Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

scpi: Uncertainty Quantification for Synthetic Control Methods

Matias D. Cattaneo
Princeton University

Yingjie Feng
Tsinghua University

Filippo Palomba
Princeton University

Rocío Titiunik
Princeton University

Abstract

The synthetic control method offers a way to quantify the effect of an intervention using weighted averages of untreated units to approximate the counterfactual outcome that the treated unit(s) would have experienced in the absence of the intervention. This method is useful for program evaluation and causal inference in observational studies. We introduce the software package **scpi** for prediction and inference using synthetic controls, implemented in Python, R, and Stata. For point estimation or prediction of treatment effects, the package offers an array of (possibly penalized) approaches leveraging the latest optimization methods. For uncertainty quantification, the package offers the prediction interval methods introduced by Cattaneo, Feng, and Titiunik (2021) and Cattaneo, Feng, Palomba, and Titiunik (2022). The paper includes numerical illustrations and a comparison with other synthetic control software.

Keywords: program evaluation, causal inference, synthetic controls, prediction intervals, nonasymptotic inference, R, Python, Stata.

1. Introduction

The synthetic control method was introduced by Abadie and Gardeazabal (2003), and since then it has become a popular approach for program evaluation and causal inference in observational studies. It offers a way to study the effect of an intervention (e.g., treatments at the level of aggregate units, such as cities, states, or countries) by constructing weighted averages of untreated units to approximate the counterfactual outcome that the treated unit(s) would have experienced in the absence of the intervention. While originally developed for the special

case of a single treated unit and a few control units over a short time span, the methodology has been extended in recent years to a variety of other settings with longitudinal data. See Abadie (2021) for a review on synthetic control methods, and Abadie and Cattaneo (2018) for a review on general methods for program evaluation.

Most methodological developments in the synthetic control literature have focused on either expanding the causal framework or developing new implementations for prediction/point estimation. Examples of the former include disaggregated data settings (Abadie and L'Hour 2021) and staggered treatment adoption (Ben-Michael, Feller, and Rothstein 2022), while examples of the latter include employing different constrained estimation methods (see Table 3 below for references). Conceptually, implementation of the synthetic control method involves two main steps: first, treated units are "matched" to control units using only their pre-intervention data via (often constrained) regression methods, and second, prediction of the counterfactual outcomes of the treated units are obtained by combining the pre-intervention "matching" weights with the post-intervention data of the control units. As a result, the synthetic control approach offers a prediction or point estimator of the (causal) treatment effect for the treated unit(s) after the intervention is deployed.

Compared to prediction or estimation, considerably less effort has been devoted to develop principled uncertainty quantification for synthetic control methods. The most popular approach in practice is to employ design-based permutation methods taking the potential outcome variables as non-random (Abadie, Diamond, and Hainmueller 2010). Other approaches include methods based on large-sample approximations for disaggregated data under correctly specified factor-type models (Li 2020), time-series permutation-based inference (Chernozhukov, Wüthrich, and Zhu 2021), large-sample approximations for high-dimensional penalization methods (Masini and Medeiros 2021), and cross-sectional permutation-based inference in semiparametric duration-type settings (Shaikh and Toulis 2021). A conceptually distinct approach to uncertainty quantification is proposed by Cattaneo, Feng, and Titiunik (2021) and Cattaneo, Feng, Palomba, and Titiunik (2022), who take the potential outcome variables as random and develop prediction intervals for the imputed (counterfactual) outcome of the treated unit(s) in the post-intervention period employing finite-sample probability concentration methods.

This article introduces the software package **scpi** for prediction and inference using synthetic control methods, implemented in Python (van Rossum *et al.* 2011; Cattaneo, Feng, Palomba, and Titiunik 2024a), R (R Core Team 2024; Cattaneo, Feng, Palomba, and Titiunik 2024b), and Stata (StataCorp 2019; Cattaneo, Feng, Palomba, and Titiunik 2024c). For prediction or point estimation of treatment effects, the package offers an array of possibly penalized approaches leveraging the latest conic optimization methods (Domahidi, Chu, and Boyd 2013; Fu, Narasimhan, and Boyd 2020); see also Boyd and Vandenberghe (2004) for an introduction. For uncertainty quantification, the package focuses on the aforementioned prediction interval methods under random potential outcomes. The rest of the article focuses on the R implementation of the software, but we briefly illustrate analogous functionalities for Python in Appendix A, and for Stata in Appendix B.

The R package **scpi** includes the following six functions:

- `scdata()` and `scdataMulti()`. These functions take as input a '`DataFrame`' object and process it to prepare the data matrices used for point estimation/prediction and inference/uncertainty quantification. The function `scdata` is specific to the single treated unit

case, whereas `scdataMulti` can be used with multiple treated units and/or when treatment is adopted in a staggered fashion. Both functions allow the user to specify multiple features of the treated unit(s) to be matched by the synthetic unit(s), as well as feature-specific covariate adjustment, and can handle both independent and identically distributed (i.i.d.) and non-stationary (cointegrated) data.

- `scest()`. This function handles '`scpi_data`' or '`scpi_data_multi`' objects produced with `scdata()` or `scdataMulti()`, respectively, and then implements a class of synthetic control predictions/point estimators for quantification of treatment effects. The implementation allows for multiple features, with and without additional covariate adjustment, and for both stationary and non-stationary data. The allowed prediction procedures include unconstrained weighted least squares as well as constrained weighted least squares with simplex, lasso-type, ridge-type parameter space restrictions and combinations thereof (see Table 2 below).

- `scpi()`. This function takes as input an '`scpi_data`' object produced with `scdata()` or an '`scpi_data_multi`' object produced with `scdataMulti()`, and then computes prediction intervals for a class of synthetic control predictions/point estimators for quantification of treatment effects. It relies on `scest()` for point estimation/prediction of treatment effects, and thus inherits the same functionalities of that function. In particular, `scpi()` is designed to be the main function in applications, offering both predictions/point estimators for treatment effects as well as inference/uncertainty quantification (i.e., prediction intervals) for synthetic control methods. The function also allows the user to separately model in-sample and out-of-sample uncertainty, offering a broad range of options for practice.

- `scplot()` and `scplotMulti()`. These functions process objects whose class is either '`scest`' or '`scpi`'. These objects contain the results of the point estimation/prediction or uncertainty quantification methods, respectively. The commands build on the **ggplot2** package (Wickham 2016) in R to compare the time series for the outcome of the treated unit(s) with the outcome time series of the synthetic control unit, along with the associated uncertainty. The functions return a '`ggplot`' object that can be further modified by the user.

The objects returned by `scest()` and `scpi()` support the methods `print()` and `summary()`. In typical applications, the user will first prepare the data using the function `scdata()` or `scdataMulti()`, and then produce predictions/point estimators for treatment effects with uncertainty quantification using the function `scpi()`. The function `scest()` is useful in cases where only predictions/point estimators are of interest. Numerical illustrations are given in Section 5.

There are many Python, R, and Stata packages available for prediction/point estimation and inference using synthetic control methods; Table 1 compares them to the package **scpi**.[1] As

[1]The table includes the following additional packages: **allsynth** (Wiltshire 2024); **ArCo** (Fonseca, Masini, Medeiros, and Vasconcelos 2017); **augsynth** (Ben-Michael 2024); **gsynth** (Xu and Liu 2021); **microsynth** (Robbins and Davenport 2023, 2021); **MSCMT** (Becker and Klössner 2024); **npsynth** (Cerulli 2020); **pensynth** (van Kesteren and Slaughter 2024); **pgsc** (Barrett 2018); **scinference** (Wuthrich 2021); **SCtools** (Silva and DeWitt 2024); **scul** (Greathouse 2022); **SCUL** (Hollingsworth 2024); **Synth** (Abadie, Diamond, and Hainmueller 2011; Hainmueller and Diamond 2023); **SyntheticControlMethods** (Engelbrektson 2021); **synth2** (Chen 2023); **synthdid** (Arkhangelsky, Athey, Hirshberg, Imbens, and Wager 2024); **tidysynth** (Dunford 2023); **treebased-sc** (Muehlbach 2021).

shown in the table, **scpi** is the first package to offer uncertainty quantification using prediction intervals with random potential outcomes for a wide range of different synthetic control predictors. The package is also one of the first to handle multiple treated units and staggered treatment adoption, offering a wider array of options in terms of predictors and inference methods when compared with the other packages currently available. Furthermore, the package includes misspecification-robust methods, employs the latest available optimization packages for conic programs, and offers automatic parallelization in execution whenever multi-core processors are present, leading to significant improvements in numerical stability and computational speed. Finally, **scpi** is the only package available in Python, R, and Stata, which gives full portability across multiple statistical software and programming languages, and also the only package employing directly native conic optimization via the `ECOS` solver (see Table 4 for details).

Package name	Statistical platform	Prediction method	Inference method	Multiple treated	Staggered adoption	Misspecification robust	Automatic parallelization	Last update
ArCo	R	LA	Asym			✓		2017-11-05
pgsc	R	SC	Perm	✓				2018-10-28
npsynth	St	SC	Perm					2020-06-23
scinference	R	SC, LA	Perm			✓		2021-05-13
gsynth	R	FA	Asym	✓	✓		✓	2021-08-06
Synth	Py	SC	Perm					2021-10-07
treebased-sc	Py	TB	Perm			✓		2021-11-01
scul	St	LA	Perm	✓				2022-08-21
tidysynth	R	SC	Perm					2023-05-21
Synth	R, St	SC	Perm					2023-06-02
microsynth	R	CA	Perm	✓			✓	2023-06-30
augsynth	R	SC, RI	Perm	✓	✓			2024-09-09
synth2	St	SC	Perm					2023-10-05
SCUL	R	LA	Perm					2023-10-10
sytnhdid	R	LS, RI	Asym	✓	✓			2024-01-15
MSCMT	R	SC	Perm				✓	2024-03-19
pensynth	R	LA	Perm					2024-03-28
SCtools	R	SC	Perm	✓			✓	2024-05-01
allsynth	St	SC	Perm	✓	✓			2024-07-11
scpi	Py, R, St	SC, LA, RI, LS, +	PI, Asym, Perm	✓	✓	✓	✓	2024-11-11

Table 1: Comparison of different packages available on `PyPi`, `CRAN`, `REPEC`, or `GitHub`. `Py` = Python (`https://www.python.org/`); `R` = R (`https://cran.r-project.org/`); `St` = Stata (`https://www.stata.com/`); `LA` = Lasso penalty; `CA` = calibration; `FA` = factor-augmented models; `LS` = unconstrained least squares; `RI` = Ridge penalty; `SC` = canonical synthetic control; `TB` = tree-based methods; `+` = user-specified options (see Table 3 below for more details); `Perm` = permutation-based inference; `Asym` = asymptotic-based inference; `PI` = prediction intervals (non-asymptotic probability guarantees). The symbol ✓ means that the feature is available. The last column reports the date of last update as of November 19, 2024.

The rest of the article is organized as follows. Section 2 introduces the canonical synthetic control setup and briefly discusses extensions to multiple treated units with possibly staggered treatment adoption. Section 3 gives a brief introduction to the theory and methodology underlying point estimation/prediction for synthetic control methods, discussing implementation details. Section 4 gives a brief introduction to the theory and methodology underlying uncertainty quantification via prediction intervals for synthetic control meth-

ods, and also discusses the corresponding issues of implementation. Section 5 showcases some of the functionalities of the package using a real-world dataset, and Section 6 concludes. The appendices illustrate the Python (Appendix A) and Stata (Appendix B) implementations of **scpi**. Detailed instructions for installation, script files to replicate the analyses, links to software repositories, and other companion information can be found in the package's website, https://nppackages.github.io/scpi/. The package **scpi** (Cattaneo *et al.* 2024b) is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=scpi.

2. Setup

We first consider the canonical synthetic control framework with a single treated unit. The researcher observes $J+1$ units for T_0+T_1 periods of time. Units are indexed by $i = 1, 2, \dots J, J+1$, and time periods are indexed by $t = 1, 2, \dots, T_0, T_0+1, \dots, T_0+T_1$. During the first T_0 periods, all units are untreated. Starting at T_0+1, unit 1 receives treatment but the other units remain untreated. Once the treatment is assigned at T_0+1, there is no change in treatment status: the treated unit continues to be treated and the untreated units remain untreated until the end of the series, T_1 periods later. The single treated unit in our context could be understood as an "aggregate" of multiple treated units; see Section 2.1 below for more discussion.

Each unit i at period t has two potential outcomes, $Y_{it}(1)$ and $Y_{it}(0)$, respectively denoting the outcome under treatment and the outcome in the absence of treatment. Two implicit assumptions are imposed: no spillovers (the potential outcomes of unit i depend only on i's treatment status) and no anticipation (the potential outcomes at t depend only on the treatment status of the same period). Then, the observed outcome Y_{it} is

$$Y_{it} = \begin{cases} Y_{it}(0), & \text{if} \quad i \in \{2, \dots, J+1\} \\ Y_{it}(0), & \text{if} \quad i = 1 \text{ and } t \in \{1, \dots, T_0\} \\ Y_{it}(1), & \text{if} \quad i = 1 \text{ and } t \in \{T_0+1, \dots, T_0+T_1\} \end{cases}.$$

The causal quantity of interest is the difference between the outcome path taken by the treated unit, and the path it would have taken in the absence of the treatment:

$$\tau_t := Y_{1t}(1) - Y_{1t}(0), \qquad t > T_0.$$

We view the two potential outcomes $Y_{1t}(1)$ and $Y_{1t}(0)$ as random variables, which implies that τ_t is a random quantity as well, corresponding to the treatment effect on a *single* treated unit. This contrasts with other analysis that regards the treatment effect as a fixed parameter (for references, see Abadie 2021).

The potential outcome $Y_{1t}(1)$ of the treated unit is observed after the treatment. To recover the treatment effect τ_t, it is necessary to have a "good" prediction of the counterfactual outcome of the treated unit, $Y_{1t}(0)$, after the intervention. The idea of the synthetic control method is to find a vector of weights $\mathbf{w} = (w_2, w_3, \dots, w_{J+1})^\top$ such that a given loss function is minimized under constraints, only using pre-intervention observations. Given the resulting set of constructed weights $\widehat{\mathbf{w}}$, the treated unit's counterfactual (potential) outcome is calculated as $\widehat{Y}_{1t}(0) = \sum_{i=2}^{J+1} \widehat{w}_i Y_{it}(0)$ for $t > T_0$. The weighted average $\widehat{Y}_{1t}(0)$ is often referred to as the

synthetic control of the treated unit, as it represents how the untreated units can be combined to provide the best counterfactual for the treated unit in the post-treatment period. In what follows, we briefly describe different approaches for point estimation/prediction leading to $\widehat{Y}_{1t}(0)$, and then summarize the uncertainty quantification methods to complement those predictions.

2.1. Extensions

Building on the canonical synthetic control setup, we can consider other settings involving multiple treated units with possibly staggered treatment adoption. In particular, we briefly discuss three potential extensions of practical interest.

- **Multiple post-treatment periods**. When outcomes are observed in multiple periods after the treatment, a researcher might be interested in the average treatment effect on the (single) treated unit across multiple post-treatment periods rather than the effect at a single period:

$$\tau := \frac{1}{T_1} \sum_{t=T_0+1}^{T_0+T_1} \Big(Y_{1t}(1) - Y_{1t}(0) \Big) = \frac{1}{T_1} \sum_{t=T_0+1}^{T_0+T_1} \tau_t.$$

  The analysis of this quantity can be accommodated by the framework above. For instance, given the predicted counterfactual outcome $\widehat{Y}_{1t}(0) = \sum_{i=2}^{J+1} \widehat{w}_i Y_{it}(0)$ for each post-treatment period $t > T_0$, the predicted average counterfactual outcome of the treated is given by

$$\sum_{i=2}^{J+1} \widehat{w}_i \Big(\frac{1}{T_1} \sum_{t=T_0+1}^{T_0+T_1} Y_{it}(0) \Big).$$

  This construction is equivalent to regarding the T_1 post-treatment periods as a "single" period and defining the post-treatment predictors as averages of the corresponding predictors across post-treatment time periods.

- **Multiple treated units**. The canonical single treated unit framework above can also be extended to the more general case of multiple treated units. For instance, suppose a researcher observes $N_0 + N_1$ units for $T_0 + T_1$ time periods, and let units be indexed by $i = 1, \ldots, N_1, N_1 + 1, \ldots, N_1 + N_0$. Without loss of generality, the first 1 to N_1 units are assumed to be treated and units from $N_1 + 1$ to $N_1 + N_0$ to be untreated. Treated and untreated potential outcomes are, respectively, denoted by $Y_{it}(1)$ and $Y_{it}(0)$ for $i = 1, \ldots, N_0 + N_1$. The observed outcome of the ith treated unit is given by $Y_{it} := \mathbb{1}(t \leq T_0) Y_{it}(0) + \mathbb{1}(t > T_0) Y_{it}(1)$.

  In such setting, a researcher might be interested in the *individual* treatment effect τ_{it}

$$\tau_{it} := Y_{it}(1) - Y_{it}(0), \quad t > T_0, \quad i = 1, \ldots, N_1,$$

  or in the *average* treatment effect on the treated $\tau_{\cdot t}$ across treated units

$$\tau_{\cdot t} := \frac{1}{N_1} \sum_{j=1}^{N_1} \Big(Y_{jt}(1) - Y_{jt}(0) \Big), \qquad t > T_0.$$

  The first causal quantity, τ_{it}, can be predicted in the single treated unit framework described above by considering one treated unit *at a time*. As an alternative, the construction

of synthetic control weights and thus the prediction of treatment effects τ_{it}'s could be done using all N_1 treated units *jointly*. Our software implementation allows for both possibilities. From a conceptual point of view, which method is more appropriate depends on the question under study. In some cases, interest may be on the predictand for an individual unit, while in other cases the focus may be on the effects for all treated units jointly. From a computational point of view, joint prediction will be more costly than individual prediction. See Section 3.1 of Cattaneo, Feng, Palomba, and Titiunik (2022) for a formal synthetic control framework with multiple treated units and detailed discussion about the two prediction strategies in this context.

To predict the second causal quantity, $\tau_{\cdot t}$, one extra step is necessary. Define an aggregate unit "ave" whose observed outcome is $Y_t^{\mathsf{ave}} := \frac{1}{N_1}\sum_{j=1}^{N_1} Y_{jt}$, for $t = 1, \ldots, T_0 + T_1$. Other features of "unit ave" used in the synthetic control construction can be defined similarly as averages of the corresponding features across multiple treated units. The single treated unit framework described above can now be applied to the average unit with outcome Y_t^{ave}.

- **Staggered treatment adoption**. Our framework can also be extended to the scenario where multiple treated units are assigned to treatment at different points in time, a *staggered adoption* design. In this case, one can understand the adoption time as a multivalued treatment assignment, and a large class of causal quantities can be defined accordingly. For example, let $T_i \in \{T_0+1, T_0+2, \ldots, T, \infty\}$ denote the adoption time of unit i where $T_i = \infty$ means unit i is never treated, and $Y_{it}(s)$ represents the potential outcome of unit i at time t that would be observed if unit i had adopted the treatment at time s. Suppose that the treatment effect on unit i one period after the treatment, i.e., $Y_{i(T_i+1)}(T_i) - Y_{i(T_i+1)}(\infty)$, is of interest. One can take all units that are treated later than $T_i + 1$ to obtain the synthetic control weights and construct the synthetic control prediction of the counterfactual outcome $Y_{i(T_i+1)}(\infty)$ accordingly. The methodology described below can be immediately applied to this problem.

The package **scpi** allows for estimation/prediction of treatment effects and uncertainty quantification via prediction intervals for the more general synthetic control settings discussed above. However, in order to streamline the exposition, the rest of this article focuses on the case of a single treated unit. See Cattaneo, Feng, Palomba, and Titiunik (2022) for a formal treatment of more general staggered adoption problems, and its supplemental appendix for further details on how the package **scpi** can be used in settings with multiple treatment units and staggered treatment adoption. Our companion replication files illustrate both the canonical single treated unit framework and the generalizations discussed above.

3. Synthetic control prediction

We consider synthetic control weights constructed simultaneously for M features of the treated unit, denoted by $\mathbf{A}_l = (a_{1,l}, \cdots, a_{T_0,l})^\top \in \mathbb{R}^{T_0}$, with index $l = 1, \ldots, M$. For each feature l, there exist $J + K$ variables that can be used to predict or "match" the T_0-dimensional vector $\mathbf{A}_l$. These $J + K$ variables are separated into two groups denoted by $\mathbf{B}_l = (\mathbf{B}_{1,l}, \mathbf{B}_{2,l}, \cdots, \mathbf{B}_{J,l}) \in \mathbb{R}^{T_0 \times J}$ and $\mathbf{C}_l = (\mathbf{C}_{1,l}, \cdots, \mathbf{C}_{K,l}) \in \mathbb{R}^{T_0 \times K}$, respectively. More precisely, for each j, $\mathbf{B}_{j,l} = (b_{j1,l}, \cdots, b_{jT_0,l})^\top$ corresponds to the lth feature of the jth unit observed in T_0 pre-treatment periods and, for each k, $\mathbf{C}_{k,l} = (c_{k1,l}, \cdots, c_{kT_0,l})^\top$ is another

vector of control variables also possibly used to predict $\mathbf{A}_l$ over the same pre-intervention time span. For ease of notation, we let $d = J + KM$.

The goal of the synthetic control method is to search for a vector of common weights $\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^J$ across the M features and a vector of coefficients $\mathbf{r} \in \mathcal{R} \subseteq \mathbb{R}^{KM}$, such that the linear combination of $\mathbf{B}_l$ and $\mathbf{C}_l$ "matches" $\mathbf{A}_l$ as close as possible, during the pre-intervention period, for all $1 \leq l \leq M$ and some convex feasibility sets $\mathcal{W}$ and $\mathcal{R}$ that capture the restrictions imposed. Specifically, we consider the following optimization problem:

$$\widehat{\boldsymbol{\beta}} := (\widehat{\mathbf{w}}^\top,\, \widehat{\mathbf{r}}^\top)^\top \in \underset{\mathbf{w}\in\mathcal{W},\, \mathbf{r}\in\mathcal{R}}{\arg\min}\ (\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r})^\top \mathbf{V} (\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r}) \tag{1}$$

where

$$\underbrace{\mathbf{A}}_{T_0\cdot M\times 1} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_M \end{bmatrix}, \quad \underbrace{\mathbf{B}}_{T_0\cdot M\times J} = \begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_M \end{bmatrix}, \quad \underbrace{\mathbf{C}}_{T_0\cdot M\times K\cdot M} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_M \end{bmatrix}$$

and $\mathbf{V}$ is a $T_0 \cdot M \times T_0 \cdot M$ weighting matrix reflecting the relative importance of different equations and time periods.

From (1), we can define the pseudo-true residual $\mathbf{u}$ as

$$\mathbf{u} = \mathbf{A} - \mathbf{B}\mathbf{w}_0 - \mathbf{C}\mathbf{r}_0, \tag{2}$$

where $\mathbf{w}_0$ and $\mathbf{r}_0$ denote the mean squared error population analog of $\widehat{\mathbf{w}}$ and $\widehat{\mathbf{r}}$. As discussed in the next section, the proposed prediction intervals are valid conditional on some information set $\mathscr{H}$. Thus, $\mathbf{w}_0$ and $\mathbf{r}_0$ above are viewed as the (possibly constrained) best linear prediction coefficients conditional on $\mathscr{H}$. We *do not* attach any structural meaning to $\mathbf{w}_0$ and $\mathbf{r}_0$: they are only (conditional) pseudo-true values whose meaning should be understood in context, and are determined by the assumptions imposed on the data generating process. In other words, we allow for misspecification when constructing the synthetic control weights $\widehat{\mathbf{w}}$, as this is the most likely scenario in practice.

Given the constructed weights $\widehat{\mathbf{w}}$ and coefficients $\widehat{\mathbf{r}}$, the counterfactual outcome at the post-treatment period T for the treated unit, $Y_{1T}(0)$, is predicted by

$$\widehat{Y}_{1T}(0) = \mathbf{x}_T^\top \widehat{\mathbf{w}} + \mathbf{g}_T^\top \widehat{\mathbf{r}} = \mathbf{p}_T^\top \widehat{\boldsymbol{\beta}}, \qquad \mathbf{p}_T := (\mathbf{x}_T^\top,\, \mathbf{g}_T^\top)^\top, \qquad T > T_0, \tag{3}$$

where $\mathbf{x}_T \in \mathbb{R}^J$ is a vector of predictors for control units observed in time T and $\mathbf{g}_T \in \mathbb{R}^{KM}$ is another set of user-specified predictors observed at time T. Variables included in $\mathbf{x}_T$ and $\mathbf{g}_T$ need not be the same as those in $\mathbf{B}$ and $\mathbf{C}$, but in practice it is often the case that $\mathbf{x}_T = (Y_{2T}(0), \cdots, Y_{(J+1)T}(0))^\top$ and $\mathbf{g}_T$ is excluded when $\mathbf{C}$ is not specified.

The next Section discusses implementation details leading to $\widehat{Y}_{1T}(0)$, including the choice of feasibility sets $\mathcal{W}$ and $\mathcal{R}$, weighting matrix $\mathbf{V}$, and additional covariates $\mathbf{C}$.

3.1. Implementation

The function `scdata()` in **scpi** prepares the data for point estimation/prediction purposes. This function takes as input an object of class '`DataFrame`' and outputs an object of class

'`scpi_data`' containing the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ described above, and a matrix of post-treatment predictors $\mathbf{P} = (\mathbf{p}_{T_0+1}, \cdots, \mathbf{p}_{T_0+T_1})^\top$. The user must provide a variable containing a unit identifier (`id.var`), a time variable (`time.var`), an outcome variable (`outcome.var`), the features to be matched (`features`), the treated unit (`unit.tr`), the control units (`unit.co`), the pre-treatment periods (`period.pre`), and the post-treatment periods (`period.post`). These options completely specify $\mathbf{A}, \mathbf{B}$, and $\mathbf{P}$. The user can also control the form of $\mathcal{R}$ in (1) or, equivalently, the form of $\mathbf{C}$, through the options `cov.adj` and `constant`. The former option allows the user to flexibly specify covariate adjustment feature by feature, while the latter option introduces a column vector of ones of size $T_0 \cdot M$ in $\mathbf{C}$. If $M = 1$, this is a simple constant term, but if $M \geq 2$ it corresponds to an intercept which is common across features.

The use of the options `cov.adj` and `constant` is best explained through some examples. If the user specifies only one feature ($M = 1$), then `cov.adj` can be an unnamed list:

```
R> cov.adj <- list(c("constant", "trend"))
```

This particular choice includes in $\mathbf{C}$ a constant term and a linear time trend. If instead multiple features ($M \geq 2$) are used to find the synthetic control weights $\widehat{\mathbf{w}}$, then `cov.adj` allows for feature-specific covariate adjustment. For example, in a two-feature setting ($M = 2$), the code

```
R> cov.adj <- list("f1" = c("constant", "trend"), "f2" = c("trend"))
```

specifies $\mathbf{C}$ as a block diagonal matrix where the first block $\mathbf{C}_1$ contains a constant term and a trend, while the second block $\mathbf{C}_2$ contains only a trend. If the user wants all features to share the same covariate adjustment, then it is sufficient to input a list with a unique element:

```
R> cov.adj <- list(c("constant", "trend"))
```

This specification creates a block diagonal matrix $\mathbf{C}$ with identical blocks. In the same example with $M = 2$, if `constant <- TRUE` and `cov.adj <- NULL`, then $\mathbf{C}$ would not be block diagonal, but rather a column vector of ones of size $2T_0$.

Finally, if $\mathbf{A}$ and $\mathbf{B}$ form a cointegrated system, by setting the option `cointegrated.data` to `TRUE` in `scdata()`, the matrix $\mathbf{P}$ is prepared in such a way that the function `scpi()` will properly handle in-sample and out-of-sample uncertainty quantification (see Section 4.3).

Once all the design matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\mathbf{P}$ have been created, we can proceed with point estimation/prediction of the counterfactual outcome of interest via the function `scest()`.

The form of the feasibility set $\mathcal{W}$ in (1) or, equivalently, the constraints imposed on the weights $\mathbf{w}$, can be set using the option `w.constr`. The package allows for the following family of constraints:

$$\mathcal{W} \in \Big\{\mathbb{R}^J,\ \{\mathbf{w} \in \mathbb{W} : ||\mathbf{w}||_p \leq Q\},\ \{\mathbf{w} \in \mathbb{R}^J : ||\mathbf{w}||_1 = Q, ||\mathbf{w}||_2 \leq Q_2\}\Big\},$$
$$\mathbb{W} \in \{\mathbb{R}^J, \mathbb{R}^J_+\}, \qquad p \in \{1, 2\}, \qquad Q \in \mathbb{R}_{++}, \qquad Q_2 \in \mathbb{R}_{++},$$

where the inequality constraint on the norm can be made an equality constraint as well. The user can specify the desired form for $\mathcal{W}$ through a list to be passed to the option `w.constr`:

Name	`w.constr`	Form of $\mathcal{W}$
OLS	`list(name = "ols")`	$\mathbb{R}^J$
Simplex	`list(name = "simplex", Q = Q)`	$\{\mathbf{w} \in \mathbb{R}^J_+ : \|\|\mathbf{w}\|\|_1 = Q\}$
Lasso	`list(name = "lasso", Q = Q)`	$\{\mathbf{w} \in \mathbb{R}^J : \|\|\mathbf{w}\|\|_1 \leq Q\}$
Ridge	`list(name = "ridge", Q = Q)`	$\{\mathbf{w} \in \mathbb{R}^J : \|\|\mathbf{w}\|\|_2 \leq Q\}$
L1-L2	`list(name = "L1-L2", Q = Q, Q2 = Q2)`	$\{\mathbf{w} \in \mathbb{R}^J_+ : \|\|\mathbf{w}\|\|_1 = Q, \|\|\mathbf{w}\|\|_2 \leq Q_2\}$

Table 2: Constraints on the weights directly implemented in **scpi**.

```
R> W1 <- list(p = "no norm", lb = -Inf)
R> W2 <- list(p = "L1", dir = "==", Q = 1, lb = 0)
R> W3 <- list(p = "L2", dir = "<=", Q = 1, lb = -Inf)
R> W4 <- list(p = "L1-L2", lb = 0, Q = 1, Q2 = 1, dir = "==/<=")
```

The four lines above create $\mathcal{W}_1 = \mathbb{R}^J$, $\mathcal{W}_2 = \{\mathbf{w} \in \mathbb{R}^J_+ : ||\mathbf{w}||_1 = 1\}$, $\mathcal{W}_3 = \{\mathbf{w} \in \mathbb{R}^J : ||\mathbf{w}||_2 \leq 1\}$, and $\mathcal{W}_4 = \{\mathbf{w} \in \mathbb{R}^J_+ : ||\mathbf{w}||_1 = 1, ||\mathbf{w}||_2 \leq 1\}$, respectively. In greater detail,

- `p` chooses the constrained norm of $\mathbf{w}$ among the options `'no norm'`, `'L1'`, `'L2'`, or `'L1-L2'`.
- `dir` sets the direction of the constraint $||\mathbf{w}||_p$ and it can be either `'=='`, `'<='`, or `'==/<='`.
- `Q` is the size of the constraint and it can be set to any positive real number.
- `lb` sets a (common) lower bound on $\mathbf{w}$ and it takes as input either `0` or `-Inf`.

Popular constraints can be called explicitly using the option `name` in the list passed to `w.constr`. Table 2 gives prototypical examples of such constraints. In particular, specifying `list(name = "simplex", Q = 1)` gives the standard constraint used in the canonical synthetic control method, that is, computing weights in (1) such that they are non-negative and sum up to one, and without including an intercept. This is the default in the function `scest()` (and `scpi()`). The following snippet showcases how each of these five constraints can be called automatically through the option `name` and manually through the options `p`, `Q`, `Q2`, `lb`, and `dir`. For simplicity, `Q` and `Q2` are set to 1 for ridge and L1-L2 constraints, but to replicate the results obtained with the option `name` one should input the proper `Q` according to the rules of thumb described further below.

```
R> w.constr <- list(name = "simplex")
R> w.constr <- list(p = "L1", lb = 0, Q = 1, dir = "==")

R> w.constr <- list(name = "ols")
R> w.constr <- list(p = "no norm", lb = -Inf, Q = NULL, dir = NULL)

R> w.constr <- list(name = "lasso")
R> w.constr <- list(p = "L1", lb = -Inf, Q = 1, dir = "<=")

R> w.constr <- list(name = "ridge")
```

Article	$\mathcal{W}$	$\mathcal{R}$	w.constr			constant
			name	Q	Q2	
Hsiao *et al.* (2012)	$\mathbb{R}^J$	$\mathbb{R}$	"ols"	NULL	NULL	TRUE
Abadie *et al.* (2010)	$\{\mathbf{w} \in \mathbb{R}_+^J : \|\|\mathbf{w}\|\|_1 = 1\}$	$\{0\}$	"simplex"	1	NULL	FALSE
Ferman and Pinto (2021)	$\{\mathbf{w} \in \mathbb{R}_+^J : \|\|\mathbf{w}\|\|_1 = 1\}$	$\mathbb{R}$	"simplex"	1	NULL	TRUE
Chernozhukov *et al.* (2021)	$\{\mathbf{w} \in \mathbb{R}^J : \|\|\mathbf{w}\|\|_1 \leq 1\}$	$\mathbb{R}$	"lasso"	1	NULL	TRUE
Amjad *et al.* (2018)	$\{\mathbf{w} \in \mathbb{R}^J : \|\|\mathbf{w}\|\|_2 \leq Q\}$	$\{0\}$	"ridge"	Q	NULL	FALSE
Arkhangelsky *et al.* (2021)	$\{\mathbf{w} \in \mathbb{R}_+^J : \|\|\mathbf{w}\|\|_1 = 1, \|\|\mathbf{w}\|\|_2 \leq Q_2\}$	$\mathbb{R}$	"L1-L2"	1	Q	TRUE

Table 3: Examples of $\mathcal{W}$ and $\mathcal{R}$ in the synthetic control literature ($M = 1$).

```
R> w.constr <- list(p = "L2", lb = -Inf, Q = 1, dir = "<=")

R> w.constr <- list(name = "L1-L2")
R> w.constr <- list(p = "L1-L2", lb = 0, Q = 1, Q2 = 1, dir = "==/<=")
```

Using the option `w.constr` in `scest()` (or `scpi()`) and the options `cov.adj` and `constant` in `scdata()` appropriately, i.e., setting $\mathcal{W}$ and $\mathcal{R}$ in (1), many synthetic control estimators proposed in the literature can be implemented. Table 3 provides a non-exhaustive list of such examples.

Tuning parameter choices

We provide rule-of-thumb choices of the tuning parameter Q for Lasso- and Ridge-type constraints.

- **Lasso** ($p = 1$)**.** Since Lasso is similar in spirit to the "simplex"-type traditional constraint in the synthetic control literature, we propose $Q = 1$ as a rule of thumb.
- **Ridge** ($p = 2$)**.** It is well known that the Ridge prediction problem can be equivalently formulated as an unconstrained penalized optimization problem and as a constrained optimization problem. More precisely, assuming $\mathbf{C}$ is not used and $M = 1$ for simplicity, the two Ridge-type problems are

$$\widehat{\mathbf{w}} := \underset{\mathbf{w} \in \mathbb{R}^J}{\arg\min} (\mathbf{A} - \mathbf{B}\mathbf{w})^\top \mathbf{V} (\mathbf{A} - \mathbf{B}\mathbf{w}) + \lambda ||\mathbf{w}||_2^2,$$

  where $\lambda \geq 0$ is a shrinkage parameter, and

$$\widehat{\mathbf{w}} := \underset{\mathbf{w} \in \mathbb{R}^J, ||\mathbf{w}||_2^2 \leq Q^2}{\arg\min} (\mathbf{A} - \mathbf{B}\mathbf{w})^\top \mathbf{V} (\mathbf{A} - \mathbf{B}\mathbf{w}),$$

  where $Q \geq 0$ is the (explicit) size of the constraint on the norm of $\mathbf{w}$. Under the assumption of Gaussian errors, a risk-minimizing choice (Hoerl, Kannard, and Baldwin 1975) of the standard shrinkage tuning parameter is

$$\lambda = J \widehat{\sigma}^2_{\text{OLS}} / ||\widehat{\mathbf{w}}_{\text{OLS}}||_2^2,$$

  where $\widehat{\sigma}^2_{\text{OLS}}$ and $\widehat{\mathbf{w}}_{\text{OLS}}$ are estimators of the variance of the pseudo-true residual $\mathbf{u}$ and the coefficients $\mathbf{w}_0$ based on least squares regression, respectively.

Since the two optimization problems above are equivalent, there exists a one-to-one correspondence between λ and Q. For example, assuming the columns of $\mathbf{B}$ are orthonormal, the closed-form solution for the Ridge estimator is $\widehat{\mathbf{w}} = (\mathbf{I} + \lambda\mathbf{I})^{-1}\widehat{\mathbf{w}}_{\text{OLS}}$, and it follows that if the constraint on the ℓ_2-norm is binding, then $Q = ||\widehat{\mathbf{w}}||_2 = ||\widehat{\mathbf{w}}_{\text{OLS}}||_2/(1+\lambda)$.

However, if $J > T_0$, $\widehat{\mathbf{w}}_{\text{OLS}}$ does not exist, hence we cannot rely on the approach suggested above. Indeed, the proposed mapping between λ and Q is ill-defined and we are unable to estimate λ. In this case, we first make the design low-dimensional by performing variable selection on $\mathbf{B}$ with Lasso. Once we select the columns of $\mathbf{B}$ whose Lasso coefficient is non-zero, we choose λ according to the rule of thumb described above.

If more than one feature is specified ($M > 1$), we compute the size of the constraint Q_l for each feature $l = 1, \dots, M$ and then select Q as the tightest constraint to favor shrinkage of $\mathbf{w}$, that is $Q := \min_{l=1,\dots,M} Q_l$.

Missing data

In case of missing values, we adopt different strategies depending on which units have missing entries and when these occur.

- *Missing pre-treatment data.* We compute $\widehat{\mathbf{w}}$ without the periods for which there is at least a missing entry for either the treated unit or one of the donors.

- *Missing post-treatment donor data.* Suppose that the ith donor has a missing entry in one of the M features in the post-treatment period $\widetilde{T}$; this implies that the predictor vector $\mathbf{p}_{\widetilde{T}}$ has a missing entry, and thus the synthetic unit and the associated prediction intervals are not available.

- *Missing post-treatment treated data.* When there is missing data for the treated unit after the treatment, the treatment effect τ_T is unavailable. However, prediction intervals for the synthetic point prediction of the counterfactual outcome, $Y_{1t}(0)$ for $t > T_0$, can still be computed in the usual way because they do not rely on the availability of such data points.

4. Uncertainty quantification

Following Cattaneo, Feng, and Titiunik (2021) and Cattaneo, Feng, Palomba, and Titiunik (2022), we view the quantity of interest τ_T within the synthetic control framework as a random variable, and hence we refrain from calling it a parameter. Building an analogy with the concept of estimand (or parameter of interest), we refer to τ_T as a *predictand*. Consequently, we prefer to call $\widehat{\tau}_T = Y_{1T}(1) - \widehat{Y}_{1T}(0)$ based on (3) a *prediction* of τ_T rather than an *estimator* of it, and our goal is to characterize the uncertainty of $\widehat{\tau}_T$ by building prediction intervals rather than confidence intervals. In practice, it is appealing to construct prediction intervals that are valid *conditional* on a set of observables. We let $\mathscr{H}$ be an information set generated by all features of control units and covariates used in the synthetic control construction, i.e., $\mathbf{B}$, $\mathbf{C}$, $\mathbf{x}_T$, and $\mathbf{g}_T$.

We first decompose the potential outcome of the treated unit based on $\mathbf{w}_0$ and $\mathbf{r}_0$ introduced in (2):

$$Y_{1T}(0) \equiv \mathbf{x}_T^\top \mathbf{w}_0 + \mathbf{g}_T^\top \mathbf{r}_0 + e_T = \mathbf{p}_T^\top \boldsymbol{\beta}_0 + e_T, \qquad T > T_0, \tag{4}$$

where e_T is defined by construction. In our analysis, $\mathbf{w}_0$ and $\mathbf{r}_0$ are assumed to be (possibly) random quantities around which $\widehat{\mathbf{w}}$ and $\widehat{\mathbf{r}}$ are concentrating in probability, respectively. Then, the distance between the predicted treatment effect on the treated and the target population one is

$$\widehat{\tau}_T - \tau_T = Y_{1T}(0) - \widehat{Y}_{1T}(0) = e_T - \mathbf{p}_T^\top(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0). \tag{5}$$

where e_T is the out-of-sample error coming from misspecification along with any additional noise occurring at the post-treatment period $T > T_0$, and the term $\mathbf{p}_T^\top(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ is the in-sample error coming from the construction of the synthetic control weights. Our goal is to find probability bounds on the two terms separately to give uncertainty quantification: for some pre-specified levels $\alpha_1, \alpha_2 \in (0, 1)$, with high probability over $\mathscr{H}$,

$$\mathsf{P}\big[M_{1,\mathtt{L}} \leq \mathbf{p}_T^\top(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) \leq M_{1,\mathtt{U}} \,\big|\, \mathscr{H}\big] \geq 1 - \alpha_1 \quad \text{and} \quad \mathsf{P}\big[M_{2,\mathtt{L}} \leq e_T \leq M_{2,\mathtt{U}} \,\big|\, \mathscr{H}\big] \geq 1 - \alpha_2.$$

It follows that these probability bounds can be combined to construct a prediction interval for τ_T with conditional coverage at least $1 - \alpha_1 - \alpha_2$: with high probability over $\mathscr{H}$,

$$\mathsf{P}\big[\widehat{\tau}_T + M_{1,\mathtt{L}} - M_{2,\mathtt{U}} \leq \tau_T \leq \widehat{\tau}_T + M_{1,\mathtt{U}} - M_{2,\mathtt{L}} \big| \mathscr{H}\big] \geq 1 - \alpha_1 - \alpha_2.$$

4.1. In-sample error

Cattaneo, Feng, and Titiunik (2021) provide a principled simulation-based method for quantifying the in-sample uncertainty coming from $\mathbf{p}_T^\top(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$. Let $\mathbf{Z} = (\mathbf{B}, \mathbf{C})$ and $\mathbf{D}$ be a non-negative diagonal (scaling) matrix of size d, possibly depending on the pre-treatment sample size T_0. Since $\widehat{\boldsymbol{\beta}}$ solves (1), $\widehat{\boldsymbol{\delta}} := \mathbf{D}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ is the optimizer of the centered criterion function:

$$\widehat{\boldsymbol{\delta}} = \operatorname*{arg\,min}_{\boldsymbol{\delta} \in \Delta} \{\boldsymbol{\delta}^\top \widehat{\mathbf{Q}} \boldsymbol{\delta} - 2\widehat{\boldsymbol{\gamma}}^\top \boldsymbol{\delta}\},$$

where $\widehat{\mathbf{Q}} = \mathbf{D}^{-1}\mathbf{Z}^\top \mathbf{V}\mathbf{Z}\mathbf{D}^{-1}$, $\widehat{\boldsymbol{\gamma}}^\top = \mathbf{u}^\top \mathbf{V}\mathbf{Z}\mathbf{D}^{-1}$, and $\Delta = \{\boldsymbol{h} \in \mathbb{R}^d : \boldsymbol{h} = \mathbf{D}(\boldsymbol{\beta} - \boldsymbol{\beta}_0), \boldsymbol{\beta} \in \mathcal{W} \times \mathcal{R}\}$. Recall that the information set conditional on which our prediction intervals are constructed contains $\mathbf{B}$ and $\mathbf{C}$. Thus, $\widehat{\mathbf{Q}}$ can be taken as fixed, and we need to characterize the uncertainty of $\widehat{\boldsymbol{\gamma}}$.

We construct a simulation-based criterion function accordingly:

$$\ell^\star(\boldsymbol{\delta}) = \boldsymbol{\delta}^\top \widehat{\mathbf{Q}} \boldsymbol{\delta} - 2(\mathbf{G}^\star)^\top \boldsymbol{\delta}, \qquad \mathbf{G}^\star \sim \mathsf{N}(\mathbf{0}, \widehat{\boldsymbol{\Sigma}}), \tag{6}$$

where $\widehat{\boldsymbol{\Sigma}}$ is some estimate of $\boldsymbol{\Sigma} = \mathbb{V}[\widehat{\boldsymbol{\gamma}} \mid \mathscr{H}]$ and $\mathsf{N}(0, \widehat{\boldsymbol{\Sigma}})$ represents the normal distribution with mean $\mathbf{0}$ and variance-covariance matrix $\widehat{\boldsymbol{\Sigma}}$. In Section 4.3 we give an explicit expression for the estimator $\widehat{\boldsymbol{\Sigma}}$ in Equation 8. In practice, the criterion function $\ell^\star(\cdot)$ can be simulated by simply drawing normal random vectors $\mathbf{G}^\star$.

Since the original constraint set Δ is infeasible, we need to construct a constraint set $\Delta^\star$ used in simulation that is close to Δ. Specifically, define the distance between a point $\mathbf{a} \in \mathbb{R}^d$ and a set $\Lambda \subseteq \mathbb{R}^d$ by

$$\operatorname{dist}(\mathbf{a}, \Lambda) = \inf_{\boldsymbol{\lambda} \in \Lambda} ||\mathbf{a} - \boldsymbol{\lambda}||,$$

where $||\cdot||$ is a generic ℓ_p vector norm on $\mathbb{R}^d$ with $p \geq 1$ (e.g., Euclidean norm or ℓ_1 norm). We require

$$\text{dist}(\mathbf{a}, \Delta^\star) \ll ||\mathbf{a}||, \quad \forall\, \mathbf{a} \in \Delta \cap \mathcal{B}(\mathbf{0}, \varepsilon), \tag{7}$$

where $\mathcal{B}(\mathbf{0}, \varepsilon)$ is an ε-neighborhood around zero for some $\varepsilon > 0$. In words, every point in the infeasible constraint set Δ has to be sufficiently *close* to the feasible constraint set used in simulation. We discuss below a principled strategy for constructing $\Delta^\star$, which allows for both linear and non-linear constraints in the feasibility set. Section 4.3 provides details on how $\Delta^\star$ is constructed and implemented in the **scpi** package.

Given the feasible criterion function $\ell^\star(\cdot)$ and constraint set $\Delta^\star$, we let

$$\begin{aligned} M_{1,\mathtt{L}} &:= (\alpha_1/2)\text{-quantile of } \inf \left\{ \mathbf{p}_T^\top \mathbf{D}^{-1} \boldsymbol{\delta} : \boldsymbol{\delta} \in \Delta^\star,\, \ell^\star(\boldsymbol{\delta}) \leq 0 \right\}, \qquad \text{and} \\ M_{1,\mathtt{U}} &:= (1-\alpha_1/2)\text{-quantile of } \sup \left\{ \mathbf{p}_T^\top \mathbf{D}^{-1} \boldsymbol{\delta} : \boldsymbol{\delta} \in \Delta^\star,\, \ell^\star(\boldsymbol{\delta}) \leq 0 \right\}, \end{aligned}$$

conditional on the data. Under mild regularity conditions, for a large class of synthetic control predictands (1), with high probability over $\mathscr{H}$,

$$\mathsf{P}\left[M_{1,\mathtt{L}} \leq \mathbf{p}_T^\top (\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) \leq M_{1,\mathtt{U}} \mid \mathscr{H} \right] \geq 1 - \alpha_1,$$

up to some small loss of the (conditional) coverage probability. Importantly, this conclusion holds whether the data are stationary or non-stationary and whether the model is correctly specified (i.e., $\mathsf{E}[\mathbf{u} \mid \mathscr{H}] = 0$) or not. If the constraints imposed are non-linear, an additional adjustment to this bound may be needed to ensure the desired coverage.

4.2. Out-of-sample error

The unobserved random variable e_T in (4) is a single error term in period T, which we interpret as the error from out-of-sample prediction, conditional on $\mathscr{H}$. Naturally, in order to have a proper bound on e_T, it is necessary to determine certain features of its conditional distribution $F_{e_T}(\mathsf{e}) = \mathsf{P}[e_T \leq \mathsf{e} \mid \mathscr{H}]$. In this section, we outline principled but agnostic approaches to quantify the uncertainty introduced by the post-treatment unobserved shock e_T. Since formalizing the validity of our methods usually requires strong assumptions, we also recommend a generic sensitivity analysis to incorporate out-of-sample uncertainty into the prediction intervals. See Section 4.5 and Section 5, in particular Figure 3 with the corresponding snippet of R code, for further clarifications on how to carry out sensitivity analysis on e_T.

- **Approach 1: Non-Asymptotic bounds**. The starting point is a non-asymptotic probability bound on e_T via concentration inequalities. For example, suppose that e_T is sub-Gaussian conditional on $\mathscr{H}$, i.e., there exists some $\sigma_{\mathscr{H}} > 0$ such that $\mathsf{E}[\exp(\lambda(e_T - \mathsf{E}[e_T \mid \mathscr{H}])) \mid \mathscr{H}] \leq \exp(\sigma^2_{\mathscr{H}} \lambda^2/2)$ a.s. for all $\lambda \in \mathbb{R}$. Then, we can take

  $$M_{2,\mathtt{L}} := \mathsf{E}[e_T \mid \mathscr{H}] - \sqrt{2\sigma^2_{\mathscr{H}} \log(2/\alpha_2)} \qquad \text{and} \qquad M_{2,\mathtt{U}} := \mathsf{E}[e_T \mid \mathscr{H}] + \sqrt{2\sigma^2_{\mathscr{H}} \log(2/\alpha_2)}.$$

  In practice, the conditional mean $\mathsf{E}[e_T \mid \mathscr{H}]$ and the sub-Gaussian parameter $\sigma_{\mathscr{H}}$ can be parameterized and/or estimated using the pre-treatment residuals.

- **Approach 2: Location-scale model**. Suppose that $e_T = \mathsf{E}[e_T \mid \mathscr{H}] + (\mathbb{V}[e_T \mid \mathscr{H}])^{1/2}\varepsilon_T$ with ε_T statistically independent of $\mathscr{H}$. This setting imposes restrictions on the distribution of $e_T \mid \mathscr{H}$, but allows for a much simpler tabulation strategy. Specifically, we can set the lower bound and upper bound on e_T as follows:

$$M_{2,\mathtt{L}} = \mathsf{E}[e_T \mid \mathscr{H}] + (\mathsf{VAR}[e_T \mid \mathscr{H}])^{1/2}\mathfrak{c}_\varepsilon(\alpha_2/2) \qquad \text{and}$$
$$M_{2,\mathtt{U}} = \mathsf{E}[e_T \mid \mathscr{H}] + (\mathsf{VAR}[e_T \mid \mathscr{H}])^{1/2}\mathfrak{c}_\varepsilon(1-\alpha_2/2),$$

  where $\mathfrak{c}_\varepsilon(\alpha_2/2)$ and $\mathfrak{c}_\varepsilon(1-\alpha_2/2)$ are $\alpha_2/2$ and $(1-\alpha_2/2)$ quantiles of ε_T, respectively. In practice, $\mathsf{E}[e_T \mid \mathscr{H}]$ and $\mathsf{VAR}[e_T \mid \mathscr{H}]$ can be parametrized and estimated using the pre-intervention residuals, or perhaps tabulated using auxiliary information. Once such estimates are available, the appropriate quantiles can be easily obtained using the standardized (estimated) residuals.

- **Approach 3: Quantile regression**. Another strategy to bound e_T is to determine the $\alpha_2/2$ and $(1-\alpha_2/2)$ conditional quantiles of $e_T \mid \mathscr{H}$, that is,

$$M_{2,\mathtt{L}} := (\alpha_2/2)\text{-quantile of } e_T \mid \mathscr{H} \qquad \text{and} \qquad M_{2,\mathtt{U}} := (1-\alpha_2/2)\text{-quantile of } e_T \mid \mathscr{H}.$$

  Consequently, we can employ quantile regression methods to estimate those quantities using pre-treatment data.

Using any of the above methods, we have the following probability bound on e_T:

$$\mathsf{P}\big[M_{2,\mathtt{L}} \leq e_T \leq M_{2,\mathtt{U}} \mid \mathscr{H}\big] \geq 1-\alpha_2.$$

4.3. Implementation

We now discuss the implementation details. The function `scpi()`, through various options, allows the user to specify different approaches to quantify in-sample and out-of-sample uncertainty based on the methods described above. Most importantly, `scpi()` permits modelling separately the in-sample error $\mathbf{p}_T^\top(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}_0)$ and the out-of-sample error e_T. In addition, the user can provide bounds on them manually with the options `w.bounds` and `e.bounds`, respectively, which can be useful for sensitivity analysis in empirical applications.

Modelling in-sample uncertainty

In-sample uncertainty stems from the prediction of $\mathbf{p}_T^\top(\widehat{\boldsymbol{\beta}}-\boldsymbol{\beta}_0)$, and its quantification reduces to determining $M_{1,\mathtt{L}}$ and $M_{1,\mathtt{U}}$. We first review the methodological proposals for constructing the constraint set $\Delta^\star$ used in simulation discussed in Cattaneo, Feng, Palomba, and Titiunik (2022), and then present the main procedure for constructing bounds on the in-sample error.

Constructing $\Delta^\star$. Our in-sample uncertainty quantification requires the centered and scaled constraint feasibility set Δ to be locally identical to (or, at least, well approximated by) the constraint set $\Delta^\star$ used in simulation described in (6), in the sense of (7). Suppose that

$$\mathcal{W} \times \mathcal{R} = \Big\{\boldsymbol{\beta} \in \mathbb{R}^d : \boldsymbol{m}_{\text{eq}}(\boldsymbol{\beta}) = \mathbf{0}, \boldsymbol{m}_{\text{in}}(\boldsymbol{\beta}) \leq \mathbf{0}\Big\},$$

where $\boldsymbol{m}_{\text{eq}}(\cdot) \in \mathbb{R}^{d_{\text{eq}}}$ and $\boldsymbol{m}_{\text{in}}(\cdot) \in \mathbb{R}^{d_{\text{in}}}$ and denote the jth constraint in $\boldsymbol{m}_{\text{in}}(\cdot)$ as $m_{\text{in},j}(\cdot)$. Given tuning parameters $\varrho_j > 0$, $j = 1, \cdots, d_{\text{in}}$, let $\mathcal{B}$ be the set of indices for the inequality constraints such that $m_{\text{in},j}(\hat{\boldsymbol{\beta}}) > -\varrho_j$. Then, we construct $\Delta^\star$ as

$$\Delta^\star = \Big\{ \mathbf{D}(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}) : \boldsymbol{m}_{\text{eq}}(\boldsymbol{\beta}) = \mathbf{0}, m_{\text{in},j}(\boldsymbol{\beta}) \leq m_{\text{in},j}(\widehat{\boldsymbol{\beta}}) \text{ for } j \in \mathcal{B}, \text{ and } m_{\text{in},l}(\boldsymbol{\beta}) \leq 0 \text{ for } l \notin \mathcal{B} \Big\}.$$

In practice, we need to choose possibly heterogeneous parameters ϱ_j, $j = 1, \ldots, d_{\text{in}}$, for different inequality constraints. Our proposed choice of ϱ_j is

$$\varrho_j := \Big|\Big| \frac{\partial}{\partial \boldsymbol{\beta}} m_{\text{in},j}(\widehat{\boldsymbol{\beta}}) \Big|\Big|_1 \times \varrho, \quad j = 1, \ldots, d_{\text{in}},$$

for some parameter ϱ where $||\cdot||_1$ denotes the ℓ_1-norm. We estimate ϱ according to the following formula if $M = 1$:

$$\varrho = \mathcal{C} \frac{\log(T_0)^c}{T_0^{1/2}},$$

where $c = 1/2$ if the data are i.i.d. or weakly dependent, and $c = 1$ if $\mathbf{A}$ and $\mathbf{B}$ form a cointegrated system, while $\mathcal{C}$ is one of the following:

$$\mathcal{C}_1 = \frac{\widehat{\sigma}_u}{\min_{1 \leq j \leq J} \widehat{\sigma}_{b_j}}, \qquad \mathcal{C}_2 = \frac{\max_{1 \leq j \leq J} \widehat{\sigma}_{b_j} \widehat{\sigma}_u}{\min_{1 \leq j \leq J} \widehat{\sigma}^2_{b_j}}, \qquad \mathcal{C}_3 = \frac{\max_{1 \leq j \leq J} \widehat{\sigma}_{b_j u}}{\min_{1 \leq j \leq J} \widehat{\sigma}^2_{b_j}},$$

with $\mathcal{C}_1$ as the default. $\widehat{\sigma}_{b_j,u}$ is the estimated (unconditional) covariance between the pseudo-true residual $\mathbf{u}$ and the feature of the jth control unit $\mathbf{B}_{j,1}$, and $\widehat{\sigma}_u$ and $\widehat{\sigma}_{b_j}$ are the estimated (unconditional) standard deviations of, respectively, $\mathbf{u}$ and $\mathbf{B}_{j,1}$. In the case of multiple features ($M > 1$), the package employs the same construction above after stacking the data.

Degrees-of-Freedom Correction. Our uncertainty quantification strategy requires an estimator of the conditional variance $\mathsf{VAR}[\mathbf{u} \mid \mathscr{H}]$, which may rely on the effective degrees of freedom, df, of the synthetic control method. In general, there exists no exact correspondence between the degrees of freedom and the number of parameters in a fitting model (Ye 1998). Therefore, the estimated degrees of freedom, $\widehat{\mathsf{df}}$, are defined according to the chosen constraint sets for $\boldsymbol{\beta}$ underlying the estimation procedure in (1):

- **OLS**. $\widehat{\mathsf{df}} = J + KM$.
- **Lasso**. Following Zou, Hastie, and Tibshirani (2007), an unbiased and consistent estimator of df is $\widehat{\mathsf{df}} = \sum_{j=1}^{J} \mathbb{1}(\widehat{w}_j > 0) + KM$ where $\widehat{w}_j$ is the jth element of the constructed weights $\widehat{\mathbf{w}}$.
- **Simplex**. Following the discussion for Lasso, $\widehat{\mathsf{df}} = \sum_{j=1}^{J} \mathbb{1}(\widehat{w}_j > 0) - 1 + KM$.
- **Ridge**. Let $s_1 \geq s_2 \geq \cdots \geq s_J \geq 0$ be singular values of $\mathbf{B}$ and λ be the complexity parameter of the corresponding Lagrangian Ridge problem, which satisfies $\lambda \widehat{\mathbf{w}} = \mathbf{B}^\top (\mathbf{A} - \mathbf{B}\widehat{\mathbf{w}})$. Then, following Friedman, Hastie, and Tibshirani (2001), $\widehat{\mathsf{df}} = \sum_{j=1}^{J} \frac{s_j^2}{s_j^2 + \lambda} + KM$.

Main Procedure. Given the constraint set $\Delta^\star$, the main procedure for computing the upper and lower bounds on the in-sample error is as follows:

Step 1. *Estimation of conditional moments of* $\mathbf{u}$. To estimate $\mathbf{\Sigma}$ and to simulate the criterion function (6) we need an estimate of $\mathsf{VAR}[\widehat{\gamma} \mid \mathscr{H}]$ which, in turn, depends on the conditional moments of $\mathbf{u}$. To estimate such moments, the user needs to specify three things:

i) whether the model is misspecified or not, via the option `u.missp`.

ii) how to model $\mathbf{u}$, via the options `u.order`, `u.lags`, and `u.design`.

iii) an estimator of $\mathsf{VAR}[\mathbf{u} \mid \mathscr{H}]$, via the option `u.sigma`.

Given the constructed weights $\widehat{\mathbf{w}} = (\widehat{w}_2, \cdots, \widehat{w}_{J+1})^\top$, define regularized weights $\widehat{\mathbf{w}}^\star = (\widehat{w}_2^\star, \cdots, \widehat{w}_{J+1}^\star)^\top$ with $\widehat{w}_j^\star = \widehat{w}_j \mathbb{1}(\widehat{w}_j > \varrho)$ for the tuning parameter ϱ specified previously. Let $\mathbf{B}^\star = \mathrm{diag}(\mathbf{B}_1^\star, \mathbf{B}_2^\star, \ldots, \mathbf{B}_M^\star)$, where $\mathbf{B}_l^\star$ denotes the matrix composed of the columns of $\mathbf{B}_l$ with non-zero regularized weight $\widehat{w}_j^\star$ only. When `cointegrated.data` in `scdata()` is set to be `TRUE`, rather than the columns of $\mathbf{B}_l$, we take the first difference of the columns of $\mathbf{B}_l$. If the user inputs `u.missp = FALSE`, then it is assumed that $\mathbb{E}[\mathbf{u} \mid \mathscr{H}] = 0$, whereas if `u.missp = TRUE` (default), then $\mathbb{E}[\mathbf{u} \mid \mathscr{H}]$ needs to be estimated.

The unknown conditional expectation $\mathbb{E}[\mathbf{u} \mid \mathscr{H}]$ is estimated using the fitted values of a flexible linear-in-parameters regression of $\widehat{\mathbf{u}} = \mathbf{A} - \mathbf{B}\widehat{\mathbf{w}} - \mathbf{C}\widehat{\mathbf{r}}$ on a design matrix $\mathbf{D}_\mathbf{u}$, which can be provided directly with the option `u.design` or by specifying the lags of $\mathbf{B}^\star$ (`u.lags`) and/or the order of the fully interacted polynomial in $\mathbf{B}^\star$ (`u.order`).

For example, if the user specifies `u.lags = 1` and `u.order = 1`, then the design matrix is $\mathbf{D}_\mathbf{u} = [\mathbf{B}^\star \;\; \mathbf{B}_{-1}^\star \;\; \mathbf{C}]$, where $\mathbf{B}_{-1}^\star$ indicates the first lag of $\mathbf{B}^\star$. If, instead, `u.order = 0` and `u.lags = 0` are specified, then $\widehat{\mathbb{E}}[\mathbf{u} \mid \mathscr{H}] = \overline{\mathbf{u}} \otimes \boldsymbol{\iota}_{T_0}$, where $\overline{\mathbf{u}} = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_M)^\top$ with $\overline{u}_l = T_0^{-1} \sum_{t=1}^{T_0} \widehat{u}_{t,l}$, $\boldsymbol{\iota}_\nu$ is a $\nu \times 1$ vector of ones, and $\otimes$ denotes the Kronecker product.

The conditional variance of $\mathbf{u}$ is estimated as

$$\widehat{\mathsf{VAR}}[\mathbf{u} \mid \mathscr{H}] = \mathrm{diag}\left(\mathtt{vc}_1(\widehat{u}_{1,1} - \widehat{\mathbb{E}}[u_{1,1} \mid \mathscr{H}])^2, \cdots, \mathtt{vc}_{T_0 \cdot M}(\widehat{u}_{T_0,M} - \widehat{\mathbb{E}}[u_{T_0,M} \mid \mathscr{H}])^2\right)$$

where $\mathtt{vc}_i$, $i = 1, \cdots, T_0 \cdot M$ is a sequence of variance-correction constants, which can be chosen among the well-known family of heteroskedasticity-robust variance-covariance estimators through the option `u.sigma`. In particular, the package currently allows for five choices:

$$\mathtt{vc}_i^{(0)} = 1, \quad \mathtt{vc}_i^{(1)} = \frac{T_0 \cdot M}{T_0 \cdot M - \mathtt{df}}, \quad \mathtt{vc}_i^{(2)} = \frac{1}{1 - \mathbf{L}_{ii}},$$
$$\mathtt{vc}_i^{(3)} = \frac{1}{(1 - \mathbf{L}_{ii})^2}, \quad \mathtt{vc}_i^{(4)} = \frac{1}{(1 - \mathbf{L}_{ii})^{\delta_i}}$$

with $\mathbf{L}_{ii}$ being the i-th diagonal entry of the leverage matrix $\mathbf{L} := \mathbf{Z}(\mathbf{Z}^\top \mathbf{V} \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{V}$, $\delta_i = \min\{4, T_0 \cdot M \cdot \mathbf{P}_{ii}/\mathtt{df}\}$, and $\mathtt{df}$ is a degrees-of-freedom correction factor, whose estimation has been explained before. For a discussion of these different estimators, we refer the reader to MacKinnon (2013) and references therein.

Step 2. *Estimation of* $\mathbf{\Sigma}$. The estimator of $\mathbf{\Sigma}$ is

$$\widehat{\mathbf{\Sigma}} = (\mathbf{Z}^\top \mathbf{V}) \widehat{\mathsf{VAR}}[\mathbf{u} \mid \mathscr{H}] (\mathbf{V} \mathbf{Z}). \tag{8}$$

Step 3. *Simulation.* The criterion function $\ell^\star(\boldsymbol{\delta})$ in (6) is simulated by drawing i.i.d. random vectors from the Gaussian distribution $\mathsf{N}(0, \widehat{\boldsymbol{\Sigma}})$, conditional on the data.

Step 4. *Optimization.* Let $\ell^\star_{(s)}(\boldsymbol{\delta})$ denote the criterion function corresponding to the s-th draw from $\mathsf{N}(0, \widehat{\boldsymbol{\Sigma}})$. For each draw s, we solve the following constrained problems:

$$l_{(s)} := \inf_{\boldsymbol{\delta}\in\Delta^\star,\, \ell^\star_{(s)}(\boldsymbol{\delta})\leq 0} \mathbf{p}_T^\top \mathbf{D}^{-1}\boldsymbol{\delta} \qquad \text{and} \qquad u_{(s)} := \sup_{\boldsymbol{\delta}\in\Delta^\star,\, \ell^\star_{(s)}(\boldsymbol{\delta})\leq 0} \mathbf{p}_T^\top \mathbf{D}^{-1}\boldsymbol{\delta}, \tag{9}$$

where $\Delta^\star$ is constructed as explained previously.

Step 5. *Estimation of $M_{1,\mathtt{L}}$ and $M_{1,\mathtt{U}}$.* Step 4 is repeated S times, where S can be specified with the option `sims`. Then, $M_{1,\mathtt{L}}$ is the $(\alpha_1/2)$-quantile of $\{l_{(s)}\}_{s=1}^S$ and $M_{1,\mathtt{U}}$ is the $(1-\alpha_1/2)$-quantile of $\{u_{(s)}\}_{s=1}^S$. The level of α_1 can be chosen with the option `u.alpha`.

Execution Speed and Parallelization. Steps 3 and 4 of the procedure above are the most computationally intensive and we optimize them in two ways. First, to solve the optimization problem in (9), **scpi** relies on **ECOSolveR** (Fu and Narasimhan 2023), an R package that provides an interface to `ECOS`, an efficient solver for conic problems (Domahidi, Chu, and Boyd 2013; Fu, Narasimhan, and Boyd 2020). See Cattaneo, Feng, Palomba, and Titiunik (2022) for more details on how to cast the different constrained SC methods into conic optimization problems. To give the reader a sense of the speed improvement, Table 4 compares the execution speed of the conic solver we rely on (first column) with other two popular optimizers in R: **nloptr** (Ypma and Johnson 2024) and **CVXR** (Fu, Narasimhan, and Boyd 2020). **nloptr** is an R interface to `NLopt`, a free/open-source library for nonlinear optimization. Similarly, **CVXR** is an R package that provides a convex modelling language and interface to solvers like `ECOS`. A nice feature of **CVXR** is that it analyzes the optimization problem given as input, checks whether this problem is (disciplined) convex (so called, DCP check), and then casts it in the form required by the particular solver that gets called (e.g., conic form). This extra step of checking the nature of the problem and rewriting it might be time-consuming. However, if one knows that the problem is convex, the DCP check can be turned off. In Table 4 we report the performance of **CVXR** both when the DCP check is on and when it is turned off.

The underlying optimization problem is the minimization problem in (9), where $\mathcal{W}$ is a simplex-type constraint and J, KM, and M are chosen to replicate the size of the empirical application in Section 5. We evaluate the performance of each package through the R function `Sys.time()`. The first row of the table reports the median computation time of each optimizer, whereas the second row shows the inter-quartile range. On the one hand, using a conic solver in place of a solver for more generic optimization programs (like **nloptr**) makes our software 2.5 times faster. On the other hand, our software is tailored to rewrite the synthetic control problem as a conic problem. This gives a 100-fold gain in speed when compared to **CVXR** when the DCP check is on and a 1.5-fold gain when the DCP check is switched off and the data is prepared beforehand.

Moreover, **scpi** can be sped up further by efficient parallelization of the tasks performed via base R functions which assign different simulations to different cores. If $\mathtt{N}_{\text{cores}}$ cores are used, the final execution time would be approximately $\mathtt{T}_{\text{exec}}/\mathtt{N}_{\text{cores}}$, where $\mathtt{T}_{\text{exec}}$ is the execution time when a single core is used.

	scpi	**CVXR** *DCP on*	**CVXR** *DCP off*	**nloptr**
Median	0.723	99.935	1.060	1.767
IQR	[0.715, 0.732]	[98.396, 100.876]	[1.042, 1.067]	[1.748, 1.810]

Table 4: Speed comparison across optimizers (*units*: milliseconds), 100 simulations.

A recently released alternative optimization package is **Clarabel** (Narasimhan, Goulart, and Chen 2024), which provides a new and versatile interior point solver for conic programs. While beyond the scope of this paper, we plan to investigate the performance of this package in future work.

Modelling out-of-sample uncertainty

To quantify the uncertainty coming from e_T, we need to impose some probabilistic structure that allows us to model the distribution $\mathsf{P}[e_T \leq \mathtt{e} \mid \mathscr{H}]$ and, ultimately, estimate $M_{2,\mathtt{L}}$ and $M_{2,\mathtt{U}}$. We discussed three different alternative approaches: (i) non-asymptotic bounds; (ii) location-scale model; and (iii) quantile regression. The user can choose the preferred way of modeling $e_T \mid \mathscr{H}$ by setting the option `e.method` to either `"gaussian"`, `"ls"`, or `"qreg"`.

The user can also choose the information used to estimate (conditional) moments or quantiles of $e_T \mid \mathscr{H}$. Practically, we allow the user to specify a design matrix $\mathbf{D_e}$ that is then used to run the appropriate regressions depending on the approach requested. By default, we set $\mathbf{D_e} = [\mathbf{B}_1^\star \;\; \mathbf{C}_1]$. Alternatively, the matrix $\mathbf{D_e}$ can be provided directly through the option `e.design` or by specifying the lags of $\mathbf{B}_1^\star$ (`e.lags`) and/or the order of the fully interacted polynomial in $\mathbf{B}_1^\star$ (`e.order`). For example, if the user specifies `e.lags = 0` and `e.order = 2`, then $\mathbf{D_e}$ contains $\mathbf{B}_1^\star$, $\mathbf{C}_1$, and all the unique second-order terms generated by the interaction of the columns of $\mathbf{B}_1^\star$. If instead `e.order = 0` and `e.lags = 0` are set, then $\widehat{\mathbb{E}}[e_T \mid \mathscr{H}]$ and $\widehat{\mathbb{V}}[e_T \mid \mathscr{H}]$ are estimated using the sample average and the sample variance of e_T using the pre-intervention data. Recall that if the option `cointegrated.data` is set to `TRUE`, $\mathbf{B}_1^\star$ is formed using the first differences of the columns in $\mathbf{B}_1$. Finally, the user can specify α_2 with the option `e.alpha`.

4.4. Simultaneous prediction intervals

Up to this point, we focused on prediction intervals that possess high coverage for the individual treatment effect in *each* period. However, it may be desirable to have prediction intervals that have high *simultaneous* coverage for several periods, usually known as *simultaneous prediction intervals* in the literature. In other words, our final goal is to construct a sequence of intervals $\{\mathcal{I}_t : T_0 + 1 \leq t \leq T_0 + L\}$ for some $1 \leq L \leq T_1$ such that with high probability over $\mathscr{H}$,

$$\mathsf{P}\big[\tau_t \in \mathcal{I}_t, \text{ for all } T_0 + 1 \leq t \leq T_0 + L \,\big|\, \mathscr{H}\big] \geq 1 - \alpha_1 - \alpha_2.$$

To construct such intervals, we need to generalize the procedures described above to quantify the in-sample error (Section 4.1) and the out-of-sample error (Section 4.2).

With regard to the in-sample uncertainty, we handle two separate cases. On the one hand, if

the constraints in Δ are linear (e.g., simplex or lasso), then

$$M_{1,\mathtt{L}} := (\alpha_1/2)\text{-quantile of } \inf\left\{\mathbf{p}_t^\top \mathbf{D}^{-1}\boldsymbol{\delta} :\ \boldsymbol{\delta}\in\Delta^\star,\ \ell^\star(\boldsymbol{\delta})\leq 0,\ T_0+1\leq t\leq T_0+L\right\} \text{ and}$$
$$M_{1,\mathtt{U}} := (1-\alpha_1/2)\text{-quantile of } \sup\left\{\mathbf{p}_t^\top \mathbf{D}^{-1}\boldsymbol{\delta} :\ \boldsymbol{\delta}\in\Delta^\star,\ \ell^\star(\boldsymbol{\delta})\leq 0,\ T_0+1\leq t\leq T_0+L\right\},$$

which guarantees that with high probability over $\mathscr{H}$

$$\mathsf{P}\big[M_{1,\mathtt{L}} \leq \mathbf{p}_t^\top(\boldsymbol{\beta}_0-\widehat{\boldsymbol{\beta}}) \leq M_{1,\mathtt{U}},\ \text{for all } T_0+1\leq t\leq T_0+L \,\big|\, \mathscr{H}\big] \geq 1-\alpha_1.$$

On the other hand, if Δ includes non-linear constraints (e.g., constraints involving the ℓ_2 norm), it is necessary to decrease the lower bound $M_{1,\mathtt{L}}$ and increase the upper bound $M_{1,\mathtt{U}}$ by some quantity $\varepsilon_{\Delta,t}>0$ for each $T_0+1\leq t\leq T_0+L$. To give an example of what $\varepsilon_{\Delta,t}$ looks like, in the case of ridge-type constraints we have

$$\varepsilon_{\Delta,t} = ||\mathbf{p}_t||_1 \times (2||\widehat{\boldsymbol{\beta}}||_2)^{-1} \times \varrho^2,$$

and see Cattaneo, Feng, Palomba, and Titiunik (2022) for more general cases. With regard to the out-of-sample uncertainty, our proposed strategy is a generalization of "Approach 1" in Section 4.2: find $M_{2,\mathtt{L},t}$ and $M_{2,\mathtt{U},t}$ such that with high probability over $\mathscr{H}$,

$$\mathsf{P}\big[M_{2,\mathtt{L},t} \leq e_t \leq M_{2,\mathtt{U},t},\ \text{for all } T_0+1\leq t\leq T_0+L \,\big|\, \mathscr{H}\big] \geq 1-\alpha_2.$$

Suppose that each e_t, $T_0+1\leq t\leq T_0+L$, is sub-Gaussian conditional on $\mathscr{H}$ (not necessarily independent over t) with sub-Gaussian parameters $\sigma_{\mathscr{H},t}\leq\sigma_{\mathscr{H}}$ for some $\sigma_{\mathscr{H}}$. Then, we can take

$$M_{2,\mathtt{L},t} := \mathsf{E}[e_t|\mathscr{H}] - \sqrt{2\sigma^2_{\mathscr{H}}\log(2L/\alpha_2)} \qquad \text{and} \qquad M_{2,\mathtt{U},t} := \mathsf{E}[e_t|\mathscr{H}] + \sqrt{2\sigma^2_{\mathscr{H}}\log(2L/\alpha_2)}.$$

We can see that, compared to what we had for "Approach 1", there is an extra term, $\sqrt{\log L}$, which makes the simultaneous prediction intervals longer. The command `scpi` automatically computes simultaneous prediction intervals. To plot them, the user needs to specify the option `joint = TRUE` in the `scplot` command.

4.5. Sensitivity analysis

While the three approaches for out-of-sample uncertainty quantification described in Section 4.2 are simple and intuitive, their validity requires potentially strong assumptions on the underlying data generating process that links the pre-treatment and post-treatment data. Such assumptions are difficult to avoid because the ultimate goal is to learn about the statistical uncertainty introduced by a single unobserved random variable after the treatment/intervention is deployed, that is, $e_T \mid \mathscr{H}$ for some $T>T_0$. Without additional data availability, or specific modelling assumptions allowing for transferring information from the pre-treatment period to the post-treatment period, it is difficult to formally construct $M_{2,\mathtt{L}}$ and $M_{2,\mathtt{U}}$ using data-driven methods.

We suggest approaching the out-of-sample uncertainty quantification as a principled sensitivity analysis, using the approaches above as a starting point. Given the formal and detailed in-sample uncertainty quantification described previously, it is natural to progressively enlarge the final prediction intervals by adding additional out-of-sample uncertainty to ask the

question: how large does the additional out-of-sample uncertainty contribution coming from $e_T \mid \mathscr{H}$ need to be in order to render the treatment effect τ_T statistically insignificant? Using the approaches above, or similar ones, it is possible to construct natural initial benchmarks. For instance, to implement Approach 1, one can use the pre-treatment outcomes or synthetic control residuals to obtain a "reasonable" benchmark estimate of the sub-Gaussian parameter $\sigma_{\mathscr{H}}$ and then progressively enlarge or shrink this parameter to check the robustness of the conclusion. Alternatively, in specific applications, natural levels of uncertainty for the outcomes of interest could be available, and hence used to tabulate the additional out-of-sample uncertainty. We illustrate this approach in Section 5.

5. Empirical illustration

We showcase the features of the package **scpi** using real data. For comparability purposes, we employ the canonical dataset in the synthetic control literature on the economic consequences of the 1990 German reunification (Abadie 2021), and focus on quantifying the causal impact of the German reunification on GDP per capita in West Germany. Thus, we compare the post-reunification outcome of West Germany with the outcome of a synthetic control unit constructed using 16 OECD countries from 1960 to 1990. Using the notation introduced above, we have $T_0 = 31$ and $J = 16$. The only feature we exploit to construct the synthetic control is yearly GDP per capita, and we add a constant term for covariate adjustment. Thus $M = 1$ and $K = 1$, and $\mathcal{R} = \mathbb{R}$. We explore the effect of the reunification from 1991 to 2003, hence $T_1 = 13$. Finally, we treat the time series for West Germany and those countries in the donor pool as a cointegrating system. Given this information, the command `scdata()` prepares all the matrices needed in the synthetic control framework described above (**A**, **B**, **C** and **P**), and returns an object that must be used as input in either `scest()` to predict $Y_{1T}(0)$, $T > T_0$, or `scpi()` to conduct inference on τ_T, $T > T_0$.

We first call `scdata()` to transform any data frame into an object of class '`scpi_data`'.

```
R> data <- scpi_germany
R> id.var <- "country"
R> time.var <- "year"
R> period.pre <- (1960:1990)
R> period.post <- (1991:2003)
R> unit.tr <- "West Germany"
R> unit.co <- setdiff(unique(data$country), unit.tr)
R> outcome.var <- "gdp"
R> constant <- TRUE
R> cointegrated.data <- TRUE
R> df  <-   scdata(df = data, id.var = id.var, time.var = time.var,
+    outcome.var = outcome.var, period.pre = period.pre,
+    period.post = period.post, unit.tr = unit.tr, unit.co = unit.co,
+    constant = constant, cointegrated.data = cointegrated.data)
```

The function `scdata()` has the following options: `df` inputs the data; `id.var` specifies the name of the variable containing unit identifiers; `time.var` declares the name of the variable denoting the time periods; `period.pre` and `period.post` set the pre-treatment and post-treatment periods, respectively; `unit.tr` and `unit.co` declare the treated unit and donor

units, respectively; `outcome.var` specifies the name of the outcome variable; `constant` is a logical variable used to include a constant term; `cointegrated.data` is a logical variable used to take cointegration into account.

After having prepared the data, the next step involves choosing the desired constraint set $\mathcal{W}$ to construct the vector of weights $\mathbf{w}$. We consider the canonical synthetic control method and thus search for optimal weights in $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}_+^J : ||\mathbf{w}||_1 = 1\}$. This constraint set is the default in `scest()` and, consequently, in `scpi()`, as the latter internally calls the former to construct $\mathbf{w}$. The snippet below illustrates how to call `scest()` and reports the results displayed in the console with the `summary()` method.

```
R> res.est <- scest(data = df, w.constr = list(name = "simplex"))
R> summary(res.est)

Synthetic Control Estimation - Setup

Constraint Type:                            simplex
Constraint Size (Q):                        1
Treated Unit:                               West Germany
Size of the donor pool:                     16
Features:                                   1
Pre-treatment period:                       1960-1990
Pre-treatment periods used in estimation:   31
Covariates used for adjustment:             1

Synthetic Control Estimation - Results

Active donors: 6

Coefficients:
            Weights
Australia     0.000
Austria       0.441
Belgium       0.000
Denmark       0.000
France        0.000
Greece        0.000
Italy         0.177
Japan         0.013
Netherlands   0.059
New Zealand   0.000
Norway        0.000
Portugal      0.000
Spain         0.000
Switzerland   0.036
UK            0.000
USA           0.274
           Covariates
```

```
0.constant      0.158
```

The next step is uncertainty quantification using `scpi()`. In this case, we quantify the in-sample and out-of-sample uncertainty the same way, using $\mathbf{B}$ and $\mathbf{C}$ as the conditioning set in both cases. To do so, it suffices to set the order of the polynomial in $\mathbf{B}$ to 1 (`u.order <- 1` and `e.order <- 1`) and not include lags (`u.lags <- 0` and `e.lags <- 0`). Furthermore, by specifying the option `u.miss <- TRUE`, we take into account that the conditional mean of $\mathbf{u}$ might differ from 0. This option, together with `u.sigma <- "HC1"`, specifies the following estimator of $\mathsf{VAR}[\mathbf{u} \mid \mathscr{H}]$:

$$\widehat{\mathsf{VAR}}[\mathbf{u} \mid \mathscr{H}] = \operatorname{diag}\left(\mathtt{vc}_1^{(1)}(\widehat{\mathbf{u}}_1 - \widehat{\mathsf{E}}[\mathbf{u}_1 \mid \mathscr{H}])^2, \cdots, \mathtt{vc}_{T_0}^{(1)}(\widehat{\mathbf{u}}_{T_0} - \widehat{\mathsf{E}}[\mathbf{u}_{T_0} \mid \mathscr{H}])^2\right).$$

Finally, by selecting `e.method <- "gaussian"`, we perform out-of-sample uncertainty quantification treating e_T as sub-Gaussian conditional on $\mathbf{B}$ and $\mathbf{C}$. As a last step, we visualize the constructed synthetic control and compare it with the observed time series for the treated unit, taking advantage of the function `scplot()`.

```
R> sims <- 1000
R> u.order <- 1
R> u.lags <- 0
R> u.sigma <- "HC1"
R> u.missp <- TRUE
R> e.order <- 1
R> e.lags <- 0
R> e.method <- "gaussian"
R> lgapp <- "linear"
R> cores <- 1
R> set.seed(8894)
R> res.pi <- scpi(data = df, sims = sims, e.method = e.method,
+    e.order = e.order, e.lags = e.lags, u.order = u.order,
+    u.lags = u.lags, lgapp = lgapp, u.sigma = u.sigma,
+    u.missp = u.missp, cores = cores, w.constr = list(name = "simplex"))
R> plot <- plot <- scplot(result = res.pi, plot.range = (1960:2003),
+    label.xy = list(x.lab = "Year",
+         y.lab = "GDP per capita (thousand US dollars)"),
+    x.ticks = c(1960, 1970, 1980, 1990, 2000, 2003), e.out = TRUE)
R> plot$plot_out + ggtitle("")
```

Figure 1 displays the plot resulting from the `scplot` call. The black line shows the level of the outcome for the treated unit, $Y_{1t}(1)$, $t = 1963, \ldots, 2003$, whilst the blue line shows the level of the outcome for the synthetic control, $\widehat{Y}_{1t}(0)$, $t = 1963, \ldots, 2003$. The blue bars report 90% prediction intervals for $Y_{1t}(0)$. In-sample uncertainty is quantified by means of 1000 simulations of (9), whereas out-of-sample uncertainty is quantified through sub-Gaussian bounds. The vertical bars are 90% prediction intervals, where the non-coverage error rate is halved between the out-of-sample and the in-sample uncertainty quantification, i.e., $\alpha_1 = \alpha_2 = 0.05$.

We also conduct the same exercise using different choices of $\mathcal{W}$ (see Table 2). In particular, we construct weights and compute prediction intervals using four other specifications: (i) a

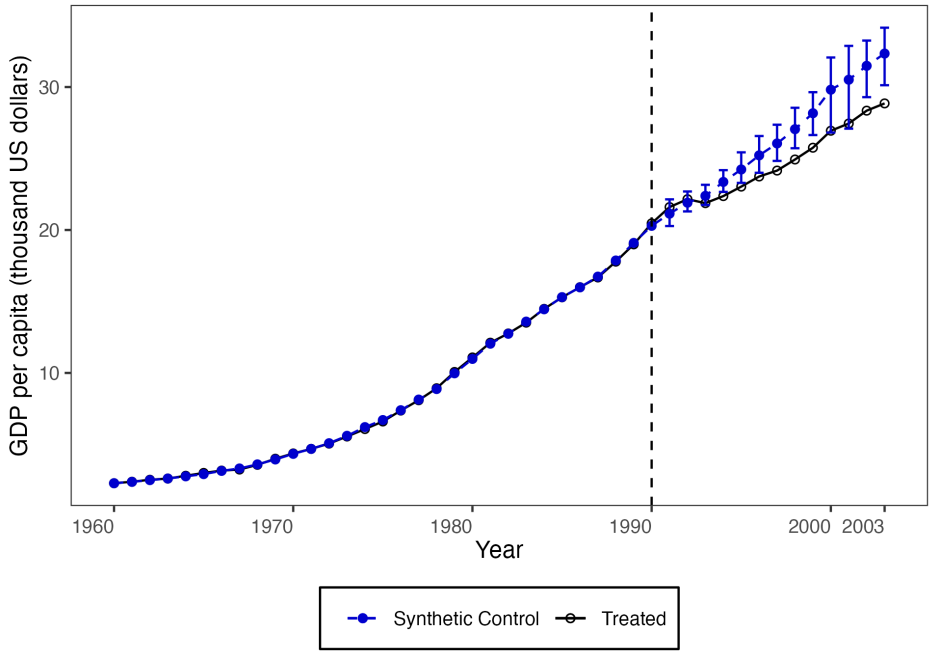
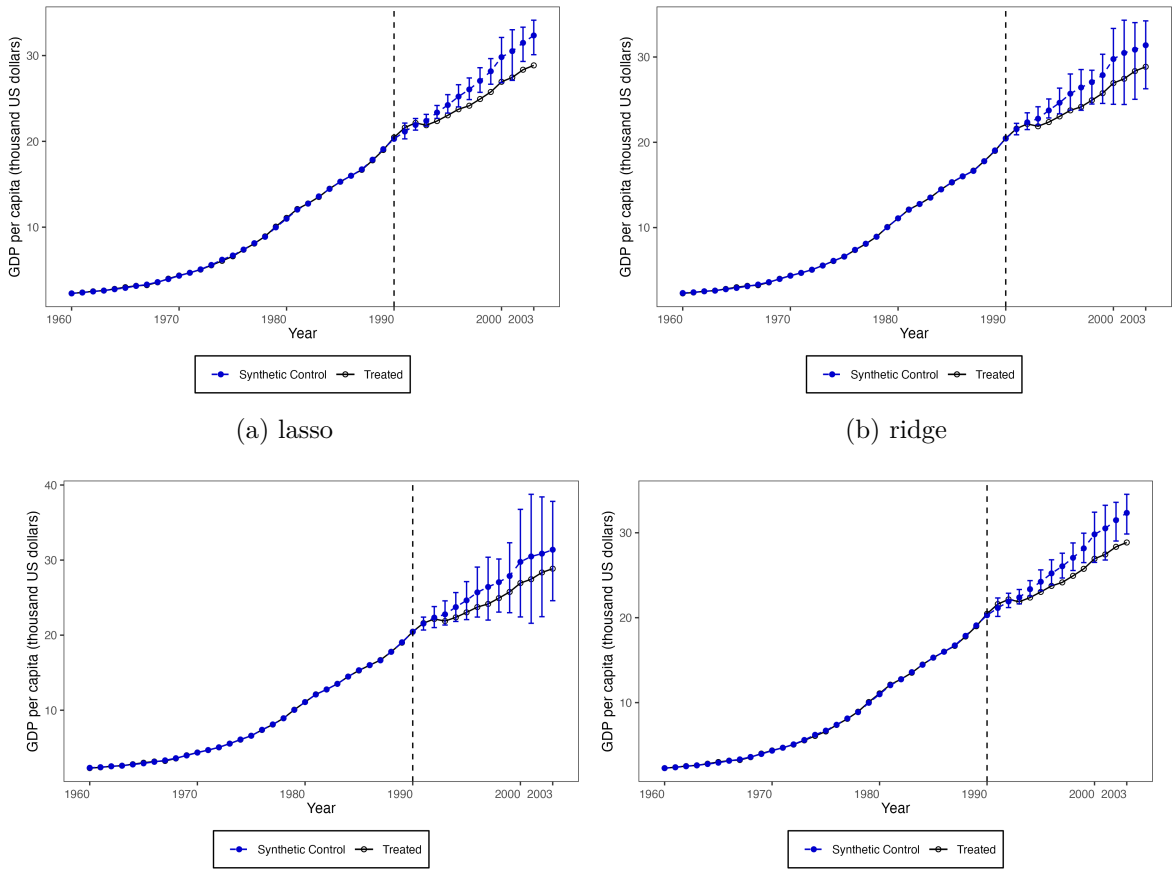


Figure 1: Treated and synthetic unit using a simplex-type $\mathcal{W}$ and 90% prediction intervals, German reunification example. *Note: Dashed vertical line indicates the period when the treatment (German reunification) occurs.*

lasso-type constraint (Figure 2a), (*ii*) a ridge-type constraint (Figure 2b), (*iii*) no constraint (Figure 2c), and (*iv*) an L1-L2 constraint. In panel (b), $Q = 0.906$, whereas in panel (d) $Q = 1, Q_2 = 0.906$.

```
R> methods <- c("lasso", "ols", "ridge", "L1-L2")
R> for (method in methods) {
+    if (method %in%  c("ridge", "L1-L2")) lgapp <- "generalized"
+    set.seed(8894)
+    res.pi  <- scpi(data = df, sims = sims, e.method = e.method,
+      e.order = e.order, e.lags = e.lags, u.order = u.order,
+      u.lags = u.lags, lgapp = lgapp, u.sigma = u.sigma,
+      u.missp = u.missp, cores = cores,
+      w.constr = list(name = method))
+    plot <- scplot(result = res.pi, plot.range = (1960:2003),
+            label.xy = list(x.lab = "Year",
+                y.lab = "GDP per capita (thousand US dollars)"),
+          x.ticks = c(1960, 1970, 1980, 1990, 2000, 2003), e.out = TRUE)
+    print(plot$plot_out + ggtitle(""))
+  }
```

Section 4.5 clarified the need for some additional sensitivity analysis when it comes to out-of-sample uncertainty quantification. Figure 3 shows the impact of shrinking and enlarging $\widehat{\sigma}_{\mathscr{H}}$ on the prediction intervals for $Y_{1t}(0)$, $t = 1997$, when we assume that e_t is sub-Gaussian



(a) lasso (b) ridge

(c) least squares (d) L1-L2

Figure 2: Uncertainty quantification for different forms of $\mathcal{W}$ using 90% prediction intervals, German reunification example.
Note: Dashed vertical line indicates the period when the treatment (German reunification) occurs.

conditional on $\mathscr{H}$. The black horizontal line shows the level of the outcome for the treated unit in 1997, $Y_{1t}(1)$ for $t = 1997$. The blue bars report 95% prediction intervals for $Y_{1t}(0)$, $t = 1997$, that only take into account in-sample uncertainty. The red dashed bars adds the out-of-sample uncertainty to obtain 90% prediction intervals. As shown in the figure, the predicted treatment effect $\widehat{\tau}_{1997}$ remains different from zero with high probability over $\mathscr{H}$ even doubling $\widehat{\sigma}_{\mathscr{H}}$.

Finally, the package offers the possibility to match the treated unit and the synthetic unit using multiple features and the possibility to compute simultaneous prediction intervals. If we want to match West Germany and the synthetic unit not only on GDP per capita but also on trade openness ($M = 2$), we can simply modify the object '`scpi_data`' as follows.

```
R> df <- scdata(df = data, id.var = id.var, time.var = time.var,
+    outcome.var = outcome.var, period.pre = period.pre,
+    period.post = period.post, unit.tr = unit.tr,
```

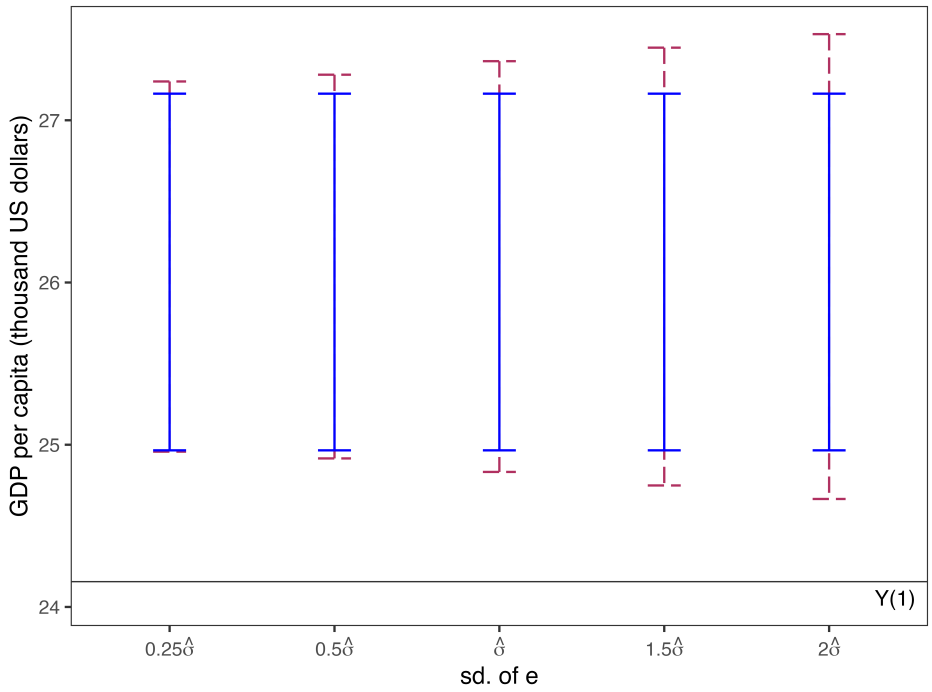


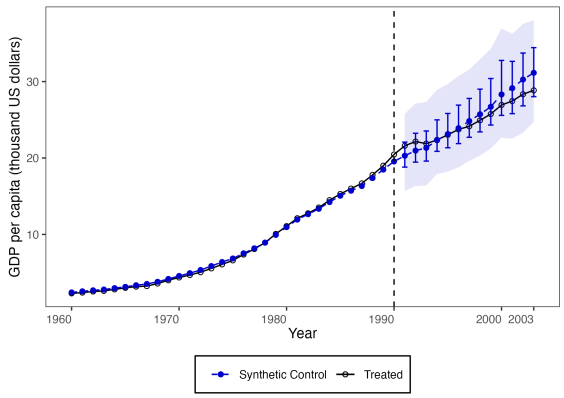
Figure 3: Sensitivity analysis on out-of-sample uncertainty with sub-Gaussian bounds, German reunification example.
Note: Dashed vertical line indicates the period when the treatment (German reunification) occurs. The black horizontal line shows the outcome for the treated unit in 1997, $Y_{1,1997}(1)$. *The blue bars report 95% prediction intervals for the counterfactual* $Y_{1,1997}(0)$ *only taking into account in-sample uncertainty. The red dashed bars adds out-of-sample uncertainty to obtain 90% prediction intervals.*

```
+     features = c("gdp", "trade"), cov.adj = list(c("constant")),
+     cointegrated.data = cointegrated.data, unit.co = unit.co)
```

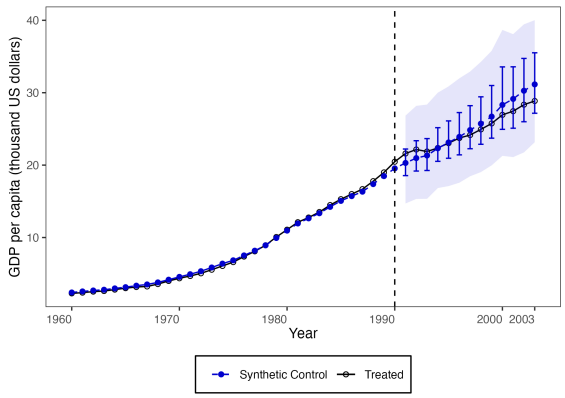
To get simultaneous prediction intervals it suffices to add the option `joint = TRUE` to the `scplot` command. Results are reported in Figure 4, where blue shaded areas depict 90% simultaneous prediction intervals for periods from 1991 to 2004. In panel (c), $Q = 0.903$, whereas in panel (e) $Q = 1, Q_2 = 0.903$.

6. Conclusion

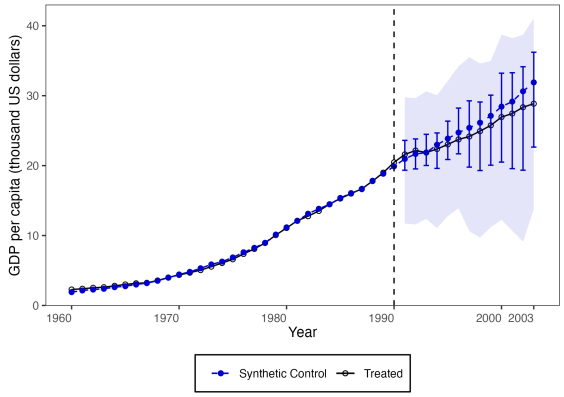
This article introduced the R software package **scpi**, which implements point estimation/prediction and inference/uncertainty quantification procedures for synthetic control methods. The package is also available in the Stata and Python statistical platforms, as described in the appendices. Further information can be found at https://nppackages.github.io/scpi/.



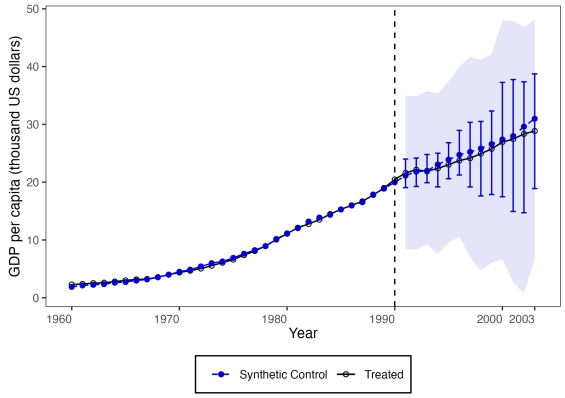
(a) simplex



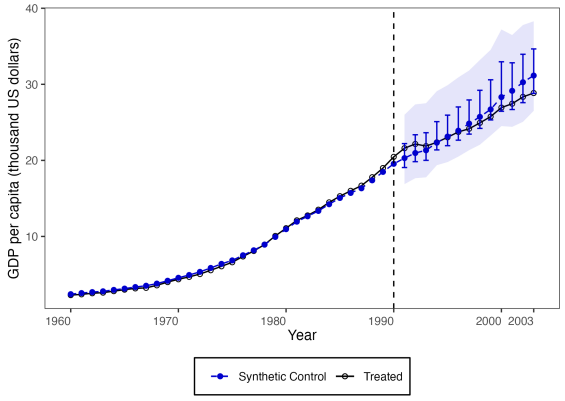
(b) lasso



(c) ridge



(d) least squares



(e) L1-L2

Figure 4: Uncertainty quantification for different forms of $\mathcal{W}$ using 90% prediction intervals, German reunification example.
Note: Dashed vertical line indicates the period when the treatment (German reunification) occurs. Blue shaded areas depict 90% simultaneous prediction intervals for periods 1991 through 2004. In panel (c), $Q = 0.903$, in panel (e) $Q = 1, Q_2 = 0.903$.

Acknowledgments

We thank Alberto Abadie and Bartolomeo Stellato for many insightful discussions. Cattaneo and Titiunik gratefully acknowledge financial support from the National Science Foundation (SES-2019432 and SES-2241575), Cattaneo gratefully acknowledges financial support from the National Institute of Health (R01 GM072611-16), and Feng gratefully acknowledges financial support from the National Natural Science Foundation of China (grants 72203122 and 72133002).

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. doi:10.1257/jel.20191450.

Abadie A, Cattaneo MD (2018). "Econometric Methods for Program Evaluation." *Annual Review of Economics*, **10**, 465–503. doi:10.1146/annurev-economics-080217-053402.

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

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

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

Abadie A, L'Hour J (2021). "A Penalized Synthetic Control Estimator for Disaggregated Data." *Journal of the American Statistical Association*, **116**(536), 1817–1834. doi:10.1080/01621459.2021.1971535.

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

Arkhangelsky D, Athey S, Hirshberg DA, Imbens GW, Wager S (2021). "Synthetic Difference in Differences." *American Economic Review*, **111**(12), 4088–4118. doi:10.1257/aer.20190159.

Arkhangelsky D, Athey S, Hirshberg DA, Imbens GW, Wager S (2024). ***synthdid**: Synthetic Difference in Differences Estimation*. R package version 0.0.9, URL synth-inference.github.io/synthdid.

Barrett P (2018). ***pgsc**: Computes Powell's Generalized Synthetic Control Estimator*. doi:10.32614/CRAN.package.pgsc. R package version 1.0.0.

Becker M, Klössner S (2024). ***MSCMT**: Multivariate Synthetic Control Method Using Time Series*. doi:10.32614/CRAN.package.MSCMT. R package version 1.4.0.

Ben-Michael E (2024). ***augsynth**: Augmented Synthetic Control Method*. R package version 0.2.0, URL https://github.com/ebenmichael/augsynth.

Ben-Michael E, Feller A, Rothstein J (2022). "Synthetic Controls With Staggered Adoption." *Journal of the Royal Statistical Society B*, **84**(2), 351–381. doi:10.1111/rssb.12448.

Boyd S, Vandenberghe L (2004). *Convex Optimization*. Cambridge university press.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2022). "Uncertainty Quantification in Synthetic Controls With Staggered Treatment Adoption." *arXiv 2210.05026*, arXiv.org E-Print Archive. doi:10.48550/arXiv.2210.05026.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2024a). ***scpi**: Prediction Intervals for Synthetic Control Methods with Multiple Treated Units and Staggered Adoption*. python package version 2.2.9, URL https://pypi.org/project/scpi-pkg/.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2024b). ***scpi**: Prediction Intervals for Synthetic Control Methods with Multiple Treated Units and Staggered Adoption*. doi:10.32614/CRAN.package.scpi. R package version 2.2.6, URL https://CRAN.R-project.org/package=scpi.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2024c). ***scpi**: Prediction Intervals for Synthetic Control Methods with Multiple Treated Units and Staggered Adoption*. Stata package version 20240824, URL https://github.com/nppackages/scpi.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. doi:10.1080/01621459.2021.1979561.

Cerulli G (2020). ***NPSYNTH**: Stata Module to Implement Nonparametric Synthetic Control Method*. URL https://ideas.repec.org/c/boc/bocode/s458398.html.

Chen Q (2023). ***SYNTH2**: Stata Module to Implement Synthetic Control Method (SCM) With Placebo Tests, Robustness Test and Visualization*. URL https://ideas.repec.org/c/boc/bocode/s459017.html.

Chernozhukov V, Wüthrich K, Zhu Y (2021). "An Exact and Robust Conformal Inference Method for Counterfactual and Synthetic Controls." *Journal of the American Statistical Association*, **116**(536), 1849–1864. doi:10.1080/01621459.2021.1920957.

Domahidi A, Chu E, Boyd S (2013). "ECOS: An SOCP Solver for Embedded Systems." In *2013 European Control Conference (ECC)*, pp. 3071–3076. IEEE.

Dunford E (2023). ***tidysynth**: A Tidy Implementation of the Synthetic Control Method*. doi:10.32614/CRAN.package.tidysynth. R package version 0.2.0.

Engelbrektson O (2021). ***SyntheticControlMethods**: A Python Package for Causal Inference Using Synthetic Controls*. Python package version 1.1.17, URL https://pypi.org/project/SyntheticControlMethods/.

Ferman B, Pinto C (2021). "Synthetic Controls With Imperfect Pretreatment Fit." *Quantitative Economics*, **12**(4), 1197–1221. doi:10.3982/qe1596.

Fonseca YR, Masini R, Medeiros MC, Vasconcelos GFR (2017). ***ArCo**: Artificial Counterfactual Package*. doi:10.32614/CRAN.package.ArCo. R package version 0.3-1.

Friedman J, Hastie T, Tibshirani R (2001). *The Elements of Statistical Learning*. Springer-Verlag. doi:10.1007/978-0-387-21606-5.

Fu A, Narasimhan B (2023). ***ECOSolveR**: Embedded Conic Solver in R*. doi:10.32614/CRAN.package.ECOSolveR. R package version 0.5.5.

Fu A, Narasimhan B, Boyd S (2020). "**CVXR**: An R Package for Disciplined Convex Optimization." *Journal of Statistical Software*, **94**(14), 1–34. doi:10.18637/jss.v094.i14.

Greathouse J (2022). ***scul**: Stata Module to Implement Regularized Synthetic Control (Using LASSO) Estimators for Single and Multiple-Treated Unit Settings*. URL https://ideas.repec.org/c/boc/bocode/s459107.html.

Hainmueller J, Diamond A (2023). ***Synth**: Synthetic Control Group Method for Comparative Case Studies*. doi:10.32614/CRAN.package.Synth. R package version 1.1-8.

Hoerl AE, Kannard RW, Baldwin KF (1975). "Ridge Regression: Some Simulations." *Communications in Statistics-Theory and Methods*, **4**(2), 105–123. doi:10.1080/03610927508827232.

Hollingsworth A (2024). ***scul**: Synthetic Control Using Lasso (SCUL)*. R package version 0.2.0.1, URL hollina.github.io/scul.

Hsiao C, Steve Ching H, Ki Wan S (2012). "A Panel Data Approach for Program Evaluation: Measuring the Benefits of Political and Economic Integration of Hong Kong With Mainland China." *Journal of Applied Econometrics*, **27**(5), 705–740. doi:10.1002/jae.1230.

Li KT (2020). "Statistical Inference for Average Treatment Effects Estimated by Synthetic Control Methods." *Journal of the American Statistical Association*, **115**(532), 2068–2083. doi:10.1080/01621459.2019.1686986.

MacKinnon JG (2013). "Thirty Years of Heteroskedasticity-Robust Inference." In X Chen, NR Swanson (eds.), *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis: Essays in Honor of Halbert L. White Jr*, pp. 437–461. Springer-Verlag. doi:10.1007/978-1-4614-1653-1_17.

Masini R, Medeiros MC (2021). "Counterfactual Analysis With Artificial Controls: Inference, High Dimensions and Nonstationarity." *Journal of the American Statistical Association*, **116**(536), 1773–1788. doi:10.1080/01621459.2021.1964978.

Muehlbach NS (2021). ***treebased_synthetic_controls***. Python package version 0.1.12, URL https://pypi.org/project/treebased-synthetic-controls.

Narasimhan B, Goulart P, Chen Y (2024). ***clarabel**: Interior Point Conic Optimization Solver*. doi:10.32614/CRAN.package.clarabel. R package version 0.9.0.1.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robbins M, Davenport S (2023). ***microsynth**: Synthetic Control Methods with Micro- And Meso-Level Data*. doi:10.32614/CRAN.package.microsynth. R package version 2.0.44.

Robbins MW, Davenport S (2021). "**Microsynth**: Synthetic Control Methods for Disaggregated and Micro-Level Data in R." *Journal of Statistical Software*, **97**, 1–31. doi:10.18637/jss.v097.i02.

Shaikh AM, Toulis P (2021). "Randomization Tests in Observational Studies With Staggered Adoption of Treatment." *Journal of the American Statistical Association*, **116**(536), 1835–1848. doi:10.1080/01621459.2021.1974458.

Silva BC, DeWitt M (2024). ***SCtools**: Extensions for Synthetic Controls Analysis*. doi:10.32614/CRAN.package.SCtools. R package version 0.3.3.

StataCorp (2019). *Stata Statistical Software: Release 16*. StataCorp LLC, College Station, TX. URL https://www.stata.com/.

van Kesteren EJ, Slaughter I (2024). ***pensynth**: Penalized Synthetic Control Estimation*. doi:10.32614/CRAN.package.pensynth. R package version 0.5.1.

van Rossum G, *et al.* (2011). *Python Programming Language*. URL http://www.python.org.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag. doi:10.1007/978-0-387-98141-3.

Wiltshire JC (2024). ***ALLSYNTH**: Stata Module to Automate Estimation of (i) Bias-Corrected Synthetic Control Gaps ("Treatment Effects")*. URL https://ideas.repec.org/c/boc/bocode/s459076.html.

Wuthrich K (2021). ***scinference**: Inference for Synthetic Controls*. R package version 0.0.0.9000, URL https://github.com/kwuthrich/scinference.

Xu Y, Liu L (2021). ***gsynth**: Generalized Synthetic Control Method*. doi:10.32614/CRAN.package.gsynth. R package version 1.2.1.

Ye J (1998). "On Measuring and Correcting the Effects of Data Mining and Model Selection." *Journal of the American Statistical Association*, **93**(441), 120–131. doi:10.1080/01621459.1998.10474094.

Ypma J, Johnson SG (2024). ***nloptr**: R Interface to NLopt*. doi:10.32614/CRAN.package.nloptr. R package version 2.1.1.

Zou H, Hastie T, Tibshirani R (2007). "On the "Degrees of Freedom" of the Lasso." *Annals of Statistics*, **35**(5), 2173–2192. doi:10.1214/009053607000000127.

A. Python illustration

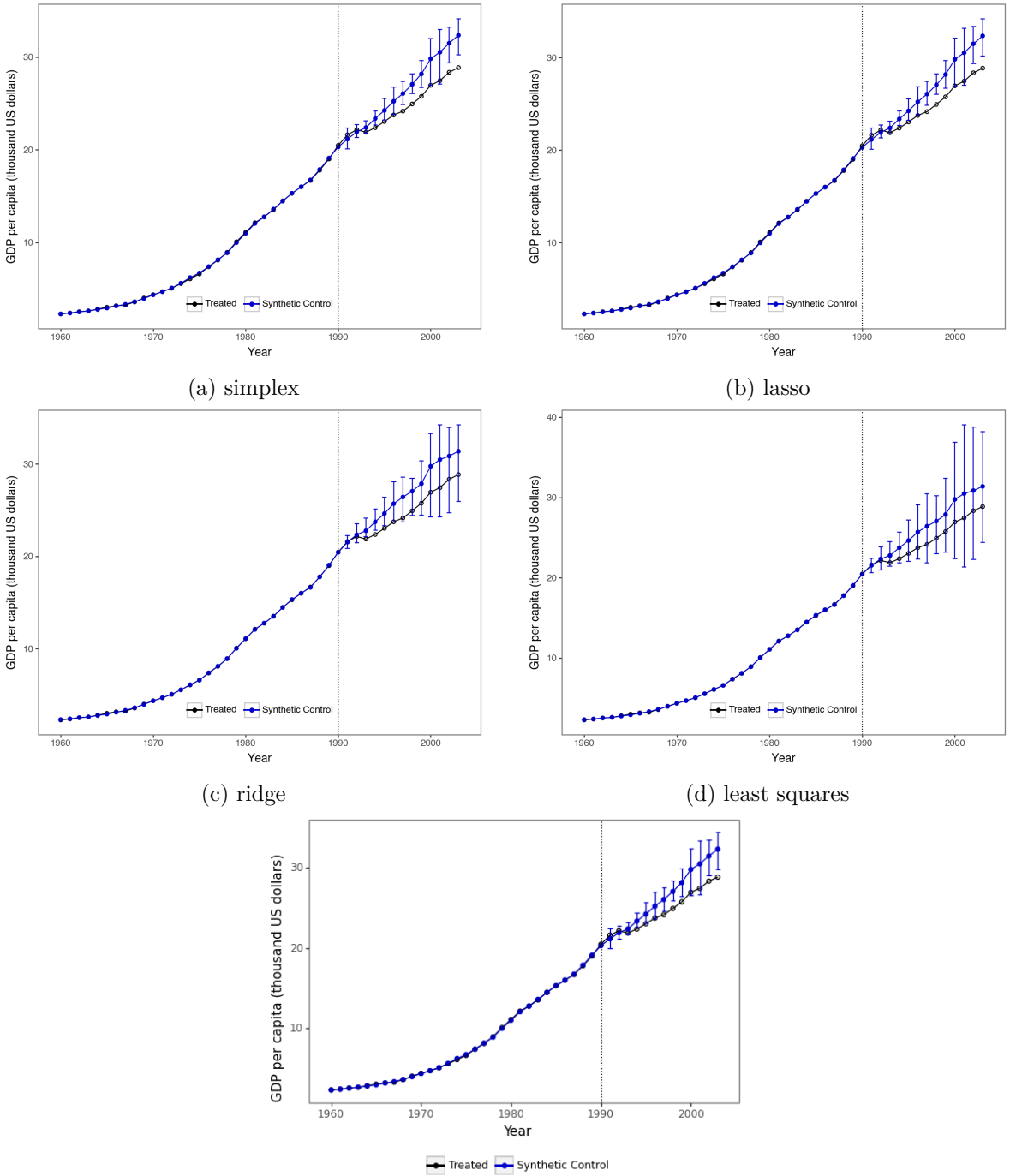
This appendix section shows how to conduct the same analysis carried out in Section 5 using the companion Python package. Figure 5 shows the main results. Replication files and data are available at https://nppackages.github.io/scpi/.

```
>>> import pandas
>>> import numpy
>>> import random
>>> import os
>>> from warnings import filterwarnings
>>> from plotnine import ggtitle, ggsave

>>> from scpi_pkg.scdata import scdata
>>> from scpi_pkg.scdataMulti import scdataMulti
>>> from scpi_pkg.scest import scest
>>> from scpi_pkg.scpi import scpi
>>> from scpi_pkg.scplot import scplot
>>> from scpi_pkg.scplotMulti import scplotMulti

>>> filterwarnings("ignore")

>>> data = pandas.read_csv("scpi_germany.csv")

>>> id_var = "country"
>>> outcome_var = "gdp"
>>> time_var = "year"
>>> period_pre = numpy.arange(1960, 1991)
>>> period_post = numpy.arange(1991, 2004)
>>> unit_tr = "West Germany"
>>> unit_co = list(set(data[id_var].to_list()))
>>> unit_co = [cou for cou in unit_co if cou != "West Germany"]
>>> constant = True
>>> cointegrated_data = True

>>> data_prep = scdata(df = data, id_var = id_var, time_var = time_var,
...  outcome_var = outcome_var, period_pre = period_pre,
...  period_post = period_post, unit_tr = unit_tr,
...  unit_co = unit_co, cointegrated_data = cointegrated_data,
...  constant = constant)

>>> est_si = scest(data_prep, w_constr = {"name": "simplex"})
>>> print(est_si)

>>> w_constr = {"name": "simplex", "Q": 1}
>>> u_missp = True
>>> u_sigma = "HC1"
```

```
>>> u_order = 1
>>> u_lags = 0
>>> e_method = "gaussian"
>>> e_order = 1
>>> e_lags = 0
>>> sims = 1000
>>> cores = 1

>>> for mtd in ["simplex", "lasso", "ridge", "L1-L2", "ols"]:
...  if mtd in ["ridge", "L1-L2"]:
...    lgapp = "generalized"
...  else:
...    lgapp = "linear"
...  numpy.random.seed(8894)
...  pi_si = scpi(data_prep, sims = sims, w_constr = {"name": mtd},
...    u_order = u_order, u_lags = u_lags, e_order = e_order,
...    e_lags = e_lags, e_method = e_method, u_missp = u_missp,
...    lgapp = lgapp, u_sigma = u_sigma, cores = cores)
...  plot = scplot(pi_si, x_lab = "Year", e_method = e_method,
...    y_lab = "GDP per capita (thousand US dollars)")
...  plot + ggtitle("")

>>> data = pandas.read_csv("scpi_germany.csv")

>>> id_var = "country"
>>> outcome_var = "gdp"
>>> time_var = "year"
>>> period_pre = numpy.arange(1960, 1991)
>>> period_post = numpy.arange(1991, 2004)
>>> unit_tr = "West Germany"
>>> unit_co = list(set(data[id_var].to_list()))
>>> unit_co = [cou for cou in unit_co if cou != "West Germany"]
>>> constant = False
>>> cointegrated_data = True
>>> cov_adj = [["constant"], ["constant"]]

>>> data_prep = scdata(df = data, id_var = id_var, time_var = time_var,
...  outcome_var = outcome_var, period_pre = period_pre,
...  period_post = period_post, unit_tr = unit_tr,
...  unit_co = unit_co, cointegrated_data = cointegrated_data,
...  constant = constant, features = ["gdp", "trade"],
...  cov_adj = cov_adj)

>>> est_si = scest(data_prep, w_constr = {"name": "simplex"})
>>> print(est_si)

>>> w_constr = {"name": "simplex", "Q": 1}
```

```
>>> u_missp = True
>>> u_sigma = "HC1"
>>> u_order = 1
>>> u_lags = 0
>>> e_method = "gaussian"
>>> e_order = 1
>>> e_lags = 0
>>> sims = 1000
>>> cores = 1

>>> for mtd in ["simplex", "lasso", "ridge", "L1-L2", "ols"]:
...  if mtd in ["ridge", "L1-L2"]:
...    lgapp = "generalized"
...  else:
...    lgapp = "linear"
...  numpy.random.seed(8894)
...  pi_si = scpi(data_prep, sims = sims, w_constr = {"name": mtd},
...    u_order = u_order, u_lags = u_lags, e_order = e_order,
...    e_lags = e_lags, e_method = e_method, u_missp = u_missp,
...    lgapp = lgapp, u_sigma = u_sigma, cores = cores)
...  plot = scplot(pi_si, x_lab = "Year", e_method = e_method,
...    y_lab = "GDP per capita (thousand US dollars)")
...  plot + ggtitle("")
```

Case I: $M = 1$



(a) simplex

(b) lasso

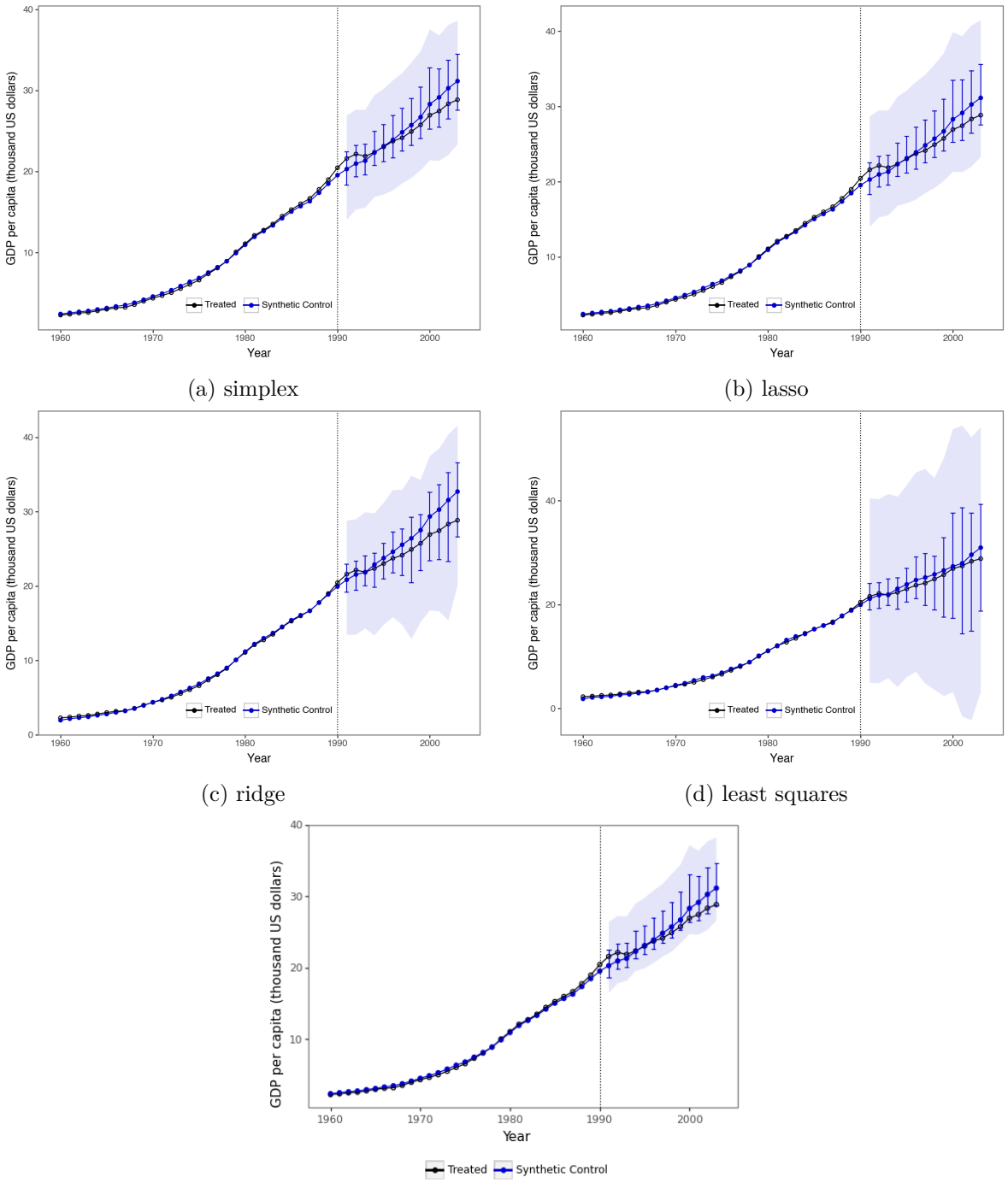
(c) ridge

(d) least squares

(e) L1-L2

Figure 5: Uncertainty quantification for different forms of $\mathcal{W}$ using 90% prediction intervals, German reunification example.

Case II: $M = 2$



(a) simplex

(b) lasso

(c) ridge

(d) least squares

(e) L1-L2

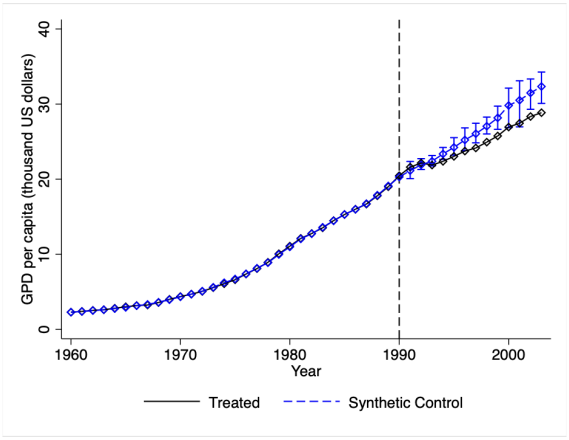
Figure 6: Uncertainty quantification for different forms of $\mathcal{W}$ using 90% prediction intervals, German reunification example.

B. Stata illustration

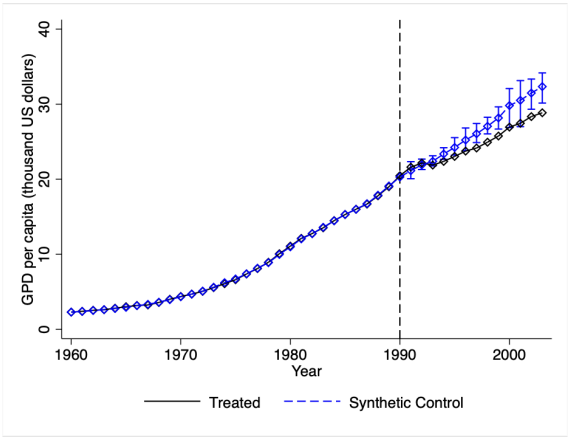
This appendix section replicates the analysis conducted in Section 5 for $M = 1$ using the companion Stata package. Main results are shown in Figure 7. Replication files and data are available at https://nppackages.github.io/scpi/.

```
. use "scpi_germany.dta", clear
. scdata gdp, dfname("python_scdata") id(country) outcome(gdp) ///
>        time(year) treatment(status) cointegrated constant
. local lgapp "linear"
. foreach method in "simplex" "lasso" "ols" "ridge" "L1-L2" {
.         if inlist("`method'", "ridge", "L1-L2") {
.                 local lgapp "generalized"
.         }
.         scpi, dfname("python_scdata") name(`method') ///
>           u_missp lgapp("`lgapp'") sims(1000) e_method(gaussian) ///
>          set_seed(8894)
.         scplot, uncertainty("gaussian") gphoptions(note("") ///
>           xtitle("Year") ///
>           ytitle("GPD per capita (thousand US dollars)"))
.         graph export "stata_germany_unc_`method'.png", replace
. }

. scdata gdp trade, dfname("python_scdata") id(country) outcome(gdp) ///
>        time(year) treatment(status) cointegrated  covadj("constant")
. local lgapp "linear"
. foreach method in "simplex" "lasso" "ols" "ridge" "L1-L2" {
.         if inlist("`method'", "ridge", "L1-L2") {
.                 local lgapp "generalized"
.         }
.         scpi, dfname("python_scdata") name(`method')  ///
>          e_method(gaussian) u_missp lgapp("`lgapp'") sims(1000) ///
>            set_seed(8894)
.         scplot, uncertainty("gaussian") gphoptions(note("") ///
>          xtitle("Year") ///
>          ytitle("GPD per capita (thousand US dollars)")) joint
.         graph export "stata_germany_unc_`method'_multi.png", replace
. }
```

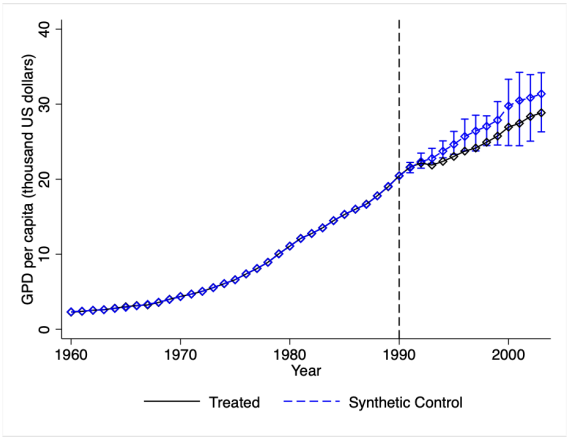
Case I: $M = 1$



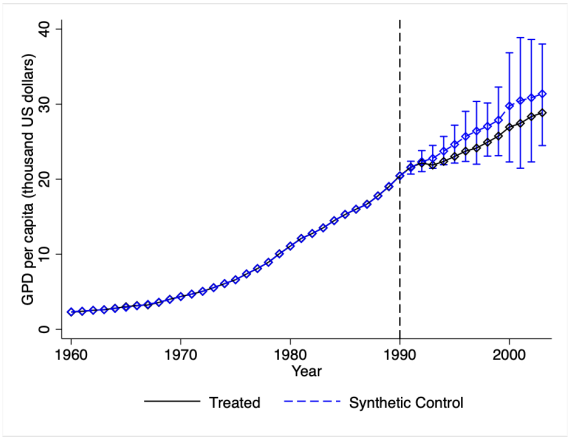
(a) simplex



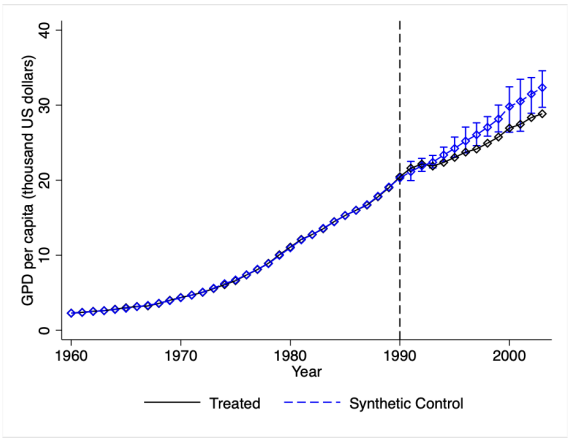
(b) lasso



(c) ridge



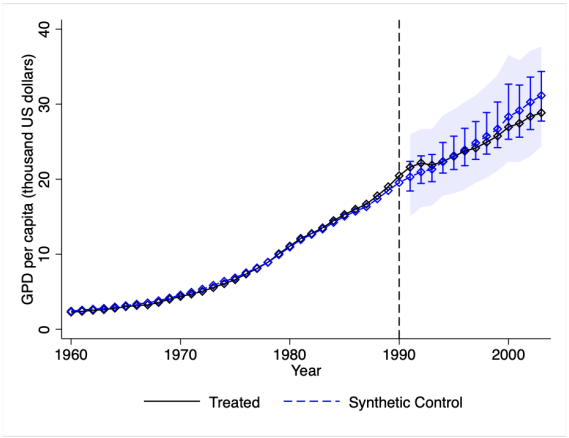
(d) least squares



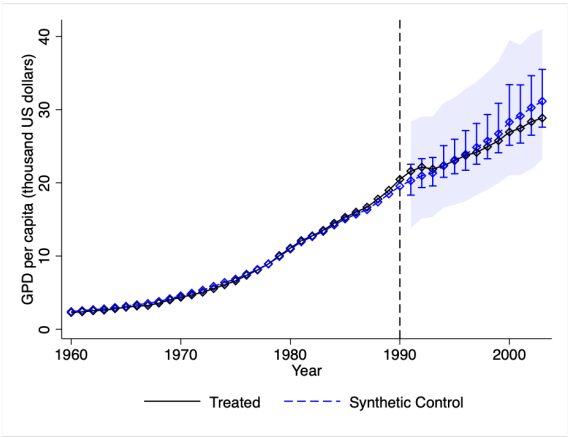
(e) L1-L2

Figure 7: Uncertainty quantification for different forms of $\mathcal{W}$ using 90% prediction intervals, German reunification example.

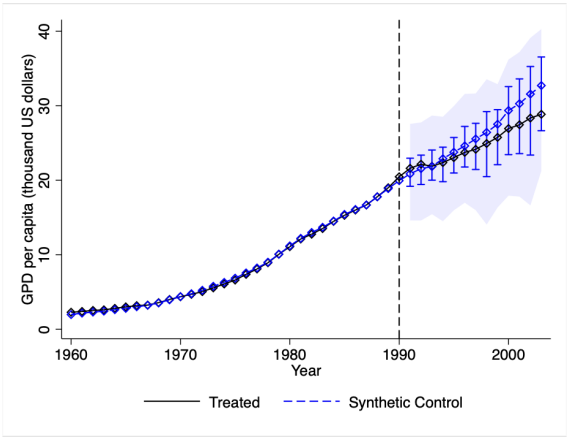
Case II: $M = 2$



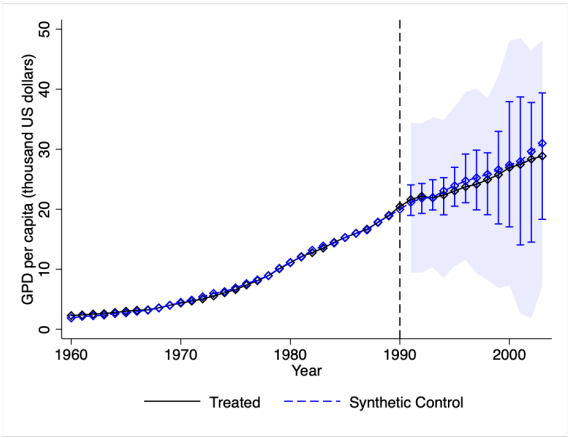
(a) simplex



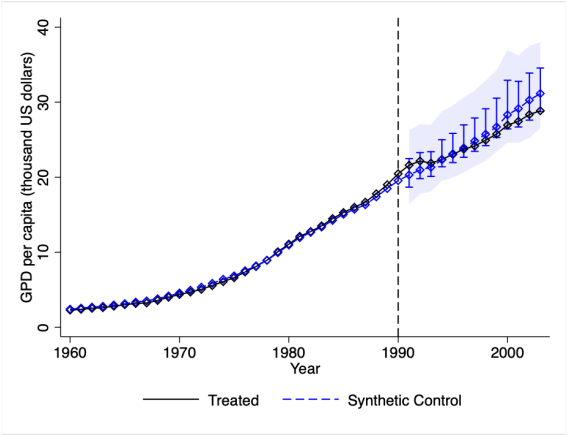
(b) lasso



(c) ridge



(d) least squares



(e) L1-L2

Figure 8: Uncertainty quantification for different forms of $\mathcal{W}$ using 90% prediction intervals, German reunification example.

Affiliation:

Matias D. Cattaneo
Department of Operations Research and Financial Engineering
Princeton University
227 Sherrerd Hall
Princeton, New Jersey 08544, United States of America
E-mail: cattaneo@princeton.edu
URL: https://cattaneo.princeton.edu/

Yingjie Feng
School of Economics and Management
Tsinghua University
B628 Lihua Building
Beijing 100084, China
E-mail: fengyj@sem.tsinghua.edu.cn
URL: https://sites.google.com/site/yingjieum/

Filippo Palomba
Department of Economics
Princeton University
20 Washington Rd
Princeton, New Jersey 08544, United States of America
E-mail: fpalomba@princeton.edu

Rocío Titiunik
Department of Politics
Princeton University
001 Fisher Hall
Princeton, New Jersey 08544, United States of America
E-mail: titiunik@princeton.edu
URL: https://scholar.princeton.edu/titiunik

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd