment

We thank the Inter-American Development Bank for financial support.

## 6 References

- Abadie, A., A. Diamond, and J. Hainmueller. 2010. Synthetic control methods for comparative case studies: Estimating the effect of California’s tobacco control program. *Journal of the American Statistical Association* 105: 493–505.
- . 2015. Comparative politics and the synthetic control method. *American Journal of Political Science* 59: 495–510.
- Abadie, A., and J. Gardeazabal. 2003. The economic costs of conflict: A case study of the Basque country. *American Economic Review* 93: 113–132.
- Cavallo, E., S. Galiani, I. Noy, and J. Pantano. 2013. Catastrophic natural disasters and economic growth. *Review of Economics and Statistics* 95: 1549–1561.

### About the authors

Sebastian Galiani is a professor of economics at the University of Maryland, College Park.

Brian Quistorff is a research economist at Microsoft AI and Research, Redmond.