allsynth: (Stacked) Synthetic Control Bias-Correction Utilities for Stata

Justin C. Wiltshire[†]

Abstract

Synthetic control methods are widely-used for estimating counterfactuals and treatment effects of policy interventions. **allsynth** adds greatly-enhanced functionality to the user-written **synth** module for Stata, which is widely used by practitioners to implement the "classic" synthetic control estimation strategy. **allsynth** automates implementation of several extensions to the classic approach while retaining the syntax of **synth**. These include a bias-correction procedure that adjusts for differences in the predictor variable values between a treated unit and its synthetic control donors, as well as extensions of the classic and bias-corrected estimators to environments with many treated units and treatment periods (a "stacked" synthetic control estimator) and to in-space placebo treatment estimation for randomization inference. **allsynth** also provides enhanced automated graphing capability and thorough diagnostics to help users with implementation. **allsynth** version 1.0 can be installed by typing `ssc install allsynth, replace all` in Stata's command line.

[†]University of California, Davis, One Shields Avenue, Davis, CA 95616, USA. Email: jcwiltshire@ucdavis.edu. This project has benefited greatly from discussions with participants of the UC Davis Econometrics Reading Group and Applied Micro brown bag, as well as participants at the 2021 Stata Conference. I am particularly grateful to the numerous members of the research community who have beta-tested the package and have communicated with me about issues and to identify areas of possible improvement. All errors remain my own. User feedback is appreciated.

1 Introduction

Synthetic control methods (SCMs) (Abadie and Gardeazabal, 2003; Abadie, Diamond, and Hainmueller, 2010, 2015; Abadie, 2021) have gained widespread popularity across the social sciences as an empirical strategy for estimating counterfactuals and treatment effects of policy interventions. For any unit that receives an intervention (or "treatment"), and given an outcome variable of interest observed for several time periods pre- and post-treatment in the "treated" unit and several similar "untreated" units, SCMs forecast the trajectory of the outcome variable in the *absence* of treatment. A "synthetic control" is a weighted average of the untreated "donor pool" units, constructed so that the pre-treatment characteristics of the synthetic control resemble those of the treated unit—including and especially the (generally non-linear) trajectory of the outcome variable. Under fairly general conditions, the dynamic path of synthetic control estimates of the outcome variable are a plausible estimate of the counterfactual trajectory of the outcome variable absent treatment, allowing straightforward calculation of estimated treatment effects. In any given post-treatment period, the estimated treatment effect on a treated unit is the difference between a treated unit's outcome value and that of its estimated synthetic control.

Synthetic controls are widely estimated by practitioners using the user-written **synth** command for Stata (Abadie, Diamond, and Hainmueller, 2010). **synth** is highly effective for this purpose but offers no automated way to conduct inference, to create post-estimation figures, or to implement many of the extensions that have been proposed in the synthetic control literature since **synth** was last updated in 2014. As a result, the process of implementing these steps can be quite time-consuming for practitioners and opens up possibilities for output to be affected by coding errors. The **synth_runner** utilities package for Stata (Galiani and Quistorff, 2017) greatly extends the functionality of **synth** and automates a number of additional procedures; however, **synth_runner** can often frustrate practitioners with errors that are difficult to diagnose. Moreover, there are several more-recent synthetic control innovations which the package does not automate.

The **allsynth** package for Stata is a wrapper for **synth** that greatly extends the functionality of **synth** in several ways, and offers thorough diagnostics to help diagnose errors that arise. The primary extensions are:

i) Automated estimation of "bias-corrected" synthetic control gaps (Abadie and L'Hour, 2021; Ben-Michael, Feller, and Rothstein, 2021; Abadie, 2021), using regression-based adjustments to mitigate

bias in "classic" synthetic control estimates that results from discrepancies in predictor values between a treated unit and its donor pool units

ii) Automated estimation of RMSPE-ranked *p*-values based on in-space placebo treatments (Abadie, Diamond, and Hainmueller, 2015) for "classic" and "bias-corrected" specifications

iii) Automated estimation of (classic and bias-corrected) synthetic control average treatment effects and RMSPE-ranked *p*-values from many treated units and potentially staggered treatment timing, variously referred to as a "pooled" or "stacked" synthetic control estimating strategy (Dube and Zipperer, 2015; Wiltshire, 2021)

iv) Greatly expanded automated graphing capability

Section 2 of this paper briefly reviews the formal setup and practical considerations when implementing SCMs and conducting inference. Section 3 reviews the capabilities and syntax of **allsynth** version 1.0. Section 4 walks through 13 illustrative examples that demonstrate the use of **allsynth**. It should be noted that **allsynth** has been greatly expanded since the release of version 0.0.8 BETA, and some older syntax is deprecated and occasionally even non-functional.

2 Synthetic Control Estimators

2.1 Setup and Practical Considerations

Synthetic control methods (SCMs) were originally developed for use with a single treated unit (Abadie and Gardeazabal, 2003; Abadie, Diamond, and Hainmueller, 2010, 2015). More recently, they have been extended to situations with many treated units and staggered treatment timing (Cavallo et al., 2013; Dube and Zipperer, 2015; Acemoglu et al., 2016; Kreif et al., 2016; Galiani and Quistorff, 2017; Abadie and L'Hour, 2021; Abadie, 2021; Ben-Michael, Feller, and Rothstein, 2022; Peri, Rury, and Wiltshire, 2021; Powell, 2021; Wiltshire, 2021).

There are several flavors of this latter type of SCM. I here expound the "stacked" (or "pooled") approach (Dube and Zipperer, 2015; Wiltshire, 2021), which estimates average treatment effects as (possibly weighted) means over many treated units. The "stacked" approach nests the case with a single treated unit

and highlights important considerations which may easily be addressed with **allsynth**. The end of Section 2.2 walks through a visual example of the steps of the synthetic control estimation procedure.[1]

Suppose we observe data for $I+J$ units over t calendar periods, where units $j = 1,...,I$ are "treated" and $j = I+1,...,I+J$ are untreated "donor pool" units. Each of the donor pool units is otherwise similar to at least one treated county, and selection into treatment or non-treatment was as good as random—that is, selection into treatment or the donor pool was plausibly not due to unobserved variables that are correlated with any outcome of interest. Moreover, there are plausibly no spillover effects of treatment onto the donor pool units–that is, the Stable Unit Treatment Value Assumption (SUTVA) is plausibly satisfied.[2]

For each treated unit $i = j \leq I$ and for at least some donor pool units, an outcome of interest, Y_{jt}, is observed for a sufficiently large number of pre-treatment periods before i received treatment in calendar period $T_{0i}+1$, and for a strictly positive number of post-treatment periods through calendar period $T_i \geq T_{0i}+1$. In addition, we observe k specified predictors of that outcome for all treated and donor pool units. These can include linear combinations (in time) of the outcome variable as well as selected covariates, each in some subset of pre-treatment periods. The vector $\mathbf{X}_j = (X_{1,j},...,X_{k,j})'$ contains j's values of these predictors, and the matrix $\mathbf{X}_0 = [\mathbf{X}_{I+1},...,\mathbf{X}_{I+J}]$ contains these vectors for the donor pool.

Define Y_{jt}^N as j's potential outcome in t with $\{N\}$o intervention, and Y_{jt}^{Int} as j's potential outcome in t with an $\{Int\}$ervention. The estimated marginal treatment effect (or "gap") of interest for j in t is:

$$\tau_{jt} = Y_{jt}^{Int} - Y_{jt}^N \tag{1}$$

As we observe $Y_{it}^{Int} = Y_{it}$ in $t > T_{0i}$ for each treated unit i, estimation of τ_{it} only requires an estimate of Y_{it}^N (that is, of the counterfactual value of Y_i in t). The synthetic control estimator for Y_{it}^N is a weighted average of the outcome values of the donor pool units:

$$\hat{Y}_{it}^N = \sum_{j=I+1}^{I+J} \hat{w}_j^i Y_{jt} \;\; \forall \;\; t \tag{2}$$

Given a set of weights on the k predictors which determine their relative importance, $v_1^i,...,v_k^i$,[3] the weights $\hat{\mathbf{W}}^i = (\hat{w}_{I+1}^i...,\hat{w}_{I+J}^i)'$ minimize the distance between i and its donor pool for some norm—typically, the

[1] Practitioners should see Abadie (2021) for a comprehensive review of the synthetic control literature.

[2] Abadie (2021) discusses a complete set of contextual considerations for synthetic controls.

[3] Practitioners should note there are several methods of estimating these v_h^i weights, including minimizing the mean squared prediction error (*MSPE*) over the entire pre-treatment period or the cross-validation approach adopted in Abadie, Diamond, and Hainmueller (2015).

Euclidian norm:

$$\left(\sum_{h=1}^{k} v_h^i (X_h i - w_{I+1}^i X_{hI+1} - ... - w_{I+J}^i X_{hI+J})^2 \right)^{1/2} \tag{3}$$

subject to $\sum_{j=I+1}^{I+J} w_j^i = 1$ and to $w_j^i \geq 0 \ \forall \ j \in \{I+1, ..., I+J\}$, where the non-negativity constraint prevents against extrapolation bias.[4]

In cases where treatment is simultaneous in calendar time, $T_{0i} + 1 = T_0 + 1 \ \forall \ i$, the trajectory of average treatment effects on the treated units (ATTs) can be estimated in calendar time or event time, where all i are treated in event time $e(T_0 + 1) = 0$. The ATTs, $\overline{\tau} = (\overline{\tau}_0, ..., \overline{\tau}_{\overline{E}})$, can only be estimated in event time when treatment adoption is staggered, where i is treated in event year $e(T_{0i} + 1) = 0$ and $e \leq \overline{E}$, and each $\overline{\tau}_e$ is a weighted average of the marginal treatment effects in e. Then the ATT on an outcome of interest in e is:

$$\begin{aligned} \overline{\tau}_e &= \sum_{i=1}^{I} \gamma_i \tau_{ie} \\ &= \sum_{i=1}^{I} \gamma_i (Y_{ie} - Y_{ie}^N) \end{aligned} \tag{4}$$

with some weights $\gamma_i \geq 0 \ \forall \ i$ and $\sum_i^I \gamma_i = 1$ on the treated units.

It is generally good practice to estimate $\overline{\tau}_e$ for a panel balanced across $e \in [\underline{E}, \overline{E}]$—that is, to restrict the treated units to those $i \in I'$ that are observed for at least $-\underline{E}$ event periods before treatment and at least $\overline{E} + 1$ event periods after treatment including the treatment period $e = 0$ (where $\underline{E} < 0$ and $\overline{E} > 0$). It is also generally good practice to specify a common set of predictors for all treated units. When treatment adoption is staggered, users should consider whether the commonality of the predictors is best-specified in calendar time or event time. For various reasons, it may additionally be desirable to uniquely restrict the donor pool for each i to those $j \in J_i$.

It may also be appropriate to transform the outcome variable *prior to estimation*, whether e.g. to time differences ($Y'_{jt} = \Delta Y_{jt} = Y_{jt} - Y_{jt-1}$) or growth rates ($Y'_{jt} = 100 \times \Delta Y_{jt} / Y_{jt-1}$) to ensure that for each i the pre-treatment trajectory of the outcome can be approximated by a convex combination of donor pool units (Abadie, 2021); by demeaning ($Y'^{\,i}_{jt} = Y_{jt} - \overline{Y}_j$, where $\overline{Y}_j = \sum_t^{T_{0i}} Y_{jt}$ is the pre-treatment average of Y_{jt} over all of i's pre-treatment periods) to attenuate bias resulting from unit-level, time-variant unobserved confounds

[4] Some extensions allow negative weights to gain certain benefits at the cost of permitting extrapolation bias—for example, Doudchenko and Imbens (2016); Ben-Michael, Feller, and Rothstein (2021).

when the pre-treatment fit is imperfect (Ferman and Pinto, 2021); or by normalizing for each i and all its donor pool units to the final pre-treatment period for i ($Y'^{i}_{jt} = 100 \times Y_{jt}/Y_{jT0i}$) to ensure the estimated marginal treatment effects, $\hat{\tau}_{ie}$, are measured in deviations from a common null just prior to treatment, in units that are meaningfully comparable such that their average, $\hat{\bar{\tau}}_e$, retains maximum interpretability (Wiltshire, 2021).

2.2 Bias Correction

Another important consideration is that bias may be present in the synthetic control estimated marginal treatment effects (and thus the estimated average treatment effect estimates) because of discrepancies between the predictor variable values in each treated unit and its synthetic control donors. Abadie and L'Hour (2021) and Ben-Michael, Feller, and Rothstein (2022) propose a bias-correction procedure to address this, analogous to the approach proposed in (Abadie and Imbens, 2011) to address inexact matching on predictor variables with matching methods. Note that this will *not* necessarily improve the pre-treatment fit of the outcome variable, but rather addresses discrepancies between a treated unit and its donor pool in the values of all specified linear combinations of predictor variables, including the covariates.

This synthetic control bias correction procedure is implemented as follows: for each i, the $\hat{\mathbf{W}}^i$ are first obtained from synthetic control estimation on the uncorrected outcome values, Y_{jt} (or Y'_{jt} or Y'^{i}_{jt} if the outcome variable has been transformed). Let $\hat{\mu}^i_{0t}(x)$ be a predictor of Y_{it} given $X_i = x$, estimated by first using only i's donor pool to (possibly nonparametrically) regress Y_t on the complete set of predictor variables (including all specified linear combinations of the outcome in the pre-treatment period). Then $\tilde{Y}_{it} = Y_{it} - \hat{\mu}^i_{0t}(X_i)$ and $\tilde{\hat{Y}}^N_{it} = \sum_{j=I+1}^{I+J} \hat{w}^i_j \left(Y_{jt} - \hat{\mu}^i_{0t}(X_j)\right)$ can be calculated, admitting (1'), the bias-corrected version of (1), $\hat{\tau}_{BC_{it}} = \tilde{Y}_{it} - \tilde{\hat{Y}}^N_{it}$. These can be converted to event time (if appropriate) and combined to inform the estimator of the bias-corrected average treatment effects on the treated for each e:

$$\begin{aligned} \hat{\bar{\tau}}_{BC_e} &= \sum_{i=1}^{I} \gamma_i \hat{\tau}_{BC_{ie}} \\ &= \sum_{i=1}^{I} \gamma_i (\tilde{Y}_{ie} - \tilde{\hat{Y}}^N_{ie}) \end{aligned} \tag{5}$$

Note that this nests the "classic" synthetic control estimator for (1) with a single unit, $I = 1$, when the

estimated bias from predictor variable discrepancies is or is presumed zero.

This synthetic control estimation procedure with a single treated unit is visually demonstrated in Panels A–F of Figure 1, recreating the classic example from Abadie, Diamond, and Hainmueller (2010) looking at the effect of California's 1989 excise tax increase on per-capita cigarette sales. Panel A shows per-capita cigarette sales in California over time, while Panel B shows the same also for the 38 states in the donor pool for California. The SCM selects a subset of these donor pool states to receive strictly positive weights in the construction of the synthetic California; Panel C shows per-capita cigarette sales for only California and these donor pool states the synthetic California. Panel D shows the same again for only California and its synthetic control, which is a convex combination of those positively-weighted donor pool units given the weights (Panel D re-creates Figure 2 from Abadie, Diamond, and Hainmueller (2010)). Panel E shows the classic synthetic control estimated gap in each year (re-creating Figure 3 from Abadie, Diamond, and Hainmueller (2010). These are not from exactly the same specification used in Abadie, Diamond, and Hainmueller (2010), so the plot is highly-similar but not identical. Panel F shows both the classic and the bias-corrected estimated gaps.

2.3 Inference

Large-sample inferential approaches are generally not appropriate with synthetic control methods when the donor pool is small. Abadie, Diamond, and Hainmueller (2010, 2015) propose an exact inferential procedure for SCMs based on "in-time" or "in-space" "placebo" (or "falsification") tests in which treatment status is randomly permuted across, respectively, pre-treatment time periods or untreated units in a sample. Calculation of p-values based on in-space placebo tests remains the most widely adopted inferential approach and has spawned several extensions (Cavallo et al., 2013; Ferman and Pinto, 2017; Firpo and Possebom, 2018; Abadie and L'Hour, 2021), though it should be noted that the approach may lead to size distortions (Hahn and Shi, 2017; Ferman and Pinto, 2017) and in the "stacked" case (with many treated units) may also be under-powered (Zhang, 2019).[5]

allsynth can calculate ranked *RMSPE* p-values based on in-space placebo tests using classic and bias-

[5] Alternative inferential procedures continue to be proposed by the growing SCM literature (for example, (Dube and Zipperer, 2015; Doudchenko and Imbens, 2016; Hahn and Shi, 2017; Chernozhukov, Wuthrich, and Zhu, 2019), but the literature has not converged on a preferred alternative.

corrected synthetic control estimators, with either a single treated unit or when stacking many treated units. This involves several steps, and I here expound the stacked, bias-corrected case for maximum generality and for its relevance to **allsynth**.

For each treated unit i, treatment must be permuted across all untreated units in i's donor pool, $j \in J_i$, with i and the remaining untreated units collectively constituting the donor pool for each $j \in J_i$. (1) or (1') must then be estimated possibly many times for each $j > I$—once for each i for which j is part of i's donor pool J_i. This admits the the set of dynamic paths of (here, bias-corrected) estimated placebo gaps, $\hat{\tau}^i_{BC_j}$, for all $j > I$, one for each i such that $j \in J_i$.

The next step is to calculate the estimated placebo average gaps, with each constructed as a weighted mean in each period (event or calendar time) using the $\hat{\tau}^i_{BC_j}$ of exactly one j from each J_i, weighted by the same γ_i used to estimated (5). The averaging is across i, but as there are many $j \in J_i$ for each i there are consequently very many placebo average gaps, with the number ballooning rapidly in J and especially in I: the number of average placebo gaps is $N_G = \prod_{i=1}^{I} J_i$. With even modestly-sized J_i and I, it is clear that it quickly becomes infeasible to estimate every placebo average gap. Consequently, when $I > 1$ it is much more practical to randomly sample some subset of placebo averages, $S < N_G$. We can plot the estimated *actual* average treatment effect, $\hat{\bar{\tau}}^0_{BC_e}$, and each placebo average, $\hat{\bar{\tau}}^s_{BC_e}$ with $s = 1, 2, ..., S$, to visually compare $\hat{\bar{\tau}}^0_{BC_e}$ to the (sample) permutation distribution of placebo gaps. We can also calculate the ratio of the mean squared prediction error (RMSPE) as the post-treatment MSPE through each $E \in \{0, 1, ..., \overline{E}\}$ (if in event time) over the pre-treatment MSPE for each $s = 0, 1, ..., S$:

$$RMSPE_{s_E} = \frac{\sum_{e=0}^{E} \hat{\bar{\tau}}^s_{BC_e}{}^2/(E+1)}{\sum_{e=-1}^{\underline{E}} \hat{\bar{\tau}}^s_{BC_e}{}^2/(-\underline{E})} \tag{6}$$

which normalizes the size of the average (squared) post-treatment gaps by a measure of pre-treatment fit, discounting those estimated averages with noisy pre-treatment trajectories. We can then use the empirical distribution of these $RMSPE_s$ test statistics to calculate a p-value for each E based on the share of $RMSPE_{s_E} \geq RMSPE_0$ in E:

$$p_{RMSPE_E} = \frac{\sum_{s=1}^{S} \mathbb{1}[RMSPE_{s_E} \geq RMSPE_{0_E}]}{S+1} \tag{7}$$

Again, note that this setup nests the more restrictive "classic" setup wherein $I = 1$ and the estimated bias

from predictor variable discrepancies is or is presumed zero for all units.

2.4 Uniqueness of the $\hat{\mathbf{W}}^i$ Matrices

A final consideration of note for all practitioners of SCMs is whether the $\hat{\mathbf{W}}^i$ weighting matrices are unique, which in most cases is effectively equivalent to asking whether the values of the predictor variables for any Y_i lie outside the convex hull of those values among the donor pool units. If they do not (that is, if any $\hat{\mathbf{W}}^i$ is not unique), then the corresponding estimated synthetic control is not unique and in fact there are generally infinitely many solutions to (3). The practical consequence of non-uniqueness can be large interpolation biases.

Abadie and L'Hour (2021) observe that, when I is small, non-uniqueness of $\hat{\mathbf{W}}^i$ is rare as a result of the curse of dimensionality. If non-uniqueness is known to be an issue when I is small, it can generally be addressed by further restricting the donor pool to those untreated units that have predictor values most similar to those of the treated unit (provided each $J_i \geq k+2$ is retained if the estimates are to be bias-corrected) or by adjusting the number of covariates k;[6] thus it is valuable for practitioners to know if $\hat{\mathbf{W}}^i$ is unique when I is small. When I is large, non-uniqueness is more likely. When this obtains, ad-hoc fixes may be impractical and practitioners may prefer to simply drop all i without a unique solution—though it should be noted that this may have consequences for the interpretation of the estimated parameters. Practitioners may prefer to address such an issue through e.g. the introduction of a penalty on pairwise matching discrepancies (Abadie and L'Hour, 2021), though such measures are beyond the scope of **allsynth**.

3 Using allsynth

allsynth version 1.0 allows SCM practitioners to easily implement most of the processes described above. To use **allsynth**, the Stata packages **synth**, **distinct**, and **elasticregress** must be installed on the same system. **allsynth** has been tested and can be confirmed to work on Stata 15.1 and higher.[7]

As with **synth**, `tsset` *`panelvar timevar`* must be specified prior to calling **allsynth**, with the *`panelvar`*

[6] Abadie and L'Hour (2021) specifically suggest increasing k as a solution in such cases. In practice, in such cases a unique solution may occasionally also be successfully sought by decreasing k or by adjusting the pre-treatment period over which MSPE is minimized.

[7] Users with older versions of Stata are invited to contact the author at jcwiltshire@ucdavis.edu to test **allsynth** in those cases.

and *timevar* variables both containing only integer values.

3.1 allsynth Syntax

`allsynth` *depvar predictorvars*`, trunit(#) trperiod(#) [`*synth_options*
`pvalues bcorrect(`*string*`) gapfigure(`*string*`) transform(`*string*`) stacked(`*string*`) ]`

The syntax allows for **allsynth** to be used in the exact same way as **synth**. That is, **synth** and **allsynth** can be used interchangeably when none of the additional options of **allsynth** are specified. See `help synth` for more on **synth** and its options. In addition, **allsynth** allows the options `pvalues, bcorrect(), gapfigure(), transform()`, and `stacked()`, the last four of which accept required and optional inputs.[8] *depvar*, *predictorvars*, `trunit(#)` and `trperiod(#)` are generally required—though if `stacked()` is specified (see Section 3.2.5), then `trunit()` and `trperiod()` may be omitted.

Note that `[...]` indicates optional specifications, while `A|B` indicates at least one (sometimes exactly one) of `A` or `B` must be specified. Users should *not* include the symbols `[ , ]` , or `|`.[9]

- *depvar* is the outcome variable, which must be observed for all units in all periods if `mspeperiod()` is not specified, or in all periods in `mspeperiod()` if `mspeperiod()` *is* specified.
- *predictorvars* is the list of predictor variables including desired linear combinations thereof.
- `trunit(#)` identifies the treated unit from the specified *panelvar*.
- `trperiod()` identifies the treatment period from the specified *timevar*.

3.2 allsynth Options

3.2.1 pvalues

`pvalues` automatically estimates in-space placebo treatments across the donor pool units and calculates *RMSPE* *p*-values for each post-treatment period. `pvalues` must be specified for the placebo gaps to be

[8] **allsynth** version 0.0.8 BETA also had the option `placeboskeep`, which allowed users to specify that the estimated placebo gaps and associated data should be saved if the **synth** option `keep()` were specified. **allsynth** version 1.0 automatically saves these data if the `pvalues` option is specified, though this requires that `keep()` be specified if `pvalues` is specified.

[9] Except if desired when specifying `allsynth ..., stacked(..., ... donorcond() donorcond2() donorif())`. See Section 3.2.5.

plotted (see the entry on `gapfigure()` in Section 3.2.3. Specifying `pvalues` will greatly extend the run-time, and so it is assumed that the results should be saved. Accordingly, `keep()` must be specified.

3.2.2 bcorrect()

`bcorrect(`*`string`*`)` can be used to specify that the bias-corrected synthetic control estimates should be calculated. Note the classic synthetic control estimates are always retained. *There must be at least* $k+2$ *donor pool units* if `bcorrect()` is specified, where k is the number of specified predictors.

`bcorrect()` has its own syntax:

```
bcorrect(nosave|merge [ridge|lasso|elastic|posonly figure])
```

- Exactly one of `nosave` or `merge` must be specified if `bcorrect()` is specified. `merge` merges and saves the bias-corrected estimates to the specified `keep()` file, and requires that `keep()` be specified. `nosave` should be used if the bias-corrected estimates should *not* be merged with the specified `keep()` file, or if `keep()` is not specified. In general, `merge` should be specified.

bcorrect() options:

- Exactly one of `ridge`, `lasso`, `elastic` and `posonly` *may* be specified if the bias is to be estimated using (respectively) ridge regression, lasso regression, elastic net regression, or ordinary least squares (OLS) regression with only those donor pool units for which $\hat{w}_j^i > 0$. The default setting is to use OLS regression with all $j \in J_i$ to estimate the bias.
- `figure` specifies that the trajectories of the bias-corrected values of the outcome for the treated unit and its synthetic control ($\tilde{Y}_{it}$ and $\tilde{\hat{Y}}_{it}^N$) should be plotted. Most practitioners will have an interest only in the difference between these rather than these variables individually; however, visual examination of this plot may be instructive.

3.2.3 gapfigure()

`gapfigure(`*`string`*`)` can be used to automatically generate a plot of the trajectories of (at most two of) the estimated gaps, bias-corrected gaps, the set of placebo gaps, or the set of bias-corrected placebo gaps. The bias-corrected gaps and placebo gaps can only be plotted if the **allsynth** option `bcorrect()` is specified, and the placebo gaps can only be plotted if the **allsynth** option `pvalues` is specified. If `bcorrect()` is specified, only the bias-corrected gaps and placebo gaps can be plotted. The plot can also be customized and saved.

`gapfigure()` has its own syntax:

`gapfigure(classic|bcorrect [placebos lineback, save(`*`file`*`[, replace]) `*`twoway_options`*`])`

- At least one of `classic` and `bcorrect` must be specified if `gapfigure()` is specified, but `bcorrect` may only be specified if the **allsynth** option `bcorrect()` has been specified. `classic` plots the classic estimated gaps for the treated unit, and `bcorrect` plots the bias-corrected estimated gaps for the treated unit. They can also be plotted together.

gapfigure() options:

- `placebos` may only be specified if exactly one of `classic` and `bcorrect` is specified, and will plot the corresponding placebo gaps alongside the specified estimated gaps.
- `lineback` places a vertical dotted line on the plot in the final pre-treatment period. The default setting places a vertical dotted line on the plot in the treatment period.
- `save()` specifies that the plot should be saved to the indicated file, which may include a filepath and a filename. The default file extension is .pdf but another extension may be specified. `replace` may be specified after a comma if any identically-named file should be overwritten, as in `save(`*`file`*`, replace)`.

- *twoway_options* may be specified to modify the plot as desired (see `help twoway_options`), but note that titles of any kind must not be contained in quotations and will not display a comma if one is indicated.

3.2.4 transform()

`transform(`*string*`)` can be used to automatically transform specified variables in exactly one of two ways prior to synthetic control estimation: the variables can be demeaned over the pre-treatment period, or can be normalized to 100 in the final pre-treatment period. Only one transformation type may be specified.

`transform()` has its own syntax:

`transform(`*varlist*`, demean|normalize)`

- *varlist* indicates the variables that will be transformed. At least one variable must be specified.
- Exactly one of `demean` or `normalize` must be specified if `transform()` is specified. `demean` will (by unit) demean the specified variables over the entire pre-treatment period. `normalize` will (by unit) normalize all specified variables to 100 $\times$ the value in t divided by the value in i's final pre-treatment period: $X'^{,i}_{jt} = 100 \times X_{jt}/X_{jT0i}$. Note this necessitates that no variable in *varlist* have its value in the final pre-treatment period alone be specified as a predictor;[10] e.g. if the treatment period is 1989, then `transform(cigsale, normalize)` may not be specified if `cigsale(1988)` is specified as a predictor variable.

3.2.5 stacked()

`stacked(`*string*`)` can be used to automatically estimate the average treatment effect on the treated units using a stacked synthetic control estimating strategy. It allows users to specify if the estimates should be calculated in event time or calendar time (where either are possible), to specify whether estimates should be balanced in the specified time-type periods, to specify unit-specific weights for calculation of the averages,

[10] This is because the normalized value of that predictor in the final pre-treatment period will be constant across all units.

and to specify whether the averages should include only unit-estimates associated with unique $\hat{\mathbf{W}}^i$ matrices. It also allows users to specify conditions for selecting treated-unit-specific donor pools.

`stacked()` can also automatically generate a distribution of sampled average placebo gaps and calculate *RMSPE*-ranked *p*-values from that distribution, and allows users to specify the number of placebo average gaps included in the sample distribution.

Similarly to `gapfigure()` (see Section 3.2.3), `stacked()` can automatically generate a (customizable) plot of the trajectories of (at most two of) the estimated average gaps, bias-corrected average gaps, the set of placebo average gaps, or the set of bias-corrected placebo average gaps, all in event time, or in calendar time when units are treated simultaneously.

The **allsynth** option `keep(`*`filename`*`)` must be specified if `stacked()` is specified, and the treated-unit-specific results for each i will be saved in *`filename_panelvar_i.dta`*, while the estimated average gaps by time period will be saved in *`filename_ate.dta`*. If the **allsynth** option `pvalues` is specified, then the estimated average gaps, placebo gaps, *RMPSE*, and *RMSPE*-ranked *p*-values will be saved in *`filename_ate_distn.dta`*.

`stacked()` has its own syntax:

```
stacked(trunits(varname) trperiods(varname), clear [eventtime(numlist)
avgweights(varname) balanced donorcond(string [, string]) [
donorcond2(string [, string]) donorcond3(string [, string])
donorcond4(string [, string]) ] donorif(string) unique_w sampleavgs(real)
figure(classic|bcorrect [placebos lineback, save(file[, replace]) twoway_options])
])
```

Users should note that use of `stacked()` can dramatically increase run-time, especially if the **allsynth** option `pvalues` is specified (see Section 3.2.1), and increasingly as the number of treated units, I, and/or donor pool units, J_i for each $i \in I$, grows large.

- `trunits(`*`varname`*`)` is required, and *`varname`* must identify a dummy variable that, in all periods,

identifies the treated units with a 1 and the donor pool units with a 0.

- `trperiods(`*`varname`*`)` is required, and *varname* must identify an integer variable that, in all periods, contains the *timevar* period of treatment (assumed to be in calendar time) for every treated unit. If the treatment is simultaneous in *timevar* for all treated units, results will be displayed and saved in *timevar* periods (assumed to be calendar periods); otherwise, results will be displayed and saved in event time, with each treatment period converted to event period $e = 0$.

- `clear` is required and must be specified after a comma. `clear` is required because `stacked()` will clear all files with the file name specified in `keep()` from the specified directory or from the working directory if no directory is specified in `keep()`.

- The **allsynth** option `keep(`*`file`*`, replace)` is required (in *synth_options*) when `stacked()` is specified.

stacked() options:

- `eventtime(`*`numlist`*`)` must specify exactly two integers—one strictly negative, the other strictly positive—which identify the event-time window over which the final results should be displayed and saved. **No other symbols are allowed.** Note that `eventtime()` will not restrict the period over which the pre-treatment *MSPE* is minimized. `stacked()` will use the smallest event window out of that specified and that observed in any treated unit (or in a balanced sample across all treated units if the `stacked()` option `balanced` is specified). The default setting is the smaller of `eventtime(-5 5)` and the (possibly balanced) window observed over the treated units. If (a) treatment is not simultaneous in *timevar* or (b) treatment is simultaneous in *timevar* and `eventtime()` is specified, then results will be displayed and saved in event time. If treatment is simultaneous in *timevar* across all treated units and `eventtime()` is *not* specified, results will be displayed and saved in *timevar* periods (assumed to be calendar time).

- `avgweights(`*`varname`*`))` specifies a numeric variable that identifies the treated-unit weights, γ_i, to be used to calculated the (weighted) average treatment effects. For each treated unit i the weights must be non-missing and constant across all *timevar* periods.

- `balanced` specifies that the estimated average treatment effects (gaps) should be displayed and saved only for those event periods in which *every* treated unit is observed. This ensures common interpretability of the estimated average gap across retained event periods (this is true even if the results are displayed and saved in calendar time).

- `donorcond(`*`string`* `[, `*`string`*`])`), `donorcond2(`*`string`* `[, `*`string`*`])`), `donorcond3(`*`string`* `[, `*`string`*`])`), and `donorcond4(`*`string`* `[, `*`string`*`])`) permit users to temporarily modify the data while estimating all parameters associated with each treated unit i. This allows the user to set i-specific restrictions on the data for the purpose of restricting the donor pool for i to only those $j \in J_i$. `donorcond()`, ..., `donorcond4()` each permit a comma to separate two unique lines of code, such that up to eight total commands may be executed to set up the restriction of each J_i. These are in addition to the `keep if...` restriction which must be specified using `donorif()` if `donorcond()`, ..., `donorcond4()` are specified

- `donorif(`*`string`*`)` permits users to specify a condition under which the untreated units $j \in J$ should be kept in i's donor pool, J_i. `donorif(`*`string`*`)` imposes `keep if...` before the *`string`* entry, so users *should not* include "`keep if...`]] in *`string`*. `donorif()` only applies to untreated units, and treated unit i is always retained, so users should *not* specify `donorif(`*`string`*`)` to condition on being an untreated unit.

- `unique_w` specifies that only treated units i with unique $\hat{\mathbf{W}}^i$ matrices should be included in the estimated average treatment effect.

- `sampleavgs(`*`real`*`)` specifies an integer ≥ 30 which is the number of placebo average gaps that should be sampled from the population of $\prod_{i=1}^{I} J_i$ possibilities. The default setting is 100.

- `figure(classic|bcorrect [placebos lineback, save(`*`file`*`[, replace]) `*`twoway_options`*`])` can be used to automatically generate a plot of the trajectories of (at most two of) the estimated average gaps, average bias-corrected gaps, the set of placebo average gaps, or the set of average bias-corrected placebo gaps. It will plot these in the time type (calendar time or event time) of the estimates (see the entry for `eventtime()`, above). The bias-corrected gaps and placebo gaps can only be plotted if

the **allsynth** option `bcorrect()` is specified, and the placebo gaps can only be plotted if the **allsynth** option `pvalues` is specified. `figure()` uses the same syntax as the **allsynth** option `gapfigure()`, but unlike `gapfigure()` it is possible to specify `figure(classic placebos)` even if the **allsynth** option `bcorrect()` is specified. As with `gapfigure()`, the plot can also be customized and saved. See the syntax for `gapfigure()` in Section 3.2.3) for further explanation of the syntax for `figure()`.

3.3 Displayed, Saved, and Stored Results

3.3.1 Displayed Results

When **allsynth** is specified using only the options available using **synth**, the displayed output is the same except for a note that no bias correction or p-value calculations have been specified or provided, in case the users misunderstood that **allsynth** requires additional specifications to return those results. Note that when `bcorrect()` is specified, the variable names are temporarily changed by **allsynth**, and indicate either the outcome variable Y or a linear combination thereof, or one of the r covariates, X_m, where $m = 1, 2, ..., r$. Up to the first eight characters of the variable name are also retained.

When **allsynth** is specified with its unique options, the displayed output for a treated unit i includes an observation, for each value t in *timevar* of: the *panelvar* ID for i, the value of t, the estimated gap, and a dummy variable indicating whether the $\hat{\mathbf{W}}^i$ matrix is (likely) unique. Depending on the specification, the displayed output may also include the estimated bias-corrected gaps, or the number of units observed in *panelvar*, N. Additionally, for each $t > T_{0i}$ the displayed output may include (i) the *ratio* of the post-treatment over the pre-treatment *MSPE* (the *RMPSE*) through t, (ii) each unit's *RMSPE* rank in the empirical distribution of placebo *RMSPE*, (iii) the p-value through t, (iv) the equivalent of (i)–(iii) for the bias-corrected estimates.

3.3.2 Saved Results

When `stacked()` is *not* specified but `keep(`*file*`)` *is* specified, results are saved to *file.dta*, which may include a filepath before the filename. When `stacked()` *is* specified, `keep(`*filename*`)` *must be* specified, and the treated-unit-specific results for each i will be saved in *filename_panelvar_i.dta*. Depending on the **allsynth** options specified, several additional variables beyond those generated by **synth** will be

saved. The set of possible additional variables include: `gap`, `gap_bc`, `rmspe`, `rmspe_rank`, `rmspe_bc`, `rmspe_bc_rank`, `p`, `p_bc`, `N`, `unique_W`, `trunit`, `trperiod`, and `_stAvgweights`. All variables ending in `_bc` refer to the result from bias-corrected synthetic control estimation.

When `stacked()` *is* specified, the average estimated gaps or bias-corrected gaps will be saved to *filename_ate.dta* for each (calendar or event) time period contained in the variable `_tm`. If the **allsynth** option `pvalues` is also specified, then the estimated average gaps, placebo gaps, *RMPSE*, and *RMSPE*-ranked *p*-values will be saved in *filename_ate_distn.dta*. In this case, the sampled placebo averages are assigned an ID number ≥ 1 in the variable `_placeboID`, while the estimated average gap (or estimated average treatment effect) is assigned the `_placeboID` value of 0.

3.3.3 Stored Results

Beyond the stored results generated by **synth**, **allsynth** stores several estimation results depending on the options specified. Type `ereturn list` to see the full set of stored results. The set of possible additional stored matrices include `e(results)`, `e(gaps)`, `e(pvalues)` `e(unique_W)`. The `e(results)` matrix contains all additional stored results available. It should be noted that stored results for each treated unit except $i = I$ will be over-written if `stacked()` is specified, in which case users can find these results saved in *filename_panelvar_i.dta* for each treated unit *i*.

4 Examples

The **allsynth** help file (type `help allsynth` after installation) contains interactive (clickable on a Windows system) versions of these examples. For Stata to run these examples, ensure **synth** (Abadie, Diamond, and Hainmueller, 2010) has been installed with the ancillary files (i.e. type `ssc install synth, replace all`, then type `help synth` for details about **synth**), and ensure that **distinct** and **elasticregress** have been installed (i.e. type `ssc install distinct` and `ssc install elasticregress`).

Examples 1–10 make use of the panel data made available with the **synth** package and illustrate the use of **allsynth** when the `stacked()` option is not specified. Examples 11–13 make use of the panel data made available with the **allsynth** package and illustrate the use of **allsynth** with the `stacked()` options specified.

4.1 allsynth *without* the `stacked()` option

Examples 1–10 make use of the *synth_smoking* data included with the **synth** package (Abadie, Diamond, and Hainmueller, 2010)). Begin by loading these data and declare the dataset as a panel:

```
. sysuse synth_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

Use **allsynth** exactly as you would use **synth** to reconstruct the estimate from the synth help file (note: this is not the exact specification used in Abadie, Diamond, and Hainmueller (2010)):

```
. allsynth4 cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989)
```

This produces the same output as would have been produced if the **synth** command had been used in place of **allsynth**, as well as additional stored results in `e(results)`, `e(unique_W)`, and `e(gaps)`. If `keep(`*file*`)` had been specified, additional variables would also have been saved to *file*. **allsynth** additionally cautions that no bias correction or *p*-value calculations have been specified or provided.

The synthetic control estimation procedure is visually demonstrated in Panels A–E of Figure 1. Panel A shows per-capita cigarette sales in California over time, while Panel B shows the same also for the 38 states in the donor pool for California. The SCM selects a subset of these donor pool states to receive strictly positive weights in the construction of the synthetic California; Panel C shows per-capita cigarette sales for only California and these donor pool states the synthetic California. Panel D shows the same again for only California and its synthetic control, which is a convex combination of those positively-weighted donor pool units given the weights (Panel D re-creates Figure 2 from Abadie, Diamond, and Hainmueller (2010)). Panel E shows the classic synthetic control estimated gap in each year (re-creating Figure 3 from Abadie,

Diamond, and Hainmueller (2010). Again, note that these are not from exactly the same specification used in Abadie, Diamond, and Hainmueller (2010), so the plots is nearly-but-not-quite identical.

Example 2

Use **allsynth** exactly as you would use **synth**:

```
. allsynth4 cigsale beer retprice cigsale(1980), trunit(3) trperiod(1989)
```

These results could also have been realized using the **synth** command in place of **allsynth**, but the reduction in dimensionality of the predictor variables (relative to Example 1) has resulted in the synthetic control optimization estimating a estimated $\hat{\mathbf{W}}^i$ with more non-zero weights than predictor variables, which is likely not unique. The produced output still includes what would have been produced if the **synth** command had been used in place of **allsynth**, as well as additional stored results in e(results), e(unique_W), and e(gaps). Unlike **synth**, however, **allsynth** cautions that the $\hat{\mathbf{W}}^i$ is likely not unique and suggests some ad hoc fixes (Figure 2).

Figure 2:
allsynth warns users when the $\hat{\mathbf{W}}^i$ matrix is likely not unique

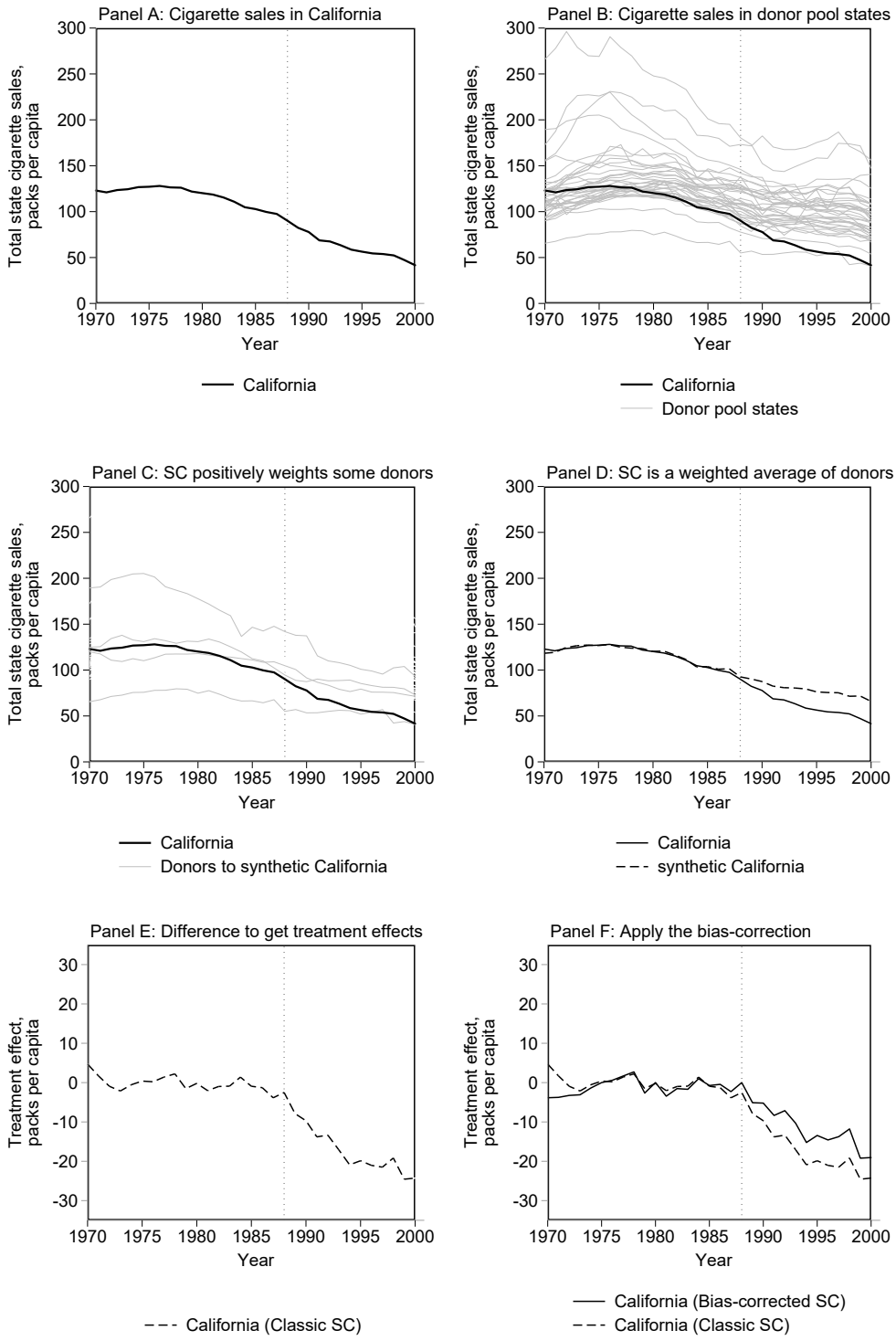
```
Warning: the -synth- weighting matrix W for treated unit (state == 3) contains
>  more non-zero weights than predictor variables and is likely not unique. Co
> nsider adjusting the number of predictor variables or appropriately restrict
> ing the donor pool
```

Example 3

Calculate, display, and save the classic and the bias-corrected "gaps" between the treated unit outcome and the synthetic control outcome, and plot the bias-corrected outcome paths of the treated unit and its synthetic control:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) bcorrect(merge figure
> ) keep(smokingresults) replace
```

Figure 1:
Step-by-step visual example of synthetic control estimation



This example reproduces the estimation in Example 1, but as `bcorrect()` and its own `figure` option are specified, it also calculates the classic and the (OLS regression estimated) bias-corrected gaps for each period (for the treated unit, 3, which is California), while plotting the bias-corrected outcome paths of the treated unit and its synthetic control. The classic results are saved in the working directory as *`smokingresults.dta`* as the `keep()` command is specified, and because `bcorrect(`*`merge`*`)` is also specified those results are merged and saved to the same file, and the variables *_Y_treated* and *_Y_synthetic* are replaced in this saved file by their bias-corrected values (meaningful only for calculating the bias-corrected gap). If `bcorrect(merge)` had instead been specified, *_Y_treated* and *_Y_synthetic* would have been left as their uncorrected values, and the bias-corrected values would have been saved as *_Y_treated_bc* and *_Y_synthetic_bc*. The `replace` option is specified so *`smokingresults.dta`* can be saved even if the file already exists (it will be overwritten). As `bcorrect()` was also specified with `figure`, **allsynth** plots *_Y_treated_bc* and *_Y_synthetic_bc* (Figure 3). **allsynth** also displays the key results after estimation (Figure 4).

Figure 3:
Specifying *`figure`* in `bcorrect()` plots the trajectories of the bias-corrected outcome values

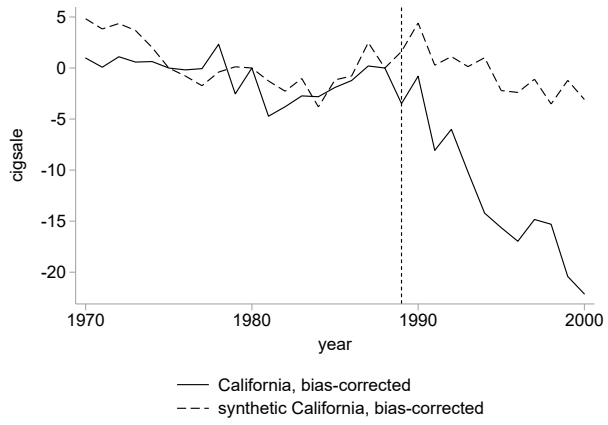


Figure 4:
allsynth displays the key results after estimation

	state	year	gap	gap_bc	unique_W
1.	3	1970	4.714999	-3.847018	1
2.	3	1971	1.701101	-3.750713	1
3.	3	1972	-.9330038	-3.256549	1
4.	3	1973	-2.134596	-3.091832	1
5.	3	1974	-.5483028	-1.377421	1
6.	3	1975	.3905984	0	1
7.	3	1976	.2115016	.5977706	1
8.	3	1977	1.404701	1.673252	1
9.	3	1978	2.2231	2.723592	1
10.	3	1979	-1.531299	-2.652859	1
11.	3	1980	-.1907034	0	1
12.	3	1981	-2.0393	-3.449938	1
13.	3	1982	-.9850991	-1.524845	1
14.	3	1983	-.8995969	-1.701223	1
15.	3	1984	1.391301	.995675	1
16.	3	1985	-.9249966	-.7384964	1
17.	3	1986	-1.280606	-.4435956	1
18.	3	1987	-3.860799	-2.29259	1
19.	3	1988	-2.5063	0	1
20.	3	1989	-7.887098	-5.108345	1
21.	3	1990	-9.693599	-5.212851	1
22.	3	1991	-13.8027	-8.347837	1
23.	3	1992	-13.344	-7.124086	1
24.	3	1993	-17.0624	-10.32301	1
25.	3	1994	-20.8943	-15.22401	1
26.	3	1995	-19.8568	-13.43382	1
27.	3	1996	-21.0405	-14.57536	1
28.	3	1997	-21.4914	-13.74036	1
29.	3	1998	-19.1642	-11.78082	1
30.	3	1999	-24.554	-19.20885	1
31.	3	2000	-24.2687	-19.05358	1

Example 4

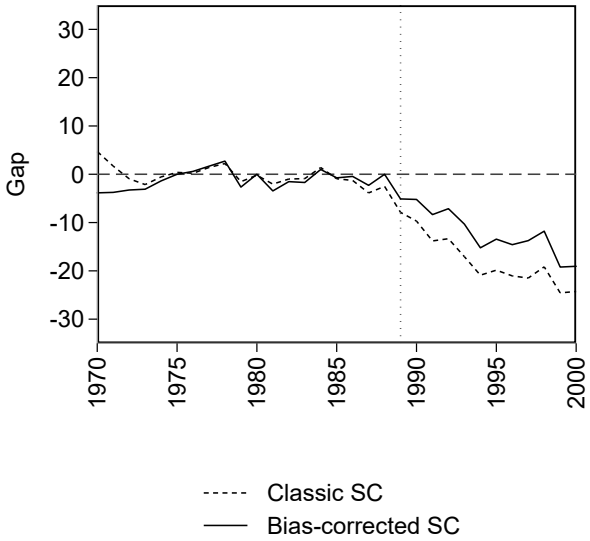
Calculate, display, and save the classic and the bias-corrected gaps between the treated unit outcome and the synthetic control outcome, and additionally plot the paths of the classic and bias-corrected gaps:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) bcorrect(merge) gapfi
> gure(classic bcorrect) keep(smokingresults) replace
```

This example reproduces the estimation in Example 3, but as `gapfigure()` and its own `classic` and `bcorrect` options are specified, it also plots the dynamic paths of the classic and the (OLS regression estimated) bias-corrected gaps against each other (Figure 2). Note how, in this case, the post-treatment bias-corrected gaps are smaller than those produced by classic synthetic control estimation. Also note that a variable with the bias-corrected gaps, *gaps_bc*, is now displayed after estimation and saved in the stored

results and the `keep()` file.

Figure 5:
Specifying `gapfigure(classic bcorrect)` plots the classic and bias-corrected gaps



Example 5

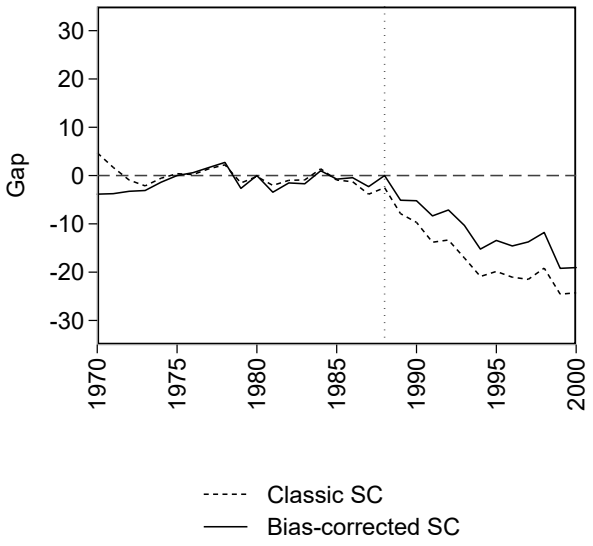
Calculate, display, and save the classic and the bias-corrected gaps between the treated unit outcome and the synthetic control outcome, and additionally plot the paths of the classic and bias-corrected gaps with the dotted vertical line now indicating the treatment period immediately preceding treatment:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) bcorrect(merge) gapfi
> gure(classic bcorrect lineback) keep(smokingresults) replace
```

This example reproduces the estimation in Example 4, but as the `gapfigure()` option `lineback` is also specified, the dotted vertical line which by default indicates the specified treatment period (here, 1989) has now been moved to the period immediately preceding the specified treatment period. This plots Figure 6 (which is also found in Panel F of Figure 1), analogous to Figure 3 in Abadie, Diamond, and Hainmueller (2010), but with the bias-corrected outcome path also added (note: this is not the exact specification used

in Abadie, Diamond, and Hainmueller (2010), which is why the classic outcome path differs slightly from Figure 3 in that paper).

Figure 6:
Specifying `gapfigure(lineback)` **shifts the vertical dotted line to the final pre-treatment period**



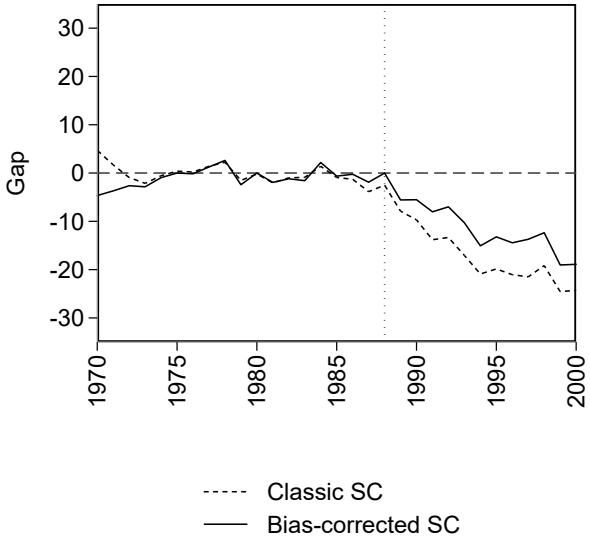
Example 6

Calculate, display, and save the classic and the bias-corrected gaps between the treated unit outcome and the synthetic control outcome. Plot the paths of the classic and bias-corrected gaps with the dotted vertical line now indicating the treatment period immediately preceding treatment. Estimate the bias using elastic net regression:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) bcorrect(merge elasti
> c) gapfigure(classic bcorrect lineback) keep(smokingresults) replace
```

This example reproduces the estimation in Example 5, but as the `bcorrect()` option `elastic` is also specified, the bias is estimated using elastic net regression instead of OLS. Figure 7 is produced, visualizing the difference.

Figure 7:
Specifying `bcorrect(elastic)` estimates the bias using elastic net regression



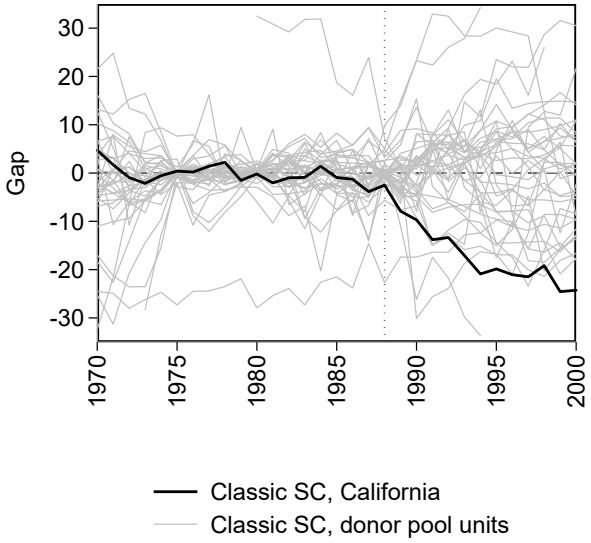
Example 7

Calculate, display, and save the classic *RMSPE*-ranked *p*-values from in-space placebo runs, and plot the dynamic paths of classic gaps for the treated unit and for each of the donor pool units (placebo treated units), with the dotted vertical line indicating the period immediately preceding treatment:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) gapfig(classic placeb
> os lineback) pval keep(smokingresults) rep
```

This example reproduces the estimation in Example 1, also calculating and displaying, storing, and saving the *RMSPE*, *RMPSE* rank, and the *p*-values for the classic estimates (and saving the placebo estimates), and additionally plots the dynamic paths of the classic gaps for the treated unit and each donor pool unit, with the dotted vertical line indicating the period immediately preceding treatment. This produces Figure 8, analogous to Figure 4 in Abadie, Diamond, and Hainmueller (2010). Note that pvalues and `keep()` must be specified as `gapfigure(classic placebos)` is specified.

Figure 8:
Specifying `pvalues` **and** `gapfigure(classic placebos)` **plots the estimated gap and placebo gaps**



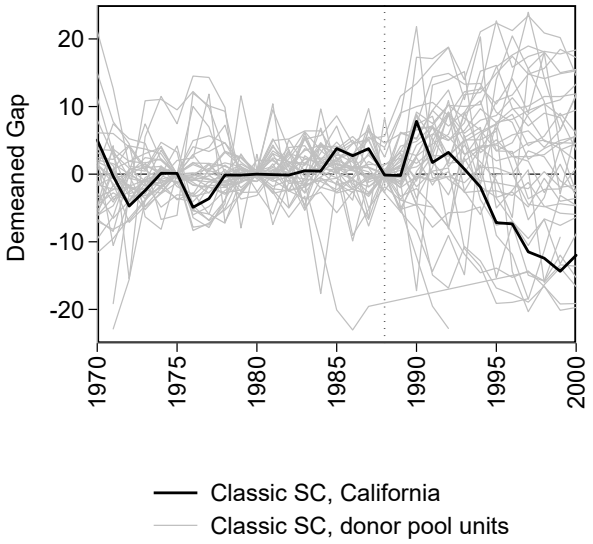
Example 8

Calculate, display, and save the classic *RMSPE*-ranked *p*-values from in-space placebo runs, and plot the dynamic paths of classic gaps for the treated unit and for each of the donor pool units (placebo treated units), with the dotted vertical line indicating the period immediately preceding treatment, and with the pre-treatment mean of *cigsale* and *retprice* for each unit subtracted from those variable values in each period:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) gapfig(classic placeb
> os lineback) pval trans(cigsale retprice, demean) keep(smokingresults) rep
```

This example reproduces the estimation in Example 7 but with *cigsale* and *retprice* demeaned (adjusted given the pre-treatment mean of each variable for each unit) as `transform(cigsale retprice, demean)` is specified. Figure 9 is also produced.

Figure 9:
Specifying `transform(cigsale retprice, demean)` subtracts (by unit) the pre-treatment means of those variables from their values



Example 9

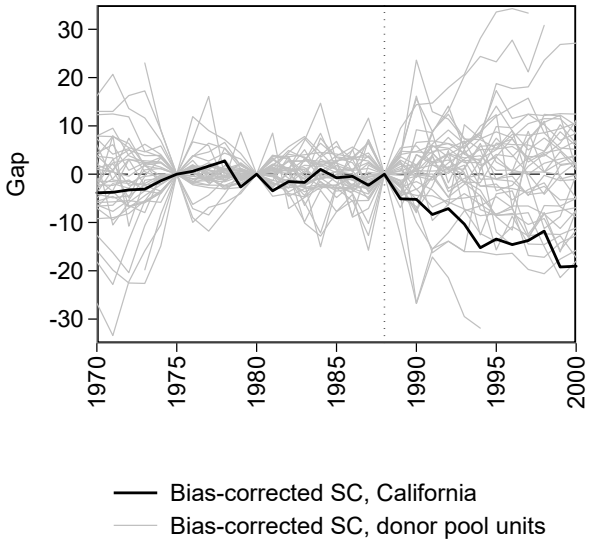
Calculate, display, and save and the bias-corrected gaps between the treated unit outcome and the synthetic control outcome, for the treated unit and also for each donor pool unit (placebo treatments), and calculate the *RMSPE*-ranked *p*-values. Plot the dynamic paths of bias-corrected gaps for the treated unit and for each of the donor pool units (placebo treated units), with the dotted vertical line now indicating the period immediately preceding treatment:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) bcor(merge) gapfig(bc
> orrect placebos lineback) pvalues keep(smokingresults) replace
```

This example reproduces the estimation in Example 3, but as `pvalues` is specified, it additionally estimates, saves, and stores the values for each unit in the donor pool (placebo runs) to calculate the *RMSPE*-ranked *p*-values. Note that while the classic gaps (the estimated marginal treatment effects from Abadie, Diamond, and Hainmueller (2010)) are all highly statistically significant in all post-treatment periods (for

each post-treatment year the *RMSPE* is larger than that of all the donor pool units), the bias-corrected gaps are not significant at the 10% level before 1994, and after that the p-values are only 0.077 (the *RMSPE*s are ranked third among 39 total runs rather than first). As `gapfigure()` and its own `bcorrect, placebos, and lineback` options are specified along with `bcorrect()`, it also plots the dynamic paths of the the (bias-corrected) gaps for the treated unit against those for each donor pool unit against each other in a figure, with the dotted vertical line indicating the period immediately preceding treatment. Note that all the option abbreviations are used. This plot (Figure 10) is the bias-corrected analogue of Figure 4 in Abadie, Diamond, and Hainmueller (2010).

Figure 10:
Specifying `gapfigure(bcorrect placebos)` **plots the bias-corrected gaps and placebo gaps**



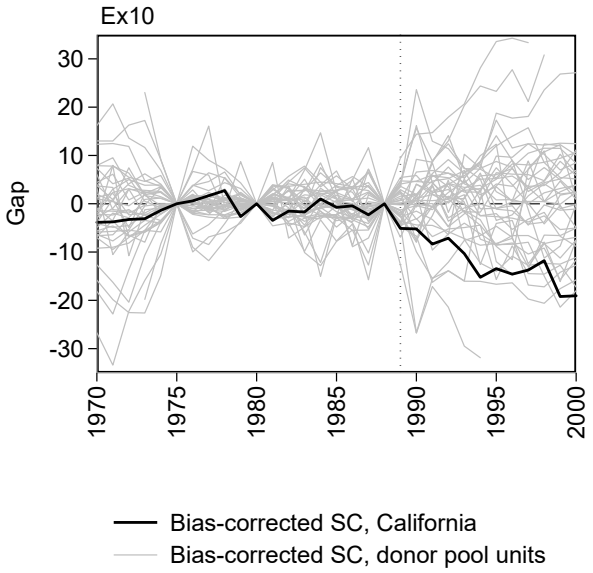
Example 10

Calculate, display, and save and the bias-corrected gaps between the treated unit outcome and the synthetic control outcome, for the treated unit and also for each donor pool unit (placebo treatments), and calculate the *RMSPE*-ranked *p*-values. Plot the bias-corrected gaps for the treated unit and for each of the donor pool units (placebo treated units) with the title "Ex 10", saving the graph as "ex10.pdf" with replacement:

```
. allsynth cigsale beer(1984(1)1988) lnincome retprice age15to24 cigsale(1988)
>  cigsale(1980) cigsale(1975), trunit(3) trperiod(1989) bcor(merge) gapfig(bc
> orrect placebos, title(Ex10) save(ex10, replace)) pval keep(smokingresults)
> rep
```

This example reproduces the estimation and plot from Example 9, but with the `gapfigure()` option `lineback` dropped. Specifying the `gapfigure()` options `title(Ex10) save(ex10, replace)` after the comma adds the title "Ex10" and saves the graph as "ex10.pdf", replacing any existing file with the same name.

Figure 11:
Specifying `transform(cigsale retprice, demean)` subtracts (by unit) the pre-treatment means of those variables from their values



4.2 allsynth *with* the `stacked()` option

Examples 11–13 make use of the `allsynth_walmart` data included with **allsynth**, and are a subset of the data from Wiltshire (2021) (for use in estimation of the average treatment effect of Walmart Supercenter entry on aggregate county employment. See Wiltshire (2021) for details).

Note that the `stacked()` commands are too long for Stata's help file languge (SMCL) to allow to be

interactive (clickable on a Windows system) on their own, so the help file (type `help allsynth`) defines several macros to allow the examples to run interactively on a Windows system. Here I do not define or call those macros, but instead treat the full commands as if they worked interactively in the help file on their own.

Begin by loading the *allsynth_walmart* data and declaring the dataset as a panel:

```
. sysuse allsynth_walmart

. tsset cty_fips year
       panel variable:  cty_fips (strongly balanced)
        time variable:  year, 1990 to 2005
                delta:  1 unit
```

Next define a new directory to store the output:

```
. capture mkdir "allsynth_walmart"
```

Example 11

Calculate the "stacked" average treatment effects of Supercenter entry on aggregate county employment in Indiana only. Note that `trunit()` and `trperiod()` need not be defined because the `stacked()` option already requires variables that define these.

First preserve the data so we can restrict it to only treated counties in Indiana and all donor pool counties, to speed up the run-time, then restrict the data to only the untreated counties and treated counties in Indiana:

```
. preserve

. keep if supercenter == 0 | floor(cty_fips/1000) == 18
(8,656 observations deleted)
```

Estimate, display, and save the classic and bias-corrected average treatment effects (gaps) of Walmart Supercenter entry on employment in treated counties in Indiana, in percentage terms of the employment in each county's final pre-treatment year:

```
. allsynth emps_n10 emps_n10(1990) emps_n10(1991) emps_n10(1992) emps_n10(1990
> (1)1994) log_emps_n10(1990(1)1994) log_pe_salary_n10(1990(1)1994) log_emps_n
> 44(1990(1)1994) log_pe_salary_n44(1990(1)1994) emps_shr_n44(1990(1)1994) pop
> _t(1990(1)1994), transform(emps_n10, normalize) bcorrect(merge) keep(allsynt
> h_walmart/employment, replace) stacked(trunits(supercenter) trperiods(super_
> year), clear figure(classic bcorrect))
```

As `stacked()` is specified, this example estimates and plots the classic and bias-corrected average treatment effects of Walmart Supercenter entry on county employment in Indiana. The *supercenter* variable identifies with a 1 all the counties in Indiana which got their first Walmart Supercenter over this period, and identifies with a 0 all of the donor pool counties in the sample where Walmart tried to build a Supercenter during this period but was blocked by local efforts. The *super_year* variable identifies (in every year) the year of Supercenter entry into the treated counties. Note that "`, clear`" is also specified within `stacked()`, as it is required (because all existing identically-named files are erased when `stacked()` is specified). The estimated average effects are in percentage terms as `transform(employment, normalize)` was specified, which normalizes employment in in each treated county and its donor pool counties to the final pre-treatment period for that treated county. The results are displayed, stored, and saved in event time (Figure 12), as Indiana's counties were treated over several years. The classic and bias-corrected estimated average treatment effects are plotted, as the `stacked()` option `figure(classic bcorrect)` is specified (Figure 13).

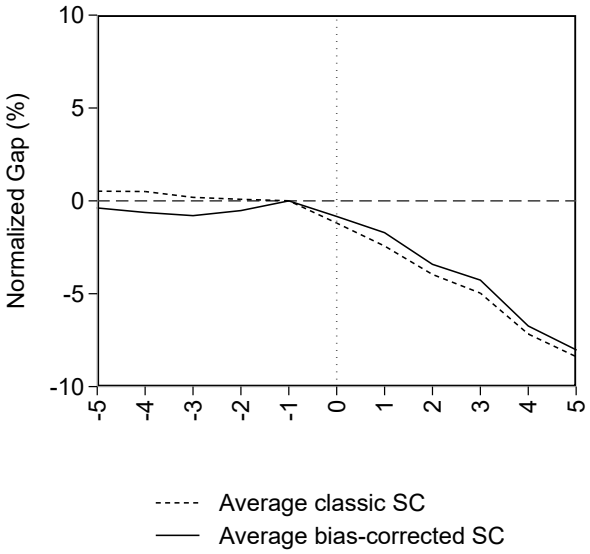
Figure 12:
Specifying `stacked()` displays the estimated average gaps (here, in event time)

```
Calculating the estimated average treatment effect for treated units
```

	_tm	gap	gap_bc
1.	-5	.5214813	-.3812578
2.	-4	.5025266	-.6251675
3.	-3	.1889037	-.7964208
4.	-2	.0832371	-.5234967
5.	-1	.0038462	0
6.	0	-1.196394	-.8396251
7.	1	-2.432352	-1.712792
8.	2	-3.962025	-3.417977
9.	3	-4.967881	-4.267233
10.	4	-7.170208	-6.737894
11.	5	-8.387778	-8.020466

```
Estimated average treatment effects saved in allsynth_walmart/employment_ate.d
> ta
```

Figure 13:
Specifying the `stacked()` option `figure(classic bcorrect)` plots the classic and bias-corrected estimated average gaps



Example 12

Calculate the stacked average treatment effects of Supercenter entry on aggregate county employment across the U.S. as in Wiltshire (2021), but without p-values. Note that the run-time for this example is 35 minutes using Stata MP on a Unix server.

First restore the data to include all treated and all donor pool counties:

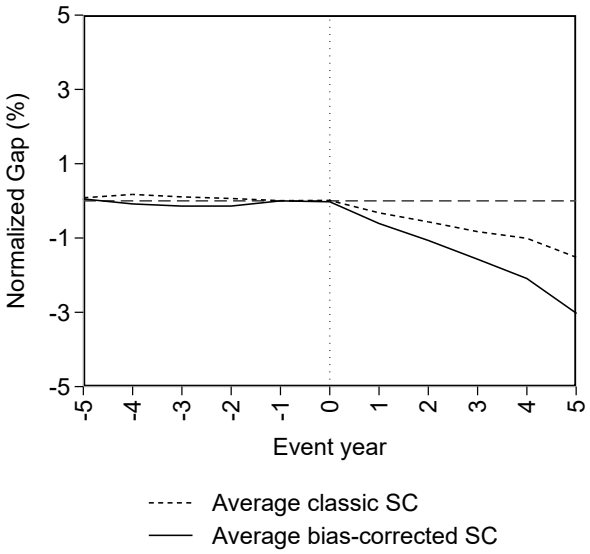
```
. restore
```

Do as in Example 11, but for all U.S. counties which received their first Supercenter over this period. Weight the estimated effects by 1990 county population, and restrict to those event years in which all treated counties are observed. Restrict the donor pool for each treated county to those donor pool counties in different commuting zones. Set the x-axis title to "Event year", and save the graph as "*ate.pdf*" in the "*allsynth_walmart*" directory:

```
. allsynth emps_n10 emps_n10(1990) emps_n10(1991) emps_n10(1992) emps_n10(1990
> (1)1994) log_emps_n10(1990(1)1994) log_pe_salary_n10(1990(1)1994) log_emps_n
> 44(1990(1)1994) log_pe_salary_n44(1990(1)1994) emps_shr_n44(1990(1)1994) pop
> _t(1990(1)1994), transform(emps_n10, normalize) bcorrect(merge) keep(allsynt
> h_walmart/employment, replace) stacked(trunits(supercenter) trperiods(super_
> year), clear avgweights(pop_1990) balanced donorcond(sum czone if supercente
> r == 1, gen cz = r(mean)) donorif(czone != cz) figure(classic bcorrect, save
> (allsynth_walmart/ate, replace) xtitle(Event year)))
```

This example does as Example 11, but for all treated counties in the U.S. As `avgweights(pop_1990)` is specified, the estimated average treatment effects will be calculated by weighting the estimated marginal treatment effects of each treated county by their 1990 populations. As `balanced` is specified, the displayed, saved, and plotted estimates will be restricted to those event years in which all treated units are observed (in this case, `balanced` does nothing as the sample is already balanced over event years $e = [-5,5]$). As `donorcond(sum czone if supercenter == 1, gen cz = r(mean)) donorif(czone != cz)` is specified, the donor pool for each treated unit will be restricted to only those donor pool counties in other commuting zones. As `, save(allsynth_walmart/ate, replace)` is specified, the generated plot of classic and bias-corrected estimated average treatment effects will be saved to *`allsynth_walmart/ate.pdf`* and will replace any existing file of the same name (Figure 14). As `xtitle(Event year)` is specified, the x-axis title will be changed to "Event year".

Figure 14:
Specifying the `stacked()` option `figure(classic bcorrect)` plots the classic and bias-corrected estimated average gaps



Example 13

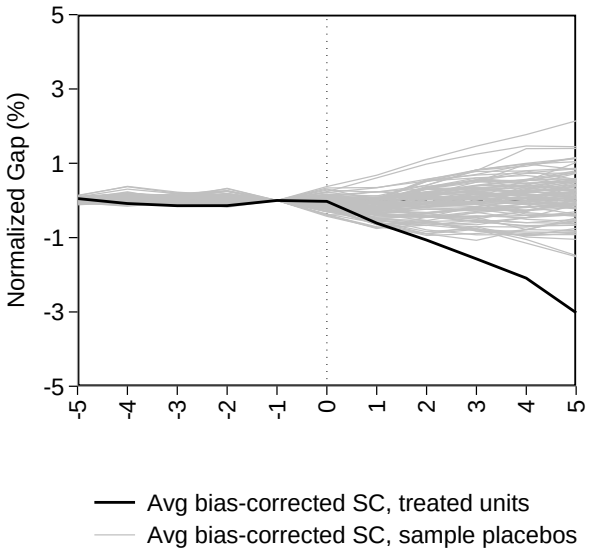
Do as Example 12, but estimate *RMSPE*-ranked *p*-values and generate the plot for the bias-corrected estimated ATE and 1000 sampled placebo average gaps, as in Wiltshire (2021). Note that this example takes over 24 hours to run using Stata MP on a Unix server.

```
. allsynth emps_n10 emps_n10(1990) emps_n10(1991) emps_n10(1992) emps_n10(1990
> (1)1994) log_emps_n10(1990(1)1994) log_pe_salary_n10(1990(1)1994) log_emps_n
> 44(1990(1)1994) log_pe_salary_n44(1990(1)1994) emps_shr_n44(1990(1)1994) pop
> _t(1990(1)1994), transform(emps_n10, normalize) bcorrect(merge) pvalues keep
> (allsynth_walmart/employment, replace) stacked(trunits(supercenter) trperiod
> s(super_year), clear avgweights(pop_1990) balanced donorcond(sum czone if su
> percenter == 1, gen cz = r(mean)) donorif(czone != cz) figure(bcorrect place
> bos, save(allsynth_walmart/ate, replace)))
```

This example does as Example 12, but also calculated the *RMSPE*-ranked *p*-values as `pvalues` is specified. As `sampleavgs(1000)` is specified, these will be based on 1000 randomly sampled placebo average gaps. The "Event year" title on the x-axis (from Example 12) is dropped, and as `figure(bcorrect`

`placebos)` is specified, the generated plot will show the bias-corrected estimated average treatment effect and the sampled placebo average gaps (Figure 15).

Figure 15:
Specifying the `stacked()` option `figure(bcorrect placebos)` plots the bias-corrected estimated average gaps and bias-corrected average placebo gaps



References

Abadie, Alberto. 2021. "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature* 59 (2):391—-425.

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

Abadie, Alberto and Guido W Imbens. 2011. "Bias-Corrected Matching Estimators for Average Treatment Effects." *Journal of Business & Economic Statistics* 29 (1):1–11.

Abadie, Alberto and Jérémy L'Hour. 2021. "A penalized synthetic control estimator for disaggregated data." *Journal of the American Statistical Association* 116 (536):1817–1834.

Acemoglu, Daron, Simon Johnson, Amir Kermani, James Kwak, and Todd Mitton. 2016. "The Value of Connections in Turbulent Times: Evidence from the United States." *Journal of Financial Economics* 121 (2):368–391.

Ben-Michael, Eli, Avi Feller, and Jesse Rothstein. 2021. "The augmented synthetic control method." *Journal of the American Statistical Association* 116 (536):1789–1803.

———. 2022. "Synthetic Controls with Staggered Adoption." *Journal of the Royal Statistical Society* 84 (2).

Cavallo, Eduardo, Sebastian Galiani, Ilan Noy, and Juan Pantano. 2013. "Catastrophic natural disasters and economic growth." *Review of Economics and Statistics* 95 (5):1549–1561.

Chernozhukov, Victor, Kaspar Wuthrich, and Yinchu Zhu. 2019. "An exact and robust conformal inference method for counterfactual and synthetic controls." *arXiv preprint arXiv:1712.09089* .

Doudchenko, Nikolay and Guido W Imbens. 2016. "Balancing, regression, difference-in-differences and synthetic control methods: A synthesis." Working paper, National Bureau of Economic Research.

Dube, A and B Zipperer. 2015. "Pooling multiple case studies using synthetic controls: An application to minimum wage policies." *Institute for the Study of Labor (IZA) Discussion Papers 8944* .

Ferman, Bruno and Cristine Pinto. 2017. "Placebo Tests for Synthetic Controls." Working paper.

———. 2021. "Synthetic controls with imperfect pretreatment fit." *Quantitative Economics* 12 (4):1197–1221.

Firpo, Sergio and Vitor Possebom. 2018. "Synthetic Control Method: Inference, Sensitivity Analysis and Confidence Sets." *Journal of Causal Inference* 6 (2).

Galiani, Sebastian and Brian Quistorff. 2017. "The synth_runner package: Utilities to automate synthetic control estimation using synth." *The Stata Journal* 17 (4):834–849.

Hahn, Jinyong and Ruoyao Shi. 2017. "Synthetic Control and Inference." *Econometrics* 5 (4):52.

Kreif, Noémi, Richard Grieve, Dominik Hangartner, Alex James Turner, Silviya Nikolova, and Matt Sutton. 2016. "Examination of the synthetic control method for evaluating health policies with multiple treated units." *Health economics* 25 (12):1514–1528.

Peri, Giovanni, Derek Rury, and Justin C Wiltshire. 2021. "The Economic Impact of Migrants from Hurricane Maria." Working paper.

Powell, David. 2021. "Synthetic Control Estimation Beyond Comparative Case Studies: Does the Minimum Wage Reduce Employment?" *Journal of Business & Economic Statistics* :1–13.

Wiltshire, Justin C. 2021. "Walmart Supercenters and Monopsony Power: How a Large, Low-Wage Employer Impacts Local Labor Markets." *Working paper* .

Zhang, Ziyan. 2019. “Inference for Synthetic Control Methods with Multiple Treated Units.” *arXiv preprint arXiv:1912.00568* .