

scpi: Uncertainty Quantification for Synthetic Control Estimators

Matias D. Cattaneo* Yingjie Feng[†] Filippo Palomba[‡] Rocio Titiunik[§]

February 11, 2022

Abstract

The synthetic control method offers a way to estimate the effect of an aggregate 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 estimation 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, Palomba, Feng and Titiunik \(2022\)](#). The discussion contains numerical illustrations and a comparison with other synthetic control software.

Keywords: program evaluation, causal inference, synthetic controls, prediction intervals, non-asymptotic inference.

*Department of Operations Research and Financial Engineering, Princeton University.

[†]School of Economics and Management, Tsinghua University.

[‡]Department of Economics, Princeton University.

[§]Department of Politics, Princeton University.

Contents

1 Introduction 1

2 Setup 4

2.1 Extensions 6

3 Synthetic Control Prediction 7

3.1 Implementation 9

4 Uncertainty Quantification 13

4.1 In-Sample Error 14

4.2 Out-of-Sample Error 16

4.3 Implementation 17

4.4 Sensitivity Analysis 22

5 Empirical Illustration 23

6 Conclusion 29

7 Acknowledgments 29

A Appendix: Python Illustration 33

B Appendix: Stata Illustration 36

1 Introduction

The synthetic control method was introduced by [Abadie and Gardeazabal \(2003\)](#), and since then it has become widely used for program evaluation and causal inference in observational studies. It offers a way to estimate 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, this methodology has been extended in recent years to a variety of other settings with longitudinal data. See [Abadie \(2021\)](#) for a recent review on synthetic control methods, and [Abadie and Cattaneo \(2018\)](#) for a recent 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 point estimation or prediction. Examples of the former include disaggregated data settings ([Abadie and L'Hour, 2021](#)) or 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 estimation 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 was deployed.

Compared to estimation or prediction, 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, Palomba, Feng 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 estimation and inference using synthetic control methods, 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 available in the literature (Fu, Narasimhan and Boyd, 2020; Johnson, 2022). For uncertainty quantification, the package focuses on the prediction interval methods introduced by Cattaneo, Feng and Titiunik (2021) and Cattaneo, Palomba, Feng and Titiunik (2022). 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 four functions:

- `sdata()`. This function takes as input a `DataFrame` object and processes it to prepare the data matrices used for point estimation/prediction and inference/uncertainty quantification. The function allows the user to specify multiple features of the treated unit(s) to be matched by the synthetic unit 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 takes as input an “`scpi_data`” object produced with `sdata()`, and then implements a class of synthetic control point estimators/predictions for treatment effect estimation. The implementation allows for multiple features, with and without additional covariate adjustment, and for both stationary and non-stationary data. The allowed estimation procedures include unconstrained weighted least squares as well as constrained weighted least squares with simplex, lasso-type and ridge-type parameter space restrictions (see Table 2 below).
- `scpi()`. This function takes as input an “`scpi_data`” object produced with `sdata()`, and then implements prediction intervals accompanying a class of synthetic control point estimators/pre-

dictions for treatment effect estimation. It relies on `scest()` for point estimation/prediction of treatment effects, and thus inherits the same functionalities of that function. `scpi()` is designed to be the main function in applications, offering both point estimators/predictions for treatment effects as well as inference/uncertainty quantification (i.e., prediction intervals) for synthetic control methods. The function also allows the user to model separately in-sample and out-of-sample uncertainty, offering a broad range of options for practice.

- `scplot()`. This function processes objects whose class is either “`scest`” or “`scpi`”. These objects contain the results of the point estimation/prediction or uncertainty quantification methods, respectively. The command builds on the `ggplot2` package in R to compare the outcome time series of the treated unit(s) with the outcome time series of the synthetic control unit, along with the associated uncertainty. The function returns 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 `sdata()`, and then produce point estimators/predictions for treatment effects with uncertainty quantification using the function `scpi()`. The function `scest()` is useful in cases where only point estimators/predictions are of interest. Numerical illustrations are given in Section 5.

Table 1: Comparison of different R packages available on CRAN or on GitHub.

Package Name	Estimation Method	Inference Method	Multiple Treated	Staggered Adoption	Misspecification Robust	Automatic Parallelization	Last Update
Synth	SC	Perm	✗	✗	✗	✗	2014-01-27
ArCo	LA	Asym	✗	✗	✓	✗	2017-11-05
pgsc	SC	Perm	✓	✗	✗	✗	2018-10-28
MSCMT	SC	Perm	✗	✗	✗	✓	2019-11-14
SCtools	SC	Perm	✓	✗	✗	✓	2020-08-26
tidysynth	SC	Perm	✗	✗	✗	✗	2021-01-27
microsynth	CA	Perm	✓	✗	✗	✓	2021-02-26
scinference	SC, LA	Perm	✗	✗	✓	✗	2021-05-13
SCUL	LA	Perm	✗	✗	✗	✗	2021-05-18
SynthCast	SC	Perm	✗	✗	✗	✗	2021-06-14
gsynth	FA	Asym	✓	✓	✗	✓	2021-08-06
scpi	SC, LA, RI, LS, +	PI, Perm	✓	✓	✓	✓	2022-02-11

Note: CA = calibration (see [Robbins et al., 2017](#)); FA = factor-augmented model (see [Bai, 2009](#)); LA = Lasso penalty; RI = Ridge penalty; LS = unconstrained least squares; SC = standard synthetic control (see [Abadie et al., 2010](#)); + = user-specified options (see Table 3 below for more details); Perm = permutation-based inference; Asym = asymptotic-based inference; PI = prediction intervals (finite-sample probability guarantees).

There are many R packages available for estimation and inference using synthetic control methods;

Table 1 compares them to the package `scpi`. As 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 estimators. The package is also one of the first to handle multiple treatment units and staggered treatment adoption, offering a wider array of options in terms of estimators and inference methods when compared with the other few packages currently available. In addition, the package includes misspecification-robust methods, employs the latest optimization packages available, 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.

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

2 Setup

We first consider the standard synthetic control framework with a single treated unit. The researcher observes $N + 1$ units for $T_0 + T_1$ periods of time. Units are indexed by $i = 1, 2, \dots, N, N + 1$, 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

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

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:

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

As in the classical causal inference framework, 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 (see Abadie, 2021, for references).

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 $Y_{1t}(0)$ of the treated after the interventions. The idea of the synthetic control method is to find a vector of weights $\mathbf{w} = (w_2, w_3, \dots, w_{N+1})'$ such that a given loss function is minimized under constraints, only using pre-intervention observations. Given the resulting set of estimated weights $\hat{\mathbf{w}}$, the treated unit’s counterfactual (potential) outcome is calculated as $\hat{Y}_{1t}(0) = \sum_{i=2}^{N+1} \hat{w}_i Y_{it}(0)$ for $t > T_0$. The weighted average $\hat{Y}_{1t}(0)$ 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 proposed by Cattaneo, Feng and Titiunik (2021) and Cattaneo, Palomba, Feng and Titiunik (2022) to complement those estimates.

2.1 Extensions

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

- **Average across multiple treated units.** The single treated unit in the canonical framework above can also be understood as an aggregate of multiple treated units. For instance, assume that there are N_1 treated units, and their treated and untreated potential outcomes are respectively denoted by $Y_{1t}^j(1)$ and $Y_{1t}^j(0)$ for $j = 1, \dots, N_1$. The observed outcome of the j th treated unit is given by $Y_{1t}^j := \mathbb{1}(t \leq T_0)Y_{1t}^j(0) + \mathbb{1}(t > T_0)Y_{1t}^j(1)$. A researcher might be interested in the *average* treatment effect on the treated, i.e.,

$$\tau_t := \frac{1}{N_1} \sum_{j=1}^{N_1} \left(Y_{1t}^j(1) - Y_{1t}^j(0) \right), \quad t > T_0.$$

The methods above can be implemented by simply defining an aggregate “unit 1” whose observed outcome is $Y_{1t} := \frac{1}{N_1} \sum_{j=1}^{N_1} Y_{1t}^j$, for $t = 1, \dots, T_0 + T_1$. Other features of “unit 1” used in \mathbf{A} can be defined similarly as averages of the corresponding features across multiple treated units.

- **Average across 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, that is,

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

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

period $t > T_0$, the estimated average counterfactual outcome of the treated is given by

$$\sum_{i=2}^{N+1} \hat{w}_i \left(\frac{1}{T_1} \sum_{t=T_0+1}^{T_0+T_1} Y_{it}(0) \right).$$

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.

- **Staggered treatment adoption.** Our framework can also be extended to the scenario where multiple treated units adopt the treatment at different times, known in the literature as a *staggered adoption* design. In such context, one can understand the adoption time as a multivalued treatment, and a large class of causal estimands can be defined accordingly. For example, let $T_i \in \{T_0 + 1, T_0 + 2, \dots, T\}$ denote the adoption time of unit i , 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., $Y_{i(T_i+1)}(1) - Y_{i(T_i+1)}(0)$, is of interest. One can take all units that are treated after $T_i + 1$ to obtain the estimated synthetic control weights and construct the synthetic control prediction of the counterfactual outcome $Y_{i(T_i+1)}(0)$ 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.

3 Synthetic Control Prediction

We consider synthetic control weights constructed simultaneously for M features of the treated unit, denoted by $\mathbf{A}_l = (a_{1,l}, \dots, a_{T_0,l})' \in \mathbb{R}^{T_0}$, with index $l = 1, \dots, M$. 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 $\mathbf{B}_l = (\mathbf{B}_{1,l}, \mathbf{B}_{2,l}, \dots, \mathbf{B}_{J,l}) \in \mathbb{R}^{T_0 \times J}$ and $\mathbf{C}_l = (\mathbf{C}_{1,l}, \dots, \mathbf{C}_{K,l}) \in \mathbb{R}^{T_0 \times K}$, respectively. More precisely, for each j , $\mathbf{B}_{j,l} = (b_{j1,l}, \dots, b_{jT_0,l})'$

corresponds to the l th feature of the j th unit observed in T_0 pre-treatment periods and, for each k , $\mathbf{C}_{k,l} = (c_{k1,l}, \dots, c_{kT_0,l})'$ 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:

$$\hat{\boldsymbol{\beta}} := (\hat{\mathbf{w}}', \hat{\mathbf{r}}')' \in \arg \min_{\mathbf{w} \in \mathcal{W}, \mathbf{r} \in \mathcal{R}} (\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r})' \mathbf{V} (\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r}) \quad (3.1)$$

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 $T_0 M \times T_0 M$ weighting matrix reflecting the relative importance of different equations and time periods.

From (3.1), we can define the pseudo-true residual \mathbf{u} :

$$\mathbf{u} = \mathbf{A} - \mathbf{B}\mathbf{w}_0 - \mathbf{C}\mathbf{r}_0, \quad (3.2)$$

where \mathbf{w}_0 and \mathbf{r}_0 denote the mean squared error estimands associated with $\hat{\mathbf{w}}$ and $\hat{\mathbf{r}}$. As discussed in the next section, the proposed prediction intervals are valid conditional on some information set \mathcal{H} . Thus, \mathbf{w}_0 and \mathbf{r}_0 above are viewed as the (possibly constrained) best linear prediction coefficients conditional on \mathcal{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 $\hat{\mathbf{w}}$, as this is the most likely scenario in practice.

Given the estimated weights $\hat{\mathbf{w}}$ and coefficients $\hat{\mathbf{r}}$, the post-treatment counterfactual outcome

$Y_{1T}(0)$ for the treated unit is predicted by

$$\hat{Y}_{1T}(0) = \mathbf{x}'_T \hat{\mathbf{w}} + \mathbf{g}'_T \hat{\mathbf{r}} = \mathbf{p}'_T \hat{\boldsymbol{\beta}}, \quad \mathbf{p}_T := (\mathbf{x}'_T, \mathbf{g}'_T)', \quad T > T_0, \quad (3.3)$$

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 $\mathbf{x}_T = (Y_{2T}(0), \dots, Y_{(N+1)T}(0))'$ and \mathbf{g}_T is excluded when \mathbf{C} is not specified.

The next section discusses implementation details leading to $\hat{Y}_{1T}(0)$, 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 `sdata()` in `scpi` prepares the data to be used. 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 $\mathbf{P} = (\mathbf{p}_{T_0+1}, \dots, \mathbf{p}_{T_0+T_1})'$. The user must provide the variable containing the identifier of the units (`id.var`), the time variable (`time.var`), the outcome variable (`outcome.var`), the features to be matched (`features`), the treated unit (`unit.tr`), the control units (`unit.co`), the pre-treatment period (`period.pre`), and the post-treatment period (`period.post`). These options completely specify \mathbf{A}, \mathbf{B} , and \mathbf{P} . The user can also control the form of \mathcal{R} in (3.1) 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 $M \cdot T_0$ in \mathbf{C} . Note that 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 just one feature ($M = 1$), then `cov.adj` can be an unnamed list:

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

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

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

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

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

This specification creates a block diagonal \mathbf{C} where the blocks are all identical. Note that, 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 $2 \cdot T_0$.

Finally, if \mathbf{A} and \mathbf{B} form a cointegrated system, it is possible to set the option `cointegrated.data <- TRUE` that will eventually be passed to the function `scpi()` to properly handle in-sample and out-of-sample uncertainty quantification (see sections 4.3 and 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 (3.1) 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:

$$\mathcal{W} = \{\mathbf{w} \in \mathbb{W} : \|\mathbf{w}\|_p \leq Q\}, \quad \mathbb{W} \in \{\mathbb{R}^N, \mathbb{R}_+^N\}, \quad p \in \{1, 2\}, \quad Q \in \mathbb{R}_{++}$$

where the inequality constraint on the norm can be made an equality constraint. The user can specify all these elements through a list to be passed to the option `w.constr`:

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

The three lines above create $\mathcal{W}_1 = \mathbb{R}^N$, $\mathcal{W}_2 = \{\mathbf{w} \in \mathbb{R}^N : \|\mathbf{w}\|_1 = 1, w_j \geq 0, j = 1, \dots, N\}$, and $\mathcal{W}_3 = \{\mathbf{w} \in \mathbb{R}^N : \|\mathbf{w}\|_2 \leq 1\}$, respectively. The option `p` specifies the norm of the weights to be constrained, `dir` is the direction of the constraint on the norm of the weights, `Q` is the size of the constraint, and `code` sets the (common) lower bound on \mathbf{w} . We allow `p` picks the norm to be constrained (if any) and it can be either ‘no norm’, ‘L1’, or ‘L2’, whereas `dir` can be either ‘==’ or ‘<=’, and `lb` is either 0 or `-Inf`, thus it chooses the form of \mathbb{W} .

Popular constraints can be called explicitly using the option `name` in the list. Table 2 gives

prototypical examples of such constraints.

Table 2: *Constraints on the weights that can be directly called.*

Name	w.constr	\mathcal{W}
OLS	<code>list(name = 'ols')</code>	\mathbb{R}^N
simplex	<code>list(name = 'simplex', Q = Q)</code>	$\{\mathbf{w} \in \mathbb{R}_+^N : \ \mathbf{w}\ _1 = Q\}$
lasso	<code>list(name = 'lasso', Q = Q)</code>	$\{\mathbf{w} \in \mathbb{R}_+^N : \ \mathbf{w}\ _1 \leq Q\}$
ridge	<code>list(name = 'ridge', Q = Q)</code>	$\{\mathbf{w} \in \mathbb{R}^N : \ \mathbf{w}\ _2 \leq Q\}$

In particular, specifying `list(name = 'simplex', Q = 1)` gives the standard constraints used in the canonical synthetic control method (Abadie, 2021), that is, computing weights in (3.1) 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 four constraints can be called automatically through the option `name` and manually through the options `p`, `Q`, `lb`, and `dir`. In the snippet, Q is set to one for Ridge for simplicity, but to replicate the results obtained with the option `name` one should input the proper Q according to the rule of thumb described further below.

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

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

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

## Ridge
w.constr <- list(name = "ridge")
w.constr <- list(p = "L2", lb = -Inf, Q = 1, dir = "<=")
```

Using the option `w.constr` in `scest()` (or `scpi()`) and the options `cov.adj` and `constant` in `sccdata()` appropriately, i.e., setting \mathcal{W} and \mathcal{R} in (3.1), many synthetic control estimators proposed in the literature can be implemented. Table 3 provides a non-exhaustive list of examples.

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

Article	\mathcal{W}	\mathcal{R}	w.constr name	Q	constant
Hsiao et al. (2012)	\mathbb{R}^N	\mathbb{R}	"ols"	NULL	T
Abadie et al. (2010)	$\{\mathbf{w} \in \mathbb{R}_+^N : \ \mathbf{w}\ _1 = 1\}$	$\{0\}$	"simplex"	1	F
Fernan and Pinto (2021)	$\{\mathbf{w} \in \mathbb{R}_+^N : \ \mathbf{w}\ _1 = 1\}$	\mathbb{R}	"simplex"	1	T
Chernozhukov et al. (2021)	$\{\mathbf{w} \in \mathbb{R}_+^N : \ \mathbf{w}\ _1 \leq 1\}$	\mathbb{R}	"lasso"	1	T
Amjad et al. (2018)	$\{\mathbf{w} \in \mathbb{R}^N : \ \mathbf{w}\ _2 \leq Q\}$	$\{0\}$	"ridge"	Q	F

Tuning parameter choices

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

- **Lasso** ($p = 1$). Being Lasso 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 estimation 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

$$\hat{\mathbf{w}} := \arg \min_{\mathbf{w} \in \mathbb{R}^N} (\mathbf{A} - \mathbf{B}\mathbf{w})' \mathbf{V} (\mathbf{A} - \mathbf{B}\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2,$$

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

$$\hat{\mathbf{w}} := \arg \min_{\mathbf{w} \in \mathbb{R}^N, \|\mathbf{w}\|_2^2 \leq Q^2} (\mathbf{A} - \mathbf{B}\mathbf{w})' \mathbf{V} (\mathbf{A} - \mathbf{B}\mathbf{w}),$$

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

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

where $\hat{\sigma}_u^2$ and $\hat{\mathbf{w}}_{\text{OLS}}$ are estimators of the variance of the pseudo-true residual \mathbf{u} and the coefficients \mathbf{w}_0 based on least squares regression. 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{Z} are orthonormal, the closed-form solution for the Ridge estimator is $\hat{\mathbf{w}} = (\mathbf{I} + \lambda \mathbf{I})^{-1} \hat{\mathbf{w}}_{\text{OLS}}$, and

it follows that if the constraint on the ℓ^2 -norm is binding, then $Q = \|\hat{\mathbf{w}}\|_2 = \|\hat{\mathbf{w}}_{\text{OLS}}\|_2/(1 + \lambda)$. If more than one feature is specified ($M > 1$), our suggestion is to compute a shrinkage parameter Q_ℓ for each feature $\ell = 1, \dots, M$ and then select $Q := \max_{\ell=1, \dots, M} Q_\ell$. A simple way to do that would be to call `scest()` separately for each feature ℓ , store the corresponding value of Q_ℓ , and set $Q = \max_{\ell=1, \dots, M} Q_\ell$.

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.* In this case we compute $\hat{\mathbf{w}}$ 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 i th donor has a missing entry in one of the M features in the post-treatment period \tilde{T} . It implies that $\mathbf{p}_{\tilde{T}}$ has a missing entry, and thus the synthetic unit and the associated prediction intervals are not available.
- *Missing post-treatment treated data.* Data for the treated unit after the treatment is only used to quantify the treatment effect τ_T , therefore confidence intervals for the synthetic point estimate can be computed in the usual way.

4 Uncertainty Quantification

Following Cattaneo, Feng and Titiunik (2021) and Cattaneo, Palomba, Feng 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”. Consequently, we prefer to call $\hat{\tau}_T = Y_{1T}(1) - \hat{Y}_{1T}(0)$ based on (3.3) a “prediction” of τ_T rather than an “estimator” of it, and our goal is to characterize the uncertainty of $\hat{\tau}_T$ 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 \mathcal{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 the synthetic control estimands \mathbf{w}_0 and \mathbf{r}_0 introduced in (3.2):

$$Y_{1T}(0) \equiv \mathbf{x}'_T \mathbf{w}_0 + \mathbf{g}'_T \mathbf{r}_0 + e_T = \mathbf{p}'_T \boldsymbol{\beta}_0 + e_T, \quad T > T_0, \quad (4.1)$$

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

$$\widehat{\tau}_T - \tau_T = Y_{1T}(0) - \widehat{Y}_{1T}(0) = e_T - \mathbf{p}'_T(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0). \quad (4.2)$$

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 $\mathbf{p}'_T(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ is the in-sample error coming from the estimation 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 \mathcal{H} ,

$$\mathbb{P}[M_{1,L} \leq \mathbf{p}'_T(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) \leq M_{1,U} \mid \mathcal{H}] \geq 1 - \alpha_1 \quad \text{and} \quad \mathbb{P}[M_{2,L} \leq e_T \leq M_{2,U} \mid \mathcal{H}] \geq 1 - \alpha_2.$$

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 \mathcal{H} ,

$$\mathbb{P}[\widehat{\tau}_T + M_{1,L} - M_{2,U} \leq \tau_T \leq \widehat{\tau}_T + M_{1,U} - M_{2,L} \mid \mathcal{H}] \geq 1 - \alpha_1 - \alpha_2.$$

4.1 In-Sample Error

Cattaneo, Feng and Titiunik (2021) provide a principled simulation-based method for quantifying the in-sample uncertainty coming from $\mathbf{p}'_T(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$. 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 (3.1), $\widehat{\boldsymbol{\delta}} := \mathbf{D}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ is the optimizer of the centered criterion function:

$$\widehat{\boldsymbol{\delta}} = \arg \min_{\boldsymbol{\delta} \in \Delta} \{\boldsymbol{\delta}' \widehat{\mathbf{Q}} \boldsymbol{\delta} - 2\widehat{\boldsymbol{\gamma}}' \boldsymbol{\delta}\},$$

where $\widehat{\mathbf{Q}} = \mathbf{D}^{-1}\mathbf{Z}'\mathbf{V}\mathbf{Z}\mathbf{D}^{-1}$, $\widehat{\boldsymbol{\gamma}}' = \mathbf{u}'\mathbf{V}\mathbf{Z}\mathbf{D}^{-1}$, and $\Delta = \{\mathbf{h} \in \mathbb{R}^d : \mathbf{h} = \mathbf{D}(\boldsymbol{\beta} - \boldsymbol{\beta}_0), \boldsymbol{\beta} \in \mathcal{W} \times \mathcal{R}\}$. 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:

$$\ell^*(\boldsymbol{\delta}) = \boldsymbol{\delta}'\widehat{\mathbf{Q}}\boldsymbol{\delta} - 2(\mathbf{G}^*)'\boldsymbol{\delta}, \quad \mathbf{G}^* \sim \mathbf{N}(0, \widehat{\boldsymbol{\Sigma}}), \quad (4.3)$$

where $\widehat{\boldsymbol{\Sigma}}$ is some estimate of $\boldsymbol{\Sigma} = \mathbb{V}[\widehat{\boldsymbol{\gamma}}|\mathcal{H}]$ and $\mathbf{N}(0, \widehat{\boldsymbol{\Sigma}})$ represents the normal distribution with mean 0 and variance-covariance matrix $\widehat{\boldsymbol{\Sigma}}$. In practice, the criterion function $\ell^*(\cdot)$ can be simulated by simply drawing normal random vectors \mathbf{G}^* . This idea is extended in [Cattaneo, Palomba, Feng and Titiunik \(2022\)](#) to allow for non-linear constraints in the feasibility set (e.g., Ridge least squares regression).

Moreover, let Δ^* denote the constraint set used in simulation, which is assumed to be locally identical to (or approximated by) the infeasible (normalized) constraint set Δ :

$$\Delta^* \cap \mathcal{B}(\mathbf{0}, \varepsilon) = \Delta \cap \mathcal{B}(\mathbf{0}, \varepsilon), \quad \text{for some small } \varepsilon > 0, \quad (4.4)$$

where $\mathcal{B}(\mathbf{0}, \varepsilon)$ is an ε -neighborhood around zero. [Cattaneo, Palomba, Feng and Titiunik \(2022\)](#) provide principled tuning parameter choices and discusses their implementation in practice. Section 4.3 provides more details on how Δ^* is constructed and implemented in the `scpi` package.

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

$$M_{1,L} := (\alpha_1/2)\text{-quantile of } \inf \{\mathbf{p}_T'\mathbf{D}^{-1}\boldsymbol{\delta} : \boldsymbol{\delta} \in \Delta^*, \ell^*(\boldsymbol{\delta}) \leq 0\}, \quad \text{and}$$

$$M_{1,U} := (1 - \alpha_1/2)\text{-quantile of } \sup \{\mathbf{p}_T'\mathbf{D}^{-1}\boldsymbol{\delta} : \boldsymbol{\delta} \in \Delta^*, \ell^*(\boldsymbol{\delta}) \leq 0\},$$

conditional on the data. Under mild regularity conditions, [Cattaneo, Feng and Titiunik \(2021\)](#) and [Cattaneo, Palomba, Feng and Titiunik \(2022\)](#) show that for a large class synthetic control estimators (3.1), with high probability over \mathcal{H} ,

$$\mathbb{P}[M_{1,L} \leq \mathbf{p}_T'(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0) \leq M_{1,U} \mid \mathcal{H}] \geq 1 - \alpha_1,$$

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 ($\mathbb{E}[\mathbf{u}|\mathcal{H}] = 0$) or not.

4.2 Out-of-Sample Error

The unobserved random variable e_T in (4.1) is a single error term in period T , which we interpret as the error from out-of-sample prediction, conditional on \mathcal{H} . Naturally, in order to have a proper bound on e_T , it is necessary to determine certain features of its conditional distribution $F_{e_T}(\mathbf{e}) = \mathbb{P}[e_T \leq \mathbf{e}|\mathcal{H}]$. 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.

- **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 \mathcal{H} , i.e., there exists some $\sigma_{\mathcal{H}} > 0$ such that $\mathbb{E}[\exp(\lambda(e_T - \mathbb{E}[e_T|\mathcal{H}]))|\mathcal{H}] \leq \exp(\sigma_{\mathcal{H}}^2 \lambda^2 / 2)$ a.s. for all $\lambda \in \mathbb{R}$. Then, we can take

$$M_{2,L} := \mathbb{E}[e_T|\mathcal{H}] - \sqrt{2\sigma_{\mathcal{H}}^2 \log(2/\alpha_2)} \quad \text{and} \quad M_{2,U} := \mathbb{E}[e_T|\mathcal{H}] + \sqrt{2\sigma_{\mathcal{H}}^2 \log(2/\alpha_2)}.$$

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

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

$$M_{2,L} = \mathbb{E}[e_T|\mathcal{H}] + (\mathbb{V}[e_T|\mathcal{H}])^{1/2}\mathbf{c}_{\varepsilon}(\alpha_2/2) \quad \text{and} \quad M_{2,U} = \mathbb{E}[e_T|\mathcal{H}] + (\mathbb{V}[e_T|\mathcal{H}])^{1/2}\mathbf{c}_{\varepsilon}(1 - \alpha_2/2),$$

where $\mathbf{c}_{\varepsilon}(\alpha_2/2)$ and $\mathbf{c}_{\varepsilon}(1 - \alpha_2/2)$ are $\alpha_2/2$ and $(1 - \alpha_2/2)$ quantiles of ε_t , respectively. In practice, $\mathbb{E}[e_T|\mathcal{H}]$ and $\mathbb{V}[e_T|\mathcal{H}]$ 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 $e_T|\mathcal{H}$, that is,

$$M_{2,L} := \alpha_2/2\text{-quantile of } e_T|\mathcal{H} \quad \text{and} \quad M_{2,U} := (1 - \alpha_2/2)\text{-quantile of } e_T|\mathcal{H}.$$

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 :

$$\mathbb{P}[M_{2,L} \leq e_T \leq M_{2,U} \mid \mathcal{H}] \geq 1 - \alpha_2.$$

4.3 Implementation

We now discuss how to implement uncertainty quantification. 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 to model separately the in-sample error $\mathbf{p}'_T(\hat{\beta} - \beta_0)$ 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 estimation of $\mathbf{p}'_T(\hat{\beta} - \beta_0)$, and its quantification reduces to determining $M_{1,L}$ and $M_{1,U}$. We first review the methodological proposals for constructing the constraint set Δ^* used in simulation discussed in [Cattaneo, Palomba, Feng and Titiunik \(2022\)](#), and then present the main procedure for constructing bounds on the in-sample error.

Constructing Δ^* . Our in-sample uncertainty quantification requires that the centered and scaled constraint feasibility set Δ be locally identical to (or, at least, well approximated by) the constraint set Δ^* used in the simulations described in (4.3), in the sense of (4.4). In practice, we modify

the estimated synthetic control weights $\hat{\mathbf{w}} = (\hat{\omega}_2, \dots, \hat{\omega}_{N+1})'$ using a tuning parameter $\varrho > 0$ and construct Δ^* accordingly.

First, ϱ is estimated according to the following formula where we let $M = 1$ for simplicity:

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

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

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

with \mathcal{C}_1 by default, and where $\hat{\sigma}_{b_j, u}$ is the estimated (unconditional) covariance between the pseudo-true residual \mathbf{u} and the feature of the j th control unit $\mathbf{B}_{j,1}$, and $\hat{\sigma}_u$ and $\hat{\sigma}_{b_j}$ are the estimated (unconditional) standard deviation 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.

Second, given the choice of ϱ , we construct Δ^* depending on the constraint set \mathcal{W} :

- **Simplex.** Let $\hat{\beta}^* = (\hat{\mathbf{w}}^*, \hat{\mathbf{r}}')'$, $\hat{\mathbf{w}}^* = (\hat{\omega}_2^*, \dots, \hat{\omega}_{N+1}^*)'$, and $\hat{\omega}_j^* = \hat{\omega}_j \mathbf{1}(|\hat{\omega}_j| > \varrho)$. Then, $\Delta^* = \{\mathbf{D}(\beta - \hat{\beta}^*) : \beta = (\mathbf{w}', \mathbf{r}')', \mathbf{w} \in \mathbb{R}_+^N, \|\mathbf{w}\|_1 = \|\hat{\mathbf{w}}^*\|_1\}$.
- **Lasso.** The regularization takes two steps. First, if $\|\hat{\mathbf{w}}\|_1 \in [Q - \sqrt{N}\varrho, Q]$, then set $Q^* = \|\hat{\mathbf{w}}\|_1$, otherwise set $Q^* = Q$. Second, define $\hat{\mathbf{w}}^* = (\hat{\omega}_2^*, \dots, \hat{\omega}_{N+1}^*)'$, and $\hat{\omega}_j^* = \hat{\omega}_j \mathbf{1}(|\hat{\omega}_j| > \varrho)$. Then, let $\Delta^* = \{\mathbf{D}(\beta - \hat{\beta}^*) : \beta = (\mathbf{w}', \mathbf{r}')', \|\mathbf{w}\|_1 \leq Q^*\}$ where $\hat{\beta}^* = (\hat{\mathbf{w}}^*, \hat{\mathbf{r}}')'$.
- **Ridge.** If $\|\hat{\mathbf{w}}\|_2 \in [Q - \varrho, Q]$, then set $Q^* = \|\hat{\mathbf{w}}\|_2$, otherwise set $Q^* = Q$. Then, $\Delta^* = \{\mathbf{D}(\beta - \hat{\beta}^*) : \beta = (\mathbf{w}', \mathbf{r}')', \|\mathbf{w}\|_2 \leq Q^*\}$ where $\hat{\beta}^* = (\hat{\mathbf{w}}^*, \hat{\mathbf{r}}')'$.

Degrees-of-Freedom Correction. Our uncertainty quantification strategy requires an estimator of the conditional variance $\mathbb{V}[\mathbf{u}|\mathcal{H}]$, which may rely on the effective degrees of freedom \mathbf{df} of the synthetic control method. A naïve approach to estimate \mathbf{df} would use $\hat{\mathbf{df}} = T_0 \cdot M - |j : \hat{\omega}_j^* > 0| - KM$. However, in general there exists no exact correspondence between \mathbf{df} and the number of parameters in a fitting model (Ye, 1998). Therefore, $\hat{\mathbf{df}}$ is defined according to the chosen constraint sets for β underlying the estimation procedure in (3.1).

Define the “active donor set” $\mathcal{A} = \{j : \hat{w}_j^* > 0\}$. We can decompose \mathbf{df} as the sum of two terms: $\mathbf{df}(\mathbf{B}\hat{\mathbf{w}})$ and $\mathbf{df}(\mathbf{C}\hat{\mathbf{r}})$. On the one hand, $\hat{\mathbf{df}}(\mathbf{C}\hat{\mathbf{r}}) = KM$ in all cases. On the other hand, $\mathbf{df}(\mathbf{B}\hat{\mathbf{w}})$ and its estimator are defined depending on the chosen constraint set \mathcal{W} :

- **OLS.** $\mathbf{df}(\mathbf{B}\hat{\mathbf{w}}) = J$.
- **Lasso.** Following [Zou, Hastie and Tibshirani \(2007\)](#), $\mathbf{df}(\mathbf{B}\hat{\mathbf{w}}) = \mathbb{E}[\text{rank}(\mathbf{B}_{\mathcal{A}})]$ and an unbiased and consistent estimator is $\hat{\mathbf{df}}(\mathbf{B}\hat{\mathbf{w}}) = \sum_{j=1}^J \mathbb{1}(\hat{w}_j > 0)$.
- **Simplex.** Following the discussion for Lasso, $\mathbf{df}(\mathbf{B}\hat{\mathbf{w}}) = \mathbb{E}[\text{rank}(\mathbf{B}_{\mathcal{A}})] - 1$ and $\hat{\mathbf{df}}(\mathbf{B}\hat{\mathbf{w}}) = \sum_{j=1}^J \mathbb{1}(\hat{w}_j > 0) - 1$, where $\mathbf{B}_{\mathcal{A}}$ denotes the submatrix of \mathbf{B} whose column indices belong to \mathcal{A} .
- **Ridge.** Let $d_1 \geq d_2 \geq \dots \geq d_J \geq 0$ be singular values of \mathbf{B} and λ be the complexity parameter of the corresponding Lagrangian Ridge problem, which satisfies $\lambda\hat{\mathbf{w}} = \mathbf{B}'(\mathbf{A} - \mathbf{B}\hat{\mathbf{w}})$. Then, following [Friedman, Hastie and Tibshirani \(2001\)](#), $\hat{\mathbf{df}}(\mathbf{B}\hat{\mathbf{w}}) = \sum_{j=1}^J \frac{d_j^2}{d_j^2 + \lambda}$.

Main procedure. Given the constraint set Δ^* , 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 Σ and to simulate the criterion function (4.3) we need an estimate of $\mathbb{V}[\hat{\gamma}|\mathcal{H}]$ 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 $\mathbb{V}[\mathbf{u}|\mathcal{H}]$, via the option `u.sigma`.

Let $\mathbf{B}^* = \text{diag}(\mathbf{B}_1^*, \mathbf{B}_2^*, \dots, \mathbf{B}_M^*)$, where \mathbf{B}_l^* denote the matrix composed of the columns of \mathbf{B}_l with non-zero regularized weight $\hat{\omega}_j^*$ only. If the option `cointegrated.data` in `sdata()` 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 $\mathbb{E}[\mathbf{u}|\mathcal{H}] = 0$, whereas if `u.missp = TRUE` (default), then $\mathbb{E}[\mathbf{u}|\mathcal{H}]$ needs to be estimated.

The unknown conditional expectation $\mathbb{E}[\mathbf{u}|\mathcal{H}]$ is estimated using the fitted values of a flexible linear-in-parameters regression of $\hat{\mathbf{u}} = \mathbf{A} - \mathbf{B}\hat{\mathbf{w}} - \mathbf{C}\hat{\mathbf{r}}$ 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}^* (`u.lags`) and/or the order of the fully interacted polynomial in \mathbf{B}^* (`u.order`).

For example, if the user specifies `u.lags = 1` and `u.order = 1`, then the design matrix $\mathbf{D}_{\mathbf{u}} = [\mathbf{B}^* \quad \mathbf{B}_{-1}^* \quad \mathbf{C}]$, where \mathbf{B}_{-1}^* indicates the first lag of \mathbf{B}^* . If, instead, `u.order = 0` and `u.lags = 0` are specified, then $\hat{\mathbb{E}}[\mathbf{u}|\mathcal{H}] = \bar{\mathbf{u}} \otimes \boldsymbol{\iota}_{T_0}$, where $\bar{\mathbf{u}} = (\bar{u}_1, \bar{u}_2, \dots, \bar{u}_M)'$ with $\bar{u}_l = T_0^{-1} \sum_{t=1}^{T_0} \hat{u}_{t,l}$, $\hat{\mathbf{u}} = (\hat{u}_{1,1}, \dots, \hat{u}_{T_0,1}, \hat{u}_{1,2}, \dots, \hat{u}_{T_0,2}, \dots, \hat{u}_{1,M}, \dots, \hat{u}_{T_0,M})'$, and $\boldsymbol{\iota}_{\nu}$ is a $\nu \times 1$ vector of ones.

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

$$\hat{\mathbb{V}}[\mathbf{u}|\mathcal{H}] = (\hat{\mathbf{u}} - \hat{\mathbb{E}}[\mathbf{u}|\mathcal{H}])(\hat{\mathbf{u}} - \hat{\mathbb{E}}[\mathbf{u}|\mathcal{H}])' \odot \mathbf{vc}$$

where \odot denote the Hadamard product and \mathbf{vc} is a variance-correction diagonal matrix of dimension $T_0 \cdot M \times T_0 \cdot M$. The user can choose the diagonal elements of \mathbf{vc} among various members of the well-known heteroskedasticity-robust family of variance-covariance estimators through the option `u.sigma`. In particular,

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

with \mathbf{L}_{ii} being the i -th diagonal entry of the leverage matrix $\mathbf{L} := \mathbf{Z}(\mathbf{Z}'\mathbf{V}\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{V}$, $\delta_i = \min\{4, T_0 \cdot M \cdot \mathbf{P}_{ii}/\text{df}\}$, and df is a degrees-of-freedom correction factor, whose estimation is explained above.

Step 2. *Estimation of $\boldsymbol{\Sigma}$.* The estimator of $\boldsymbol{\Sigma}$ is $\hat{\boldsymbol{\Sigma}} = (\mathbf{Z}'\mathbf{V})\hat{\mathbb{V}}[\mathbf{u}|\mathcal{H}](\mathbf{V}\mathbf{Z})$.

Step 3. *Simulation.* The criterion function $\ell^*(\boldsymbol{\delta})$ in (4.3) is simulated by drawing i.i.d. random vectors from the Gaussian distribution $\mathbf{N}(0, \hat{\boldsymbol{\Sigma}})$, conditional on the data.

Step 4. *Optimization.* Let $\ell_{(s)}^*(\boldsymbol{\delta})$ denote the criterion function corresponding to the s -th draw from

$N(0, \widehat{\Sigma})$. For each draw s , we solve the following constrained problems:

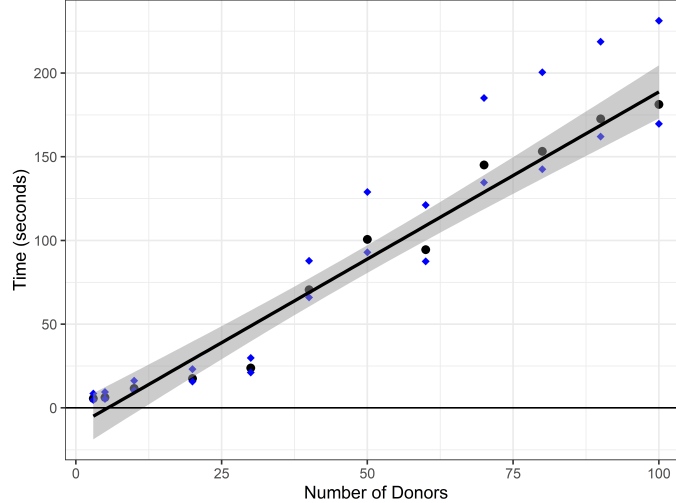
$$l_{(s)} := \inf_{\delta \in \Delta^*, \ell_{(s)}^*(\delta) \leq 0} \mathbf{p}'_T \mathbf{D}^{-1} \delta \quad \text{and} \quad u_{(s)} := \sup_{\delta \in \Delta^*, \ell_{(s)}^*(\delta) \leq 0} \mathbf{p}'_T \mathbf{D}^{-1} \delta, \quad (4.5)$$

where Δ^* is constructed as explained previously.

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

Parallelization and Execution Speed. Steps 3 and 4 of the procedure above are the most computationally intensive ones. However, the procedure we implement can be sped up by efficient parallelization of the tasks performed by the command `scpi`. Specifically, different simulations are assigned to different cores by means of the package `parallel`. Therefore, if N_{cores} cores are used, the final execution time would be approximately $T_{\text{exec}}/N_{\text{cores}}$, where T_{exec} is the execution time when a single core is used. Finally, Figure 1 shows that the execution time of the main function `scpi` is linear in the number of donors J used to compute the synthetic unit.

Figure 1: Execution time of `scpi` with $T_0 = 1000$, $T_1 = 1$, $M = 1$, $S = 200$, and $N_{\text{cores}} = 1$.



Notes: We evaluate the performance of the function `scpi` through the R package `microbenchmark`. Black dots represent the median execution time, whereas blue dots are the 5-th and 95-th percentiles. The black line is obtained by fitting a linear regression of median execution time on the number of donors. The shaded area shows 95% confidence bands. This simulation was run in Windows 10 x64, RAM 8.00 GB, processor Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz.

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 $\mathbb{P}[e_T \leq \mathbf{e}|\mathcal{H}]$ and, ultimately, estimate $M_{2,L}$ and $M_{2,U}$. We discussed three different alternative approaches: (i) non-asymptotic bounds; (ii) location-scale model; and (iii) quantile regressions. The user can choose the preferred way of modeling $e_T|\mathcal{H}$ with the option `e.method`.

The user can also choose the information used to estimate (conditional) moments or quantiles of $e_T|\mathcal{H}$. 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 $\mathbf{D}_e = [\mathbf{B}_1^* \ \mathbf{C}_1]$. Alternatively, the matrix \mathbf{D}_e can be provided directly through the option `e.design` or by specifying the lags of \mathbf{B}_1^* (`e.lags`) and/or the order of the fully interacted polynomial in \mathbf{B}_1^* (`e.order`). If the user specifies `e.lags = 0` and `e.order = 2`, then \mathbf{D}_e contains \mathbf{B}_1^* , \mathbf{C}_1 , and all the unique second-order terms generated by the interaction of the columns of \mathbf{B}_1^* . If instead `e.order = 0` and `e.lags = 0` are set, then $\hat{\mathbb{E}}[e_T|\mathcal{H}]$ and $\hat{\mathbb{V}}[e_T|\mathcal{H}]$ 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^* 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 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, $e_T|\mathcal{H}$ 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 $M_{2,L}$ and $M_{2,U}$ 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 $e_T|\mathcal{H}$ 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 $\sigma_{\mathcal{H}}$ 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 estimating the causal impact of the German reunification on GDP per capita in West Germany. Thus, we compare the post-reunification outcome for West Germany with the outcome of a synthetic control 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 $K = 0$, and $\mathcal{R} = \mathbb{R}$. We explore the effect of the reunification from 1991 to 2003, hence $T_1 = 13$. Finally, we treat the times series for West Germany and the donor pool as a cointegrating system. Given this information, the command `sdata()` prepares all the matrices needed to estimate the synthetic control (\mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{P}), and returns an object that must be used as input in either `scest()` to conduct point estimation, or `scpi()` to conduct inference.

We first call `sdata()` to transform any data frame into an object of class “`scpi_data`”.

```
# Load data
> data <- scpi_germany
>
> ## Set parameters for data preparation
> id.var      <- "country"      # ID variable
> time.var    <- "year"         # Time variable
```

```

> period.pre <- (1960:1990)           # Pre-treatment period
> period.post <- (1991:2003)          # Post-treatment period
> unit.tr <- "West Germany"           # Treated unit
> unit.co <- unique(data$country)[-7] # Donors pool
> outcome.var <- "gdp"                # Outcome variable
> constant <- T                       # Include constant term
> cointegrated.data <- T              # Cointegrated data
>
# Data preparation
> df <- sdata(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)

```

After having prepared the data, the next step involves choosing the desired constraint set \mathcal{W} to estimate the vector of weights \mathbf{w} . We consider the canonical synthetic control method and thus search for optimal weights in $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}_+^J : \|\widehat{\mathbf{w}}\|_1 = 1, w_j \geq 0, j = 1, \dots, J\}$. Such constraint set is the default in `scest()` and, consequently, in `scpi()`, as the latter internally calls the former to estimate \mathbf{w} . The snippet below illustrates how to call `scest()` and reports the results obtained in the console with the `summary()` method.

```

# Estimate SC with a simplex-type constraint (default)
> res.est <- scest(data = df, w.constr = list(name="simplex"))
> 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 **B** and **C** as the conditioning set in both cases. To do so, it is enough to set the order of the polynomial in **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 **u** might differ from 0. This option, together with `u.sigma <- "HC1"`, specifies the following estimator for $\mathbb{V}[\mathbf{u}|\mathcal{H}]$:

$$\widehat{\mathbb{V}}[\mathbf{u}|\mathcal{H}] = \frac{1}{T_0} \sum_{t=1}^{T_0} \text{vc}_t^{(1)} \mathbf{b}_t \mathbf{b}_t' (\hat{\mathbf{u}}_t - \widehat{\mathbb{E}}[\mathbf{u}_t|\mathcal{H}])^2.$$

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

```
## Quantify uncertainty
> sims      <- 2000          # Number of simulations
> u.order   <- 1            # Degree of polynomial in B and C when modelling u
> u.lags    <- 0            # Lags of B to be used when modelling u
> u.sigma   <- "HC1"        # Estimator for the variance-covariance of u
> u.missp   <- T            # If TRUE then the model is treated as misspecified
> e.order   <- 1            # Degree of polynomial in B and C when modelling e
> e.lags    <- 0            # Lags of B to be used when modelling e
> e.method  <- "qreg"       # Estimation method for out-of-sample uncertainty
> cores     <- 3            # Number of cores to be used by scpi

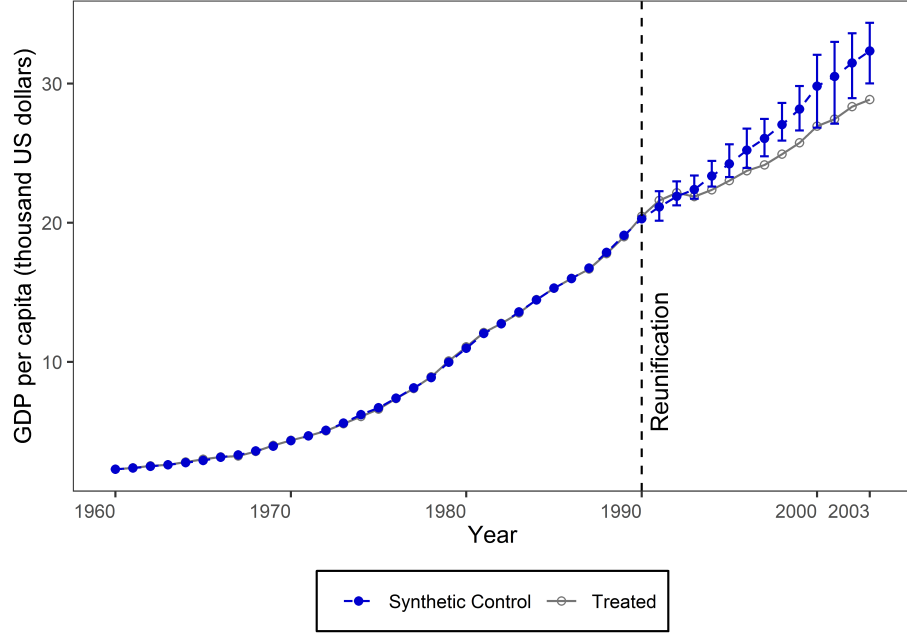
> set.seed(8894)
> res.pi <- scpi(data = df, sims = sims, e.method = e.method, e.order = e.order,
+               e.lags = e.lags, u.order = u.order, u.lags = u.lags,
+               u.sigma = u.sigma, u.missp = u.missp, cores = cores,
+               w.constr = list(name = "simplex"))

# Visualize results
> 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 = NULL, e.out = T,
+               event.label = list(lab = "Reunification", height = 10))

> plot <- plot$plot_out + ggtitle("")
> ggsave(filename = 'germany_unc_simplex.png', plot = plot)
```

Figure 2 displays the plot resulting from the `scplot` call. 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. $\alpha_1 = \alpha_2 = 0.05$.

Figure 2: Treated and synthetic unit using a simplex-type \mathcal{W} and 90% prediction intervals



We also conduct the same exercise using different choices for \mathcal{W} (see Table 2). In particular, we estimate weights and compute prediction intervals using three other specifications: (i) a lasso-type constraint (Figure 3a), (ii) a ridge-type constraint (Figure 3b), and (iii) no constraint (Figure 3c).

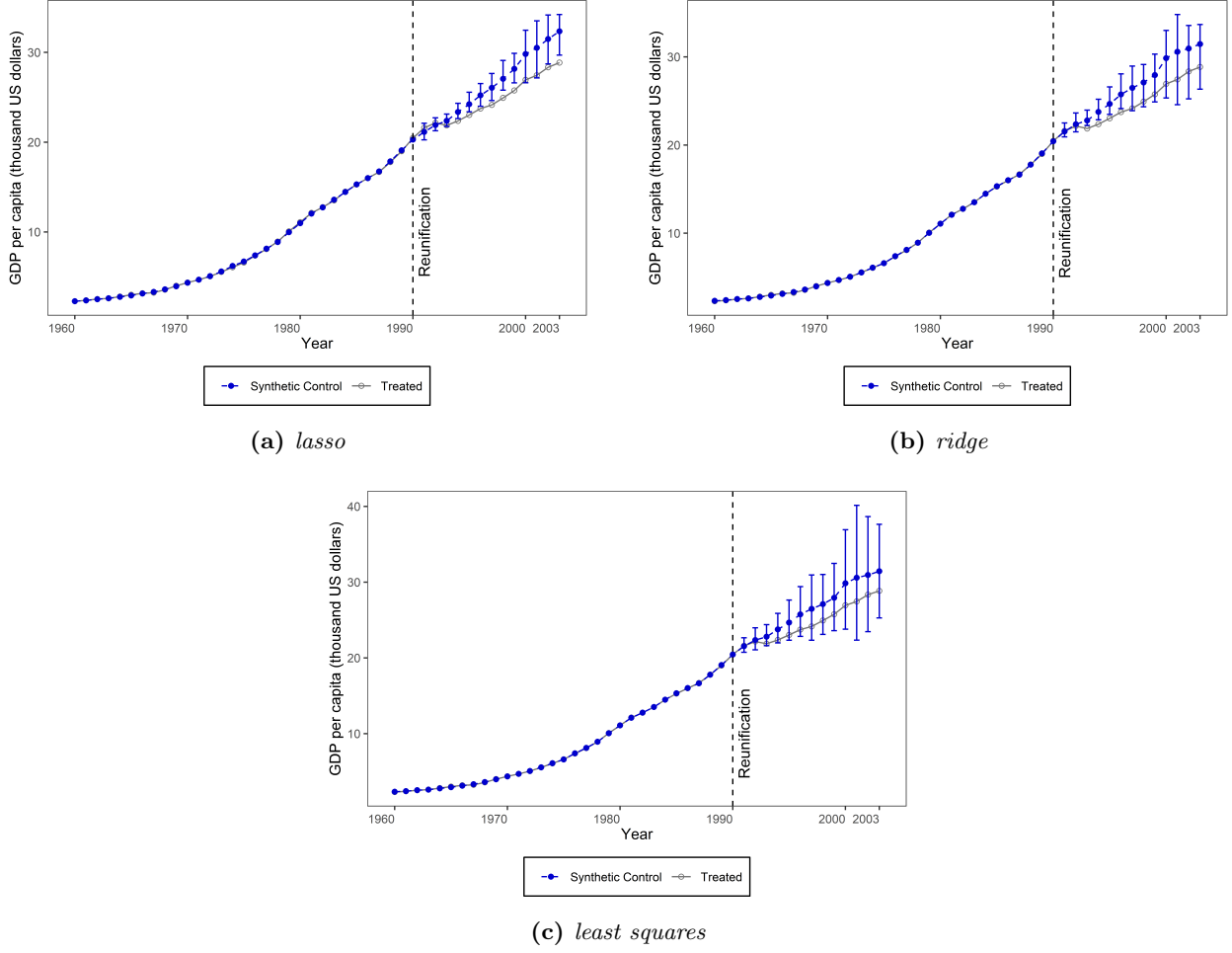
```
# Comparison of different constraint sets for the weights
> methods <- c("lasso", "ridge", "ols")

> for (method in methods) {
>   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,
+                 u.sigma = u.sigma, u.misss = u.misss, cores = cores,
+                 w.constr = list(name = method))

  # Visualize results
>   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 = NULL, e.out = T,
+                 event.label = list(lab = "Reunification", height = 10))

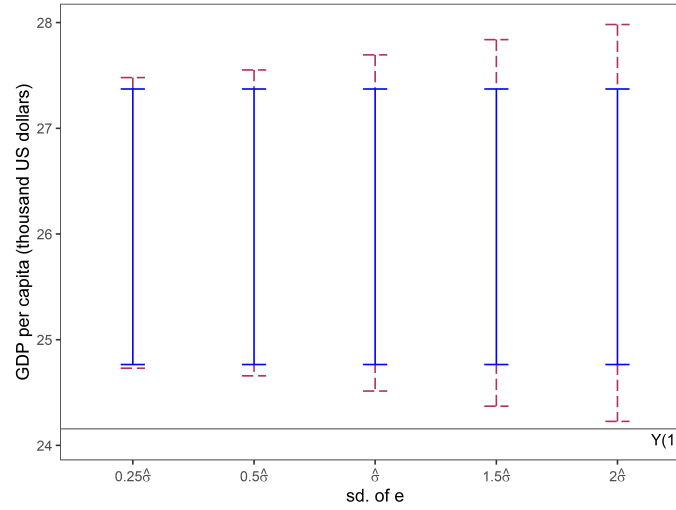
>   plot <- plot$plot_out + ggtitle("")
>   ggsave(filename = paste0('germany_unc_', method, '.png'), plot = plot)
}
```

Figure 3: *Uncertainty quantification with different types of \mathcal{W} using 90% prediction intervals.*



Section 4.4 clarified the need of some additional sensitivity analysis when it comes to out-of-sample uncertainty quantification. Figure 4 shows the impact of shrinking and enlarging $\hat{\sigma}_{\mathcal{H}}$ on the prediction intervals for $\hat{Y}_{1997}(0)$ when we assume that e_T is sub-Gaussian conditional on \mathcal{H} . As shown in the figure, the estimated treatment effect $\hat{\tau}_{1997}$ remains statistically significant even doubling $\hat{\sigma}_{\mathcal{H}}$.

Figure 4: *Sensitivity analysis on out-of-sample uncertainty with sub-Gaussian bounds.*



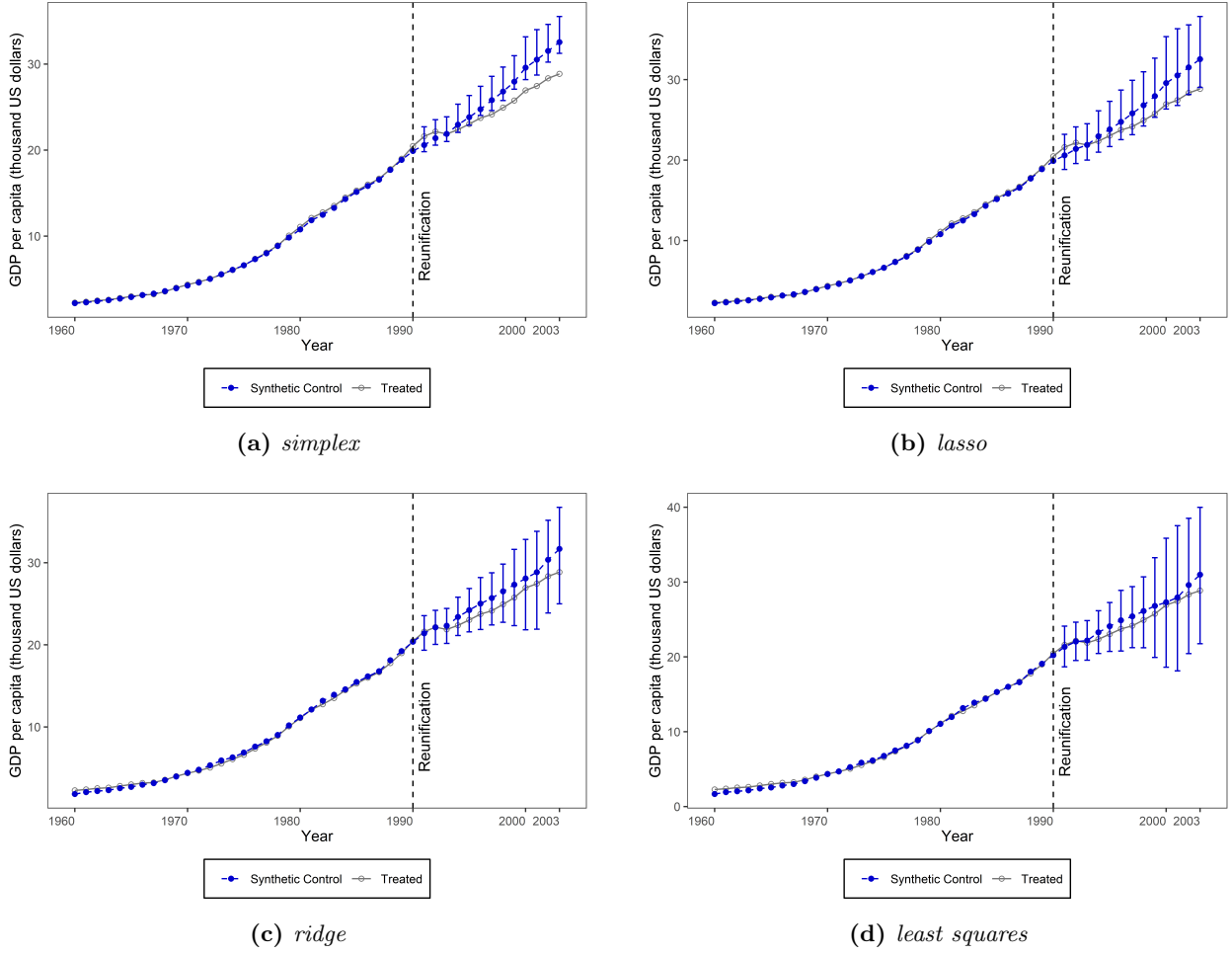
Notes: The black horizontal line shows the level of the outcome for the treated unit in 1997, $Y_T(1)$ for $T = 1997$. The blue bars report 95% prediction intervals for $Y_T(0)$, $T = 1997$, that only take into account in-sample uncertainty. The red dashed bars adds the out-of-sample uncertainty to obtain 90% prediction intervals.

Finally, the package offers the possibility to match the treated unit and the synthetic unit using multiple features. 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.

```
## Data preparation
df <- sccdata(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,
             features = c("gdp", "trade"), unit.co = unit.co,
             cointegrated.data = cointegrated.data)
```

Results are reported in Figure 5.

Figure 5: *Uncertainty quantification with different types of \mathcal{W} using 90% prediction intervals.*



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/>.

7 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 Cattaneo gratefully acknowledges financial support from the National Institute of Health (R01 GM072611-16).

References

- Abadie, A. (2021), “Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects,” *Journal of Economic Literature*, 59, 391–425.
- Abadie, A., and Cattaneo, M. D. (2018), “Econometric Methods for Program Evaluation,” *Annual Review of Economics*, 10, 465–503.
- Abadie, A., Diamond, A., and 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, 493–505.
- Abadie, A., and Gardeazabal, J. (2003), “The Economic Costs of Conflict: A Case Study of the Basque Country,” *American Economic Review*, 93, 113–132.
- Abadie, A., and L’Hour, J. (2021), “A Penalized Synthetic Control Estimator for Disaggregated Data,” *Journal of the American Statistical Association*, 116, 1817–1834.
- Amjad, M., Shah, D., and Shen, D. (2018), “Robust Synthetic Control,” *The Journal of Machine Learning Research*, 19, 802–852.
- Bai, J. (2009), “Panel Data Models with Interactive Fixed Effects,” *Econometrica*, 77, 1229–1279.
- Ben-Michael, E., Feller, A., and Rothstein, J. (2022), “Synthetic Controls with Staggered Adoption,” *Journal of the Royal Statistical Society, Series B*, forthcoming.
- Cattaneo, M. D., Feng, Y., and Titiunik, R. (2021), “Prediction Intervals for Synthetic Control Methods,” *Journal of the American Statistical Association*, 116, 1865–1880.
- Cattaneo, M. D., Palomba, F., Feng, Y., and Titiunik, R. (2022), “Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption,” working paper.

- Chernozhukov, V., Wüthrich, K., and Zhu, Y. (2021), “An Exact and Robust Conformal Inference Method for Counterfactual and Synthetic Controls,” *Journal of the American Statistical Association*, 116, 1849–1864.
- Ferman, B., and Pinto, C. (2021), “Synthetic Controls with Imperfect Pretreatment Fit,” *Quantitative Economics*, 12, 1197–1221.
- Friedman, J., Hastie, T., and Tibshirani, R. (2001), *The Elements of Statistical Learning*, Springer, New York.
- Fu, A., Narasimhan, B., and Boyd, S. (2020), “CVXR: An R Package for Disciplined Convex Optimization,” *Journal of Statistical Software*, 94, 1–34.
- Hoerl, A. E., Kannard, R. W., and Baldwin, K. F. (1975), “Ridge Regression: Some Simulations,” *Communications in Statistics-Theory and Methods*, 4, 105–123.
- Hsiao, C., Steve Ching, H., and 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, 705–740.
- Johnson, S. G. (2022), “The NLOpt nonlinear-optimization package,” <https://nlopt.readthedocs.io/en/latest/>.
- Li, K. T. (2020), “Statistical Inference for Average Treatment Effects Estimated by Synthetic Control Methods,” *Journal of the American Statistical Association*, 115, 2068–2083.
- Masini, R., and Medeiros, M. C. (2021), “Counterfactual Analysis with Artificial Controls: Inference, High Dimensions and Nonstationarity,” *Journal of the American Statistical Association*, 116, 1773–1788.
- Robbins, M. W., Saunders, J., and Kilmer, B. (2017), “A framework for synthetic control methods with high-dimensional, micro-level data: evaluating a neighborhood-specific crime intervention,” *Journal of the American Statistical Association*, 112, 109–126.
- Shaikh, A. M., and Toulis, P. (2021), “Randomization Tests in Observational Studies With Staggered Adoption of Treatment,” *Journal of the American Statistical Association*, 116, 1835–1848.

- Ye, J. (1998), “On measuring and Correcting the Effects of Data Mining and Model Selection,” *Journal of the American Statistical Association*, 93, 120–131.
- Zou, H., Hastie, T., and Tibshirani, R. (2007), “On the “degrees of freedom” of the lasso,” *The Annals of Statistics*, 35, 2173–2192.

A Appendix: Python Illustration

This appendix section shows how to conduct the same analysis carried out in Section 5 for $M = 1$ using the companion Python package. Figure 6 shows the main results.

```
#####
# Replication file for Cattaneo, Feng, Palomba, and Titiunik (2022)
#####

#####
# Load SCPI_PKG package
import pandas
import numpy
import random
import os
from warnings import filterwarnings
from plotnine import ggtitle, ggsave
from scpi_pkg.scddata import scdata
from scpi_pkg.scest import scest
from scpi_pkg.scpi import scpi
from scpi_pkg.scplot import scplot

filterwarnings('ignore')

#####
# Load database
data = pandas.read_csv('scpi_germany.csv')

#####
# Set options for data preparation
id_var = 'country'
outcome_var = 'gdp'
time_var = 'year'
period_pre = numpy.arange(1960, 1991)
period_post = numpy.arange(1991, 2003)
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)

#####
# SC - point estimation with simplex
est_si = scest(data_prep, w_constr={'name': 'simplex'})
print(est_si)

#####
# Set options for inference
w_constr = {'name': 'simplex', 'Q': 1}
```

```

u_missp = True
u_sigma = 'HCl'
u_order = 1
u_lags = 0
e_method = 'qreg'
e_order = 1
e_lags = 0
sims = 2000
cores = 1

# Simplex
random.seed(8894)
pi_si = scpi(data_prep, sims=sims, w_constr=w_constr, u_order=u_order,
             u_lags=u_lags, e_order=e_order, e_lags=e_lags,
             e_method=e_method, u_missp=u_missp,
             u_sigma=u_sigma, cores=cores)

plot = scplot(pi_si, x_lab='Year', e_method = 'qreg',
             y_lab='GDP per capita (thousand US dollars)')

plot = plot + ggtitle('')
ggsave(filename='py_germany_unc_simplex.png', plot=plot)

# Lasso
random.seed(8894)
pi_la = scpi(data_prep, sims=sims, w_constr={'name': 'lasso'},
             u_order=u_order, u_lags=u_lags,
             e_order=e_order, e_lags=e_lags,
             e_method=e_method, u_missp=u_missp,
             u_sigma=u_sigma, cores=cores)

plot_name = 'py_germany_unc_lasso.png'
plot = scplot(pi_la, x_lab='Year', e_method = 'qreg',
             y_lab='GDP per capita (thousand US dollars)')
plot = plot + ggtitle('')
ggsave(filename=plot_name, plot=plot)

# Ridge
random.seed(8894)
pi_ri = scpi(data_prep, sims=sims, w_constr={'name': 'ridge'},
             u_order=u_order, u_lags=u_lags,
             e_order=e_order, e_lags=e_lags,
             e_method=e_method, u_missp=u_missp,
             u_sigma=u_sigma, cores=cores)

plot_name = 'py_germany_unc_ridge.png'
plot = scplot(pi_ri, x_lab='Year', e_method = 'qreg',
             y_lab='GDP per capita (thousand US dollars)')
plot = plot + ggtitle('')
ggsave(filename=plot_name, plot=plot)

# Least Squares
random.seed(8894)
pi_ls = scpi(data_prep, sims=sims, w_constr={'name': 'ols'},
             u_order=u_order, u_lags=u_lags,
             e_order=e_order, e_lags=e_lags,
             e_method=e_method, u_missp=u_missp,
             u_sigma=u_sigma, cores=cores)

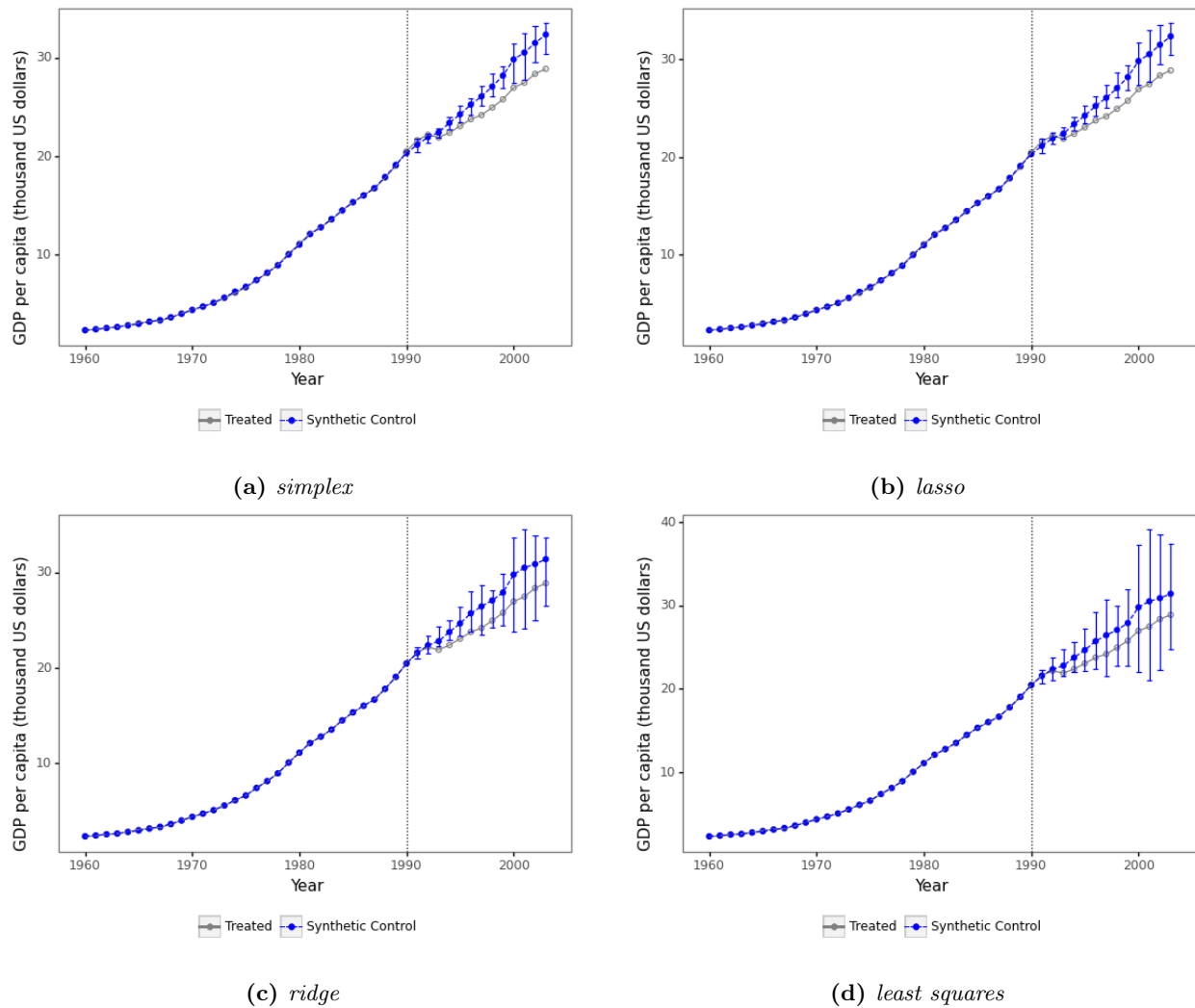
```

```

plot_name = 'py_germany_unc_ols.png'
plot = splot(pi_ls, x_lab='Year', e_method = 'qreg',
            y_lab='GDP per capita (thousand US dollars)')
plot = plot + ggtitle('')
ggsave(filename=plot_name, plot=plot)

```

Figure 6: *Uncertainty quantification with different types of \mathcal{W} using 90% prediction intervals.*



B Appendix: 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 file — Cattaneo, Feng, Palomba, and Titiunik (2022)
*****

* Load dataset
use "scpi_germany.dta", clear

* Prepare data
scdata gdp, dfname("python_scd_data") id(country) outcome(gdp) time(year) ///
    treatment(status) cointegrated constant

* Estimate Synthetic Control with a simplex-type constraint (default)
scsctest, dfname("python_scd_data") name(lasso)
scplot, scsctest

* Quantify uncertainty
foreach method in "simplex" "lasso" "ridge" "ols" {
    set seed 8894
    scpi, dfname("python_scd_data") name(`method') e_method(qreg) u_missp ///
        sims(2000)

    scplot, uncertainty("qreg") gphoptions(note("")) xtitle("Year") ///
        ytitle("GPD per capita (thousand US dollars)")
    graph export "stata_germany_unc_`method'.png", replace
}
```

Figure 7: *Uncertainty quantification with different types of \mathcal{W} using 90% prediction intervals.*

