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; Abadie et al., 2010) 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 to the distribution of placebo effects. It allows several units to receive treatment, possibly at different time periods. Automatically generating the outcome predictors and diagnostics by splitting the pre-treatment into training and validation portions is allowed. Additionally, it provides diagnostics to assess fit and generates visualizations of results.

Keywords: st0001, synth_runner, Synthetic Control Methodology, Randomization Inference

1 Introduction[1]

The Synthetic Control Methodology (SCM) (Abadie and Gardeazabal, 2003; Abadie et al., 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 pre-treatment period. Outcomes for this synthetic control are then projected into the post-treatment period using the weights identified from the pre-treatment comparison. This projection is used as the counterfactual for the treated unit. Inference is conducted using placebo tests.

Along with their paper, Abadie et al. (2010) released the `synth` Stata command for single estimations. The `synth_runner` package builds on top of that command to help conduct multiple estimations, inference, diagnostics, and generate visualizations of results. `synth_runner` is designed to accompany `synth` but not supersede it. For more details about single estimations (variable weights, observation weights, covariate balance, and

---

synthetic control outcomes when there are multiple time periods) use `synth` directly.

2 Synthetic Control Methodology

Abadie et al. (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

$$Y_{jt} = \alpha_{jt}D_{jt} + Y_{jt}^N \qquad (1)$$
$$= \alpha_{jt}D_{jt} + (\delta_t + \theta_t \mathbf{Z}_j + \lambda_t \mu_j + \varepsilon_{jt})$$

where δ_t is an unknown time factor, $\mathbf{Z}_j$ is a $(r \times 1)$ vector of observed covariates unaffected by treatment, θ_t is a $(1 \times r)$ vector of unknown parameters, λ_t is a $(1 \times F)$ vector of unknown factors, μ_j is a $(F \times 1)$ vector of unknown factor loadings, and the error ε_{jt} is independent across units and time with zero mean. Letting the first unit be the treated unit, the treatment effect is estimated 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 $\lambda_t \mu_j = \phi_j$. 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 non-parallel 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 pre-treatment periods of the T total periods. Index units $\{1, \ldots, J + 1\}$ 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 $(w_2, w_3, \ldots, w_{J+1})'$ where $\sum_{j=2}^{J+1} w_j = 1$ and $w_j \geq 0 \ \forall j \in \{2, \ldots, J + 1\}$. A weighted average of donors over the outcome is constructed as $\mathbf{Y}_0 \mathbf{W}$. Partition the outcome into pre-treatment and post-treatment vectors $\mathbf{Y}_j = (\overleftarrow{\mathbf{Y}}_j \backslash \overrightarrow{\mathbf{Y}}_j)$. Let $\mathbf{X}$ represent a set of k pre-treatment 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 pre-treatment outcomes. The inferential procedure is valid for any $\mathbf{V}$, but Abadie et al. (2010) suggest that $\mathbf{V}$ be picked to minimize the prediction error of the pre-treatment outcome between the treated unit the synthetic control. Define distance measures $\|\mathbf{A}\|_{\mathbf{B}} = \sqrt{\mathbf{A}'\mathbf{B}\mathbf{A}}$ and $\|\mathbf{A}\| = \sqrt{\mathbf{A}'\mathrm{cols}(\mathbf{A})^{-1}\mathbf{A}}$. $\left\|\overleftarrow{\mathbf{Y}}_1 - \overleftarrow{\mathbf{Y}}_0\mathbf{W}\right\|$ is then the pre-treatment RMSPE with a given weighted average of the control units. Define this as $\overleftarrow{s}_1$ and define $\overrightarrow{s}_1$ as the post-treatment RMSPE. $\mathbf{V}$ is then picked to minimize $\overleftarrow{s}_1$ (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 pre-treatment period:

$$\left\|\overleftarrow{\mathbf{Y}}_1 - \overleftarrow{\mathbf{Y}}_0\mathbf{W}\right\| = 0 = \|\mathbf{Z}_1 - \mathbf{Z}_0\mathbf{W}\| \tag{2}$$

and $\sum_{t=1}^{T_0} \lambda_t'\lambda_t$ is non-singular, then $\hat{\alpha}_1$ will have a bias that goes to zero as the number of pre-intervention periods grows large relative to the scale of the ε_{jt}.

2.2 Inference

After estimating the effect, statistical significance is determined 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 non-parametric, exact test has the advantage of not imposing any distribution on the errors.

Suppose that the estimated effect for a particular post-treatment 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

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

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

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

When treatment is randomized this becomes classical randomization inference[2]. If treatment is not randomly assigned, the p-value still has the interpretation of being the 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 $\hat{\alpha}_{1t}$.

---

[2]One may want to include $\hat{\alpha}_{1t}$ in the comparison distribution as is common in randomization inference. This adds a one to the numerator and denominator of the p-value fraction. Abadie et al. (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.

Care should be taken with these, however. As noted by Abadie et al. (2015), they do not have the standard interpretation when treatment is not considered randomly assigned.

To gauge the joint effect across all post-treatment periods Abadie et al. (2010) suggest using post-treatment RMSPE $\vec{s}_1$. In this case $\vec{s}_1$ would be compared to the corresponding $\vec{s}_1^{PL}$.

The placebo effects may be quite large if those units where not matched well in the pre-treatment 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 pre-treatment matches. Adjustment can be achieved by two mechanisms:

- Restricting the comparison set of control effects to only include 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 pre-treatment 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 $g \in \{1, \ldots, G\}$. Let J be those units that never undergo treatment. For a particular treatment g, one can estimate an effect, say the first post-treatment period effect $\hat{\alpha}_g$ (one could use any of the four types discussed above). We omit the t subscript as 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 the 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 against 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[3]. Let i index a selection where a single placebo effect is chosen from each treatment placebo set. Let $\bar{\alpha}^{PL(i)}$ represents the average of that placebo selection. Inference is now

$$p\text{-value} = \Pr\left(|\bar{\alpha}^{PL}| \geq |\bar{\alpha}|\right)$$
$$= \frac{\sum_{i=1}^{N_{\overline{PL}}} 1(|\bar{\alpha}^{PL(i)}| \geq |\bar{\alpha}|)}{N_{\overline{PL}}}$$

---

[3]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 $N_{\overline{PL}}^m = \prod_{g=1}^{G} J_g^m$.

2.4 Diagnostics

Cavallo et al. (2013) perform two basic checks to see if the synthetic control serves as a valid counterfactual. The first is to check if a weighted average of donors is able to approximate the treated unit in the pre-treatment. This should be satisfied if the treated unit lies within the convex hull of the control units. One can visually compare the difference in pre-treatment outcomes between a unit and its synthetic control. Additionally one could look at the distribution of pre-treatment RMSPE's 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 study as they can not be matched appropriately.

Secondly, one can exclude some pre-treatment outcomes from the list of predictors and see if the synthetic control matches well the treated unit in these periods.[4] As this is still pre-treatment, the synthetic control should match well. The initial section of the pre-treatment period is often designated the "training" period with the later part being the "validation" period. Cavallo et al. (2013) set aside the first half of the pre-treatment period as the training period.

3 The `synth_runner` Package

The `synth_runner` package contains several tools to help conduct SCM estimation. It requires the `synth` package which can be obtained from the SSC archive. The main program is `synth_runner`, which is outlined here. Additionally, there are simple graphing utilities (`effect_graphs`, `pval_graphs`, `single_treatment_graphs`) that show basic graphs. These are explained in the following code examples and can be modified easily.

3.1 Syntax

synth_runner *depvar predictorvars* $\big[$, <u>tru</u>nit($\#$) <u>tr</u>period($\#$) d(*varname*) <u>tren</u>ds
pre_limit_mult(*real*) training_propr(*real*) <u>gen</u>_vars noenforce_const_pre_length
ci max_lead($\#$) n_pl_avgs(*string*) pred_prog(*string*) <u>determi</u>nisticout <u>para</u>llel
pvals1s drop_units_prog(*string*) xperiod_prog(*string*) mspeperiod_prog(*string*) *synth-settings* $\big]$

Post-estimation 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_leads, n_pl_avgs, parallel, deterministicout, pred_prog, drop_units_prog, xperiod_prog, mspeperiod_prog. Options not explicitly matched will be passed to `synth` as synthsettings.

3.2 Settings

Required Settings:

---

[4]Note also that unless some pre-treatment 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 pre-treatment outcomes.

- `depvar` the outcome variable.

- `predictorvars` the list of predictor variables. See `help synth` help for more details.

For specifying the unit and time period of treatment, there are two methods. Exactly one of these is required.

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

- `d(`*varname*`)`. The `d` variable should be 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.

Options:

- `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 pre-treatment period.

- `pre_limit_mult(`$real \geq 1$`)` will not include placebo effects in the pool for inference if the match quality of that control (pre-treatment RMSPE) is greater than *pre_limit_mult* times the match quality of the treated unit.

- `training_propr(`$0 \leq real \leq 1$`)` instructs `synth_runner` to automatically generate the outcome predictors. The default (0) is to not generate any (the user then includes the desired ones in `predictorvars`). If set to a number greater than 0, then that initial proportion of the pre-treatment 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 pre-treatment period outcome variables will be used as predictors. This will make other covariate predictors redundant.

- `ci` outputs confidence intervals from randomization inference for raw effect estimates. These should only be used if the treatment is randomly assigned. If treatment is not randomly assigned then these confidence intervals do not have the standard interpretation (see above).

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

- `gen_vars` generates variables in the dataset from estimation. This is only allowed if there is a single period in which unit(s) enter treatment. These variables are required for `single_treatment_graphs` and `effect_graphs`. If `gen_vars` is specified, it will generate the following variables:

6

- **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 in effect is 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 pre-treatment match quality in terms of RMSPE. It is constant for a unit.

- **post_rmspe** contains a measure of the post-treatment 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 pre-treatment period outcome is 1.

- *depvar*_scaled_synth (if the match was done on trends) is the unit's synthetic control (scaled)

- **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.

- **n_pl_avgs**(*string*) controls the number of placebo averages to compute for inference. The total possible grows exponentially with the number of treated events. If omitted, the default behavior is 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.

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

- **noenforce_const_pre_length**: When there are multiple periods, estimations at later treatment dates will have more pre-treatment history available. By default, these histories are trimmed on the early side so that all estimations have the same amount of history. If instead, maximal histories are desired at each estimation stage, use **noenforce_const_pre_length**.

- **parallel** will enable parallel processing if the **parallel** command is installed and configured. Version 1.18.2 is needed at a minimum[5].

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

---

[5]At the time of writing, SSC does not contain a new enough version. Newer versions are available via the development website: https://github.com/gvegayon/parallel/.

- `pred_program(`*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_program` could be used to dynamically include outcome predictors. See Example 3 for usage details.

- `drop_units_prog(`*string*`)` is 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. Commonly this is because they are neighboring or interfering units. See Example 3 for usage details.

- `xperiod_prog(`*string*`)` allows for setting of synth's `xperiod` option that 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 of synth's `mspeperiod` option that 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* are pass-through options sent to `synth`. See `help synth` for more information. The following are disallowed: `counit`, `figure`, `resultsperiod`.

3.3   Saved Results

`synth_runner` returns the following scalars and matrices:

Scalars

`e(n_pl)`	The number of placebo averages used for comparison.
`e(pval_joint_post)`	The proportion of placebos that have a post-treatment RMSPE at least as large as the average for the treated units.
`e(pval_joint_post_t)`	The proportion of placebos that have a ratio of post-treatment RMSPE over pre-treatment RMSPE at least as large as the average ratio for the treated units.
`e(avg_pre_rmspe_p)`	The proportion of placebos that have a pre-treatment RMSPE at least as large as the average of the treated units. The farther this measure is from 0 towards 1, the better the relative fit of the treated units.

e(avg_val_rmspe_p)	When specifying `training_propr`, this is the proportion of placebos that have a RMSPE for the validation period at least as large as the average of the treated units. The farther this measure is from 0 towards 1, the better the relative fit of the treated units.

Matrices

e(treat_control)	The average treatment outcome (centered around treatment) and the average of the outcome of those units' synthetic controls for the pre- and post-treatment periods.
e(b)	A vector with the per-period effects (unit's actual outcome minus the outcome of its synthetic control) for post-treatment periods.
e(pvals)	A vector of the proportions of placebo effects that are at least as large as the main effect for each post-treatment period.
e(pvals_std)	A vector of the proportions of placebo standardized effects that are at least as large as the main standardized effect for each post-treatment period.
e(failed_opt_targets)	Errors when constructing the synthetic controls for non-treated 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 (e.g., `ssc install synth, all`). This panel dataset contains information for 39 US States for the years 1970–2000 (see Abadie et al. 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 use in Abadie et al. 2010):

9

```
. 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
|                                          | Total: 38
.................................| 60.00s 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, p-values (psuedo t-stats)

            | estimates       pvals   pvals_std
-------------+-------------------------------
      lead1 | -7.887098    .1315789           0
      lead2 | -9.693599    .1842105           0
      lead3 |  -13.8027    .2105263           0
      lead4 |   -13.344    .1315789           0
      lead5 |  -17.0624    .1052632           0
      lead6 |  -20.8943    .0789474           0
      lead7 |  -19.8568    .1315789    .0263158
      lead8 |  -21.0405    .1578947           0
      lead9 |  -21.4914    .1052632    .0263158
     lead10 |  -19.1642    .1842105    .0263158
     lead11 |   -24.554    .1052632           0
     lead12 |  -24.2687    .1052632    .0263158
```

The program notes progress towards estimating prediction errors and for conducting inference. Results for post-treatment 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 et al. 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) =  .1315789473684211
e(pval_joint_post_s
   td)                =  0
   e(avg_pre_rmspe_p) =  .9210526315789474

macros:
        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
. di (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 post-treatment RMSPE at least as great as the treated unit. While it is no quite significant at the 10 percent level, it does not take into account pre-treatment match quality. The `e(pval_joint_post_std)` value lists the same proportion, but scales all values by the relevant pre-treatment RMSPE. This measure does show significance. The final measure, `e(avg_pre_rmspe_p)`, is a diagnostic measure and it notes 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 one to the numerator and denominator (in cases with a single treatment). This is shown for the case of the ratio of post- to pre-treatment RMSPE.

Next we create common synthetic control graphs. Note that both `effect_graphs` and `single_treatment_graphs` require variables generated from the `gen_var` 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 it's synthetic control while the other plots the difference between the two (which for post-treatment is the "effect"). The two previous graphing commands allow the option `trlinediff(real)` that 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 post-treatment period) to 0 and the default value is -1 (to match Abadie et al. 2010). The `pval_graphs` command creates the graphs in Figure 3. These plot the p-values per-period for post-treatment periods for both raw and standardized effects.

```
. single_treatment_graphs, depvar(cigsale) trunit(3) trperiod(1989) ///
   trlinediff(-1) effects_ylabels(-30(10)30) effects_ymax(35) effects_ymin(-35)
. effect_graphs , depvar(cigsale) depvar_synth(cigsale_synth) ///
    trunit(3) trperiod(1989) trlinediff(-1) effect_var(effect)
. pval_graphs
```

◁

▷ **Example 2**

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

```
. gen 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
. // Proportion of control units that have a higher RMSPE than the
. //treated unit in the validtion period:"
```
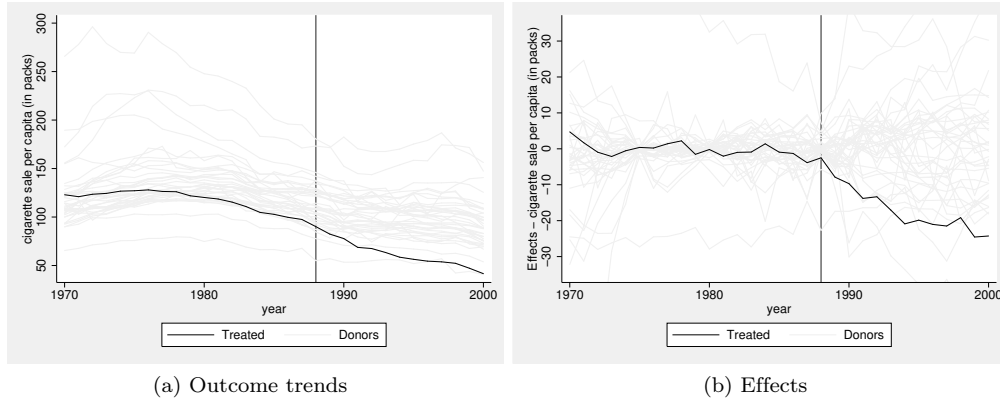
11

Figure 1: Graphs from `single_treatment_graphs`

(a) Outcome trends

(b) Effects

Figure 2: Graphs from `effect_graphs`

(a) Effect

(b) Treatment vs Control

Figure 3: Graphs from `pval_graphs`

(a) *p*-values

(b) *p*-values (psuedo t-stats)
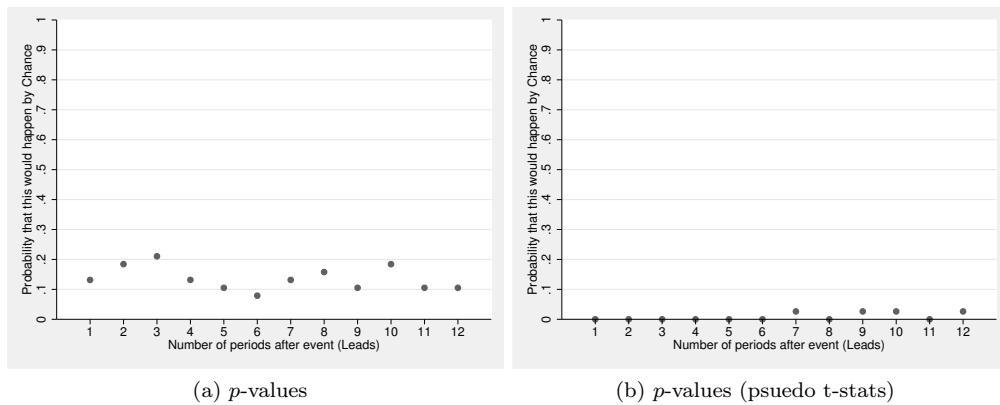
Figure 4: Graphs from `single_treatment_graphs`

(a) Outcome trends
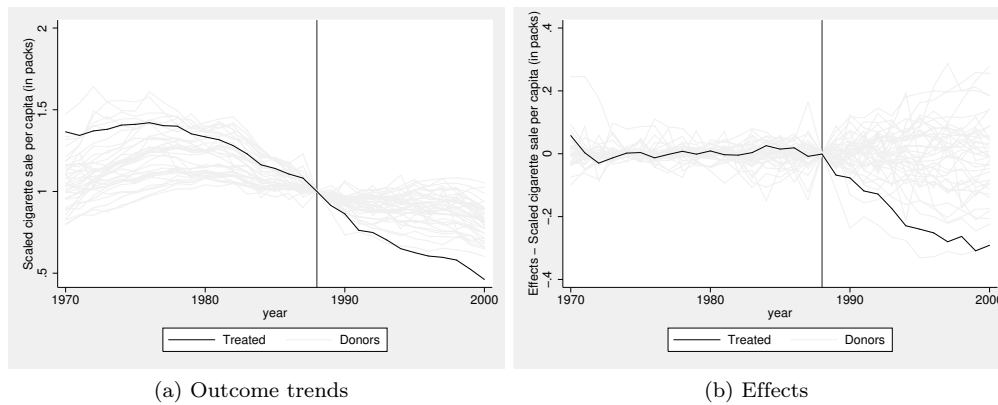
(b) Effects

```
. di round(`e(avg_val_rmspe_p)´, 0.001)
842
. single_treatment_graphs, depvar(cigsale_scaled) ///
    effect_var(effect_scaled) trunit(3) trperiod(1989)
. effect_graphs , depvar(cigsale_scaled) trunit(3) trperiod(1989) ///
    effect_var(effect_scaled) depvar_synth(cigsale_scaled_synth)
. pval_graphs
```

Again there is a single treatment period, so synthetic control data can be merged into our main dataset. In this setting we (a) specify the treated units/periods with a binary variable, (b) generate the outcome predictors automatically using the initial 13 periods of the pre-treatment era (the rest is the "validation" period), (c) we match on trends, and (d) during inference we limit the control units to those with pre-treatment match quality no more than 10 times worse than the match quality of the corresponding treatment units. Now that we have a training/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 the lines converge on the last pre-treatment period as that is the unit that all are standardized by (and all the synthetic controls then match their real units and have zero prediction error).

◁

▷ **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 comparison set due to concerns about treatment spillovers.

```
. cap program drop my_pred my_drop_units my_xperiod my_mspeperiod
. program my_pred, rclass
.   args tyear
```

Figure 5: Graphs from `effect_graphs`

(a) Effect

(b) Treatment vs Control
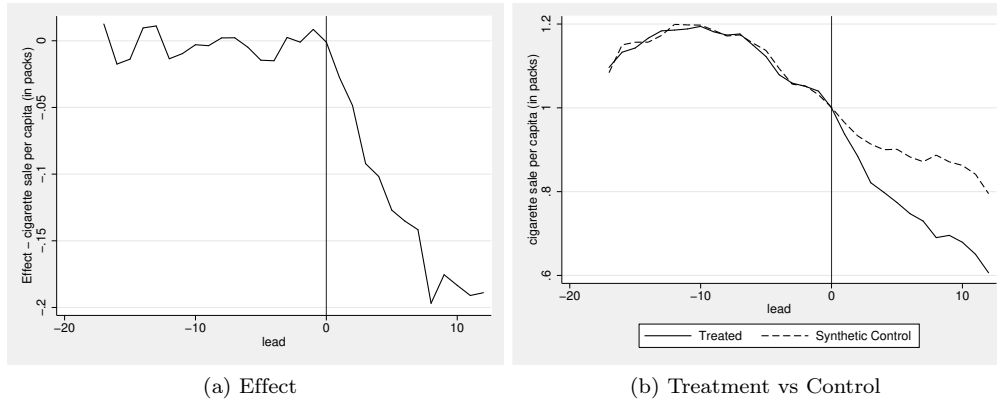
```
. return local predictors "beer(`=`tyear´-4´(1)`=`tyear´-1´) ///
        lnincome(`=`tyear´-4´(1)`=`tyear´-1´)"
. end
. program my_drop_units
.   args tunit
.   if `tunit´==39 qui drop if inlist(state,21,38)
.   if `tunit´==3 qui drop if state==21
. end
. program my_xperiod, rclass
.   args tyear
.   return local xperiod "`=`tyear´-12´(1)`=`tyear´-1´"
. end
. program my_mspeperiod, rclass
.   args tyear
.   return local mspeperiod "`=`tyear´-12´(1)`=`tyear´-1´"
. end

. gen 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)
. effect_graphs , multi depvar(cigsale)
. 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 can not automatically include all the synthetic control outputs back into the main dataset. Some of the graphs (of `single_treatment_graphs`) can also no longer be made. The option *multi* is now passed to `effect_graphs` and those 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`.

◁

14

4   Discussion

The Synthetic Control Methodology (SCM) (Abadie and Gardeazabal, 2003; Abadie et al., 2010) allows researchers to quantitatively estimate effects in many small sample settings in a manner grounded by theory. This article provides an overview of the theory of SCM and the `synth_runner` package, which builds on the `synth` package of Abadie et al. (2010). `synth_runner` provides tools to help with the common tasks of estimating 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 (a) extends the initial estimation strategy to allow for multiple units that receive treatment (at potentially different times), (b) allows for matching on trends in the outcome variable rather than on the level, and (c) automates the process of splitting pre-treatment periods into "training" and "validation" sections. It also provides graphs of diagnostics, inference, and estimate effects.

5   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(490): 493–505.

———. 2015. Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495–510.

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

Cavallo, E., S. Galiani, I. Noy, and J. Pantano. 2013. Catastrophic Natural Disasters and Economic Growth. *Review of Economics and Statistics* 95(5): 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.