Title

scpi —— Estimation and Inference for Synthetic Control Methods.

Syntax

scpi , dfname(*string***)** [**p(***#***) direc(***string***) Q(***#***) lb(***#***) V(***string***) name(***string***) u_missp) u_sigma(***string***) u_order(***#***) u_lags(***#***) u_alpha(***#***) sims(***#***) e_method(***string***) e_order(***#***) e_lags(***#***) e_alpha(***#***) lgapp(***string***) rho(***#***) rho_max(***#***) opt_est(***string***) opt_inf(***string***) pypinocheck**]
{p_end}

Description

scpi implements estimation and inference procedures for Synthetic Control (SC) methods using least squares, lasso, ridge, or simplex-type constraints according to Cattaneo, Feng, and Titiunik (2021). The command is a wrapper of the companion Python package. As such, the user needs to have a running version of Python with the package installed. A tutorial on how to install Python and link it to Stata can be found here.

Companion R and Python packages are described in Cattaneo, Feng, Palomba and Titiunik (2022).

Companion commands are: scdata for data preparation, scest for estimation procedures, and scplot for SC plots.

Related Stata, R, and Python packages useful for inference in SC designs are described in the following website:

https://nppackages.github.io/scpi/

For an introduction to synthetic control methods, see Abadie (2021) and references therein.

Options

dfname(*string***)** specifies the name of the Python object containing the processed data created with scdata.

Loss Function and Constraints

These options let the user specify the type of constraint to be imposed to estimate the SC weights and the loss function. The user controls the lower bound on the weights (option **lb**), the norm of the weights to be constrained (option **p**), the direction of the constraint on the norm (option **dir**), the size of the constraint on the norm (option **q**), and the shape of the wieghting matrix in the loss function (option **V**). Alternatively, some popular constraints can be selected through the option **name.** A detailed description of the popular constraints implemented can be found in Cattaneo, Feng, Palomba and Titiunik (2022).

lb(#**)** specifies the lower bound on the weights. The default is **lb(0)**.

p(#**)** sets the type of norm to be constrained. Options are:
- **0** no constraint on the norm of the weights is imposed.
- **1** a constraint is imposed on the L1 norm of the weights (the default).
- **2** a constraint is imposed on the L2 norm of the weights.

direc(*string***)** specifies the direction of the constraint on the norm of the weights. Options are:
- **<=** the constraint on the norm of the weights is an inequality constraint.
- **==** the constraint on the norm of the weights is an equality constraint (the default).

Q(#**)** specifies the size of the constraint on the norm of the weights.

name(*string***)** specifies the name of the constraint to be used. Options are:
- **simplex** classic synthetic control estimator where the weights are constrained to be non-negative and their L1 norm must be equal to 1.
- **lasso** weights are estimated using a Lasso-type penalization
- **ridge** weights are estimated using a Ridge-type penalization.
- **L1-L2** weights are estimated using a Simplex-type constraint and a Ridge-type penalization.
- **ols** weights are estimated without constraints using least squares

V(*string***)** specifies the weighting matrix to be used in the loss function. The default is the identity matrix (option **V("separate")**), so equal weight is given to all observations. The other possibility is to specify **V("pooled")** for the pooled fit.

In-sample Uncertainty

This set of options allows the user to specify her preferred approch to model in-sample uncertainty, that is uncertainty that stems from the estimation the weights.

u_missp if specified indicates that model misspecification should be taken into account.

u_sigma(*string***)** specifies the type of variance-covariance estimator to be used when estimating the conditional variance of the pseudo-residuals. Options are: **HC0**, **HC1** (default), **HC2**, and **HC3**.

u_order(#**)** specifies the order of the polynomial in the predictors used to estimate conditional moments of the pseudo-residuals. Default is **u_order(1)**. If there is risk of over-fitting the option is automatically set to 0. Our rule of thumb to predict over-fitting checks that the difference between the effective number of observations and the number of parameters used to predict the conditional moments of the pseudo-residuals is at least 20.

u_lags(#**)** specifies the lags of the predictors used to estimate conditional moments of the pseudo-residuals. Default is **u_lags(0)**. If there is risk of over-fitting the option is automatically set to 0 (see u_order for more information).

u_alpha(#**)** specifies the confidence level for in-sample uncertainty, that is the confidence level is 1 - u_alpha. Default is **u_alpha(0.05).**

sims(#) specifies the number of simulations to be used in quantifying in-sample uncertainty. Default is **sims(200).**

Out-of-sample Uncertainty

This set of options allows the user to specify her preferred approch to model out-of-sample uncertainty.

e_method(#) specifies the method to be used to quantify out-of-sample uncertainty. Options are:
gaussian conditional subgaussian bounds.
ls location-scale model.
qreg quantile regression.
all all of the above (the default).

e_order(#) specifies the order of the polynomial in the predictors used to estimate conditional moments of the out-of-sample error. Default is **e_order(1).** If there is risk of over-fitting the option is automatically set to 0 (see u_order for more information).

e_lags(#) specifies the lags of the predictors used to estimate conditional moments of the out-of-sample error. Default is **e_lags(0).** If there is risk of over-fitting the option is automatically set to 0 (see u_order for more information).

e_alpha(#) specifies the confidence level for out-of-sample uncertainty, i.e. the confidence level is 1 -e_alpha. Default is **e_alpha(0.05).**

Regularization

lgapp(*string*) selects the way local geometry is approximated in simulation. The options are "generalized" and "linear". The first one accommodates for possibly non-linear constraints, whilst the second one is valid with linear constraints only.

rho(#) specifies the regularizing parameter that imposes sparsity on the estimated vector of weights. If unspecified, the tuning parameter is computed based on optimization inequalities.

rho_max(#) specifies the maximum value attainable by the tuning parameter **rho.**

Others

opt_est(*string*) a string specifying the stopping criteria used by the underling optimizer (nlopt) for point estimation. The default is a sequential quadratic programming (SQP) algorithm for nonlinearly constrained gradient-based optimization ('SLSQP'). In case a lasso-type constraint is implemented, the method of moving asymptotes (MMA) is used. The default value is **opt("'maxeval' = 5000, 'xtol_rel' = 1e-8, 'xtol_abs' = 1e-8, 'ftol_rel' = 1e-12, 'ftol_abs' = 1e-12, 'tol_eq' = 1e-8, 'tol_ineq' = 1e-8").**

opt_inf(*string*) a string specifying the stopping criteria used by the underling optimizer (nlopt) for point estimation. The default is a sequential quadratic programming (SQP) algorithm for nonlinearly constrained gradient-based optimization ('SLSQP'). In case a lasso-type constraint is implemented, the method of moving asymptotes (MMA) is used. The default value is **opt("'maxeval' = 5000, 'xtol_rel' = 1e-8, 'xtol_abs' = 1e-8, 'ftol_rel' = 1e-4, 'ftol_abs' = 1e-4, 'tol_eq' = 1e-8, 'tol_ineq' = 1e-8")**.

pypinocheck) if specified avoids to check that the version of scpi_pkg in Python is the one required by **scpi** in Stata. When not specified performs the check and stores a macro called to avoid checking it multiple times.

Example: Germany Data

Setup
. use scpi_germany.dta

Prepare data
. scdata gdp, dfname("python_scdata") id(country) outcome(gdp) time(year) treatment(status) cointegrated

Estimate Synthetic Control with a simplex constraint and quantify uncertainty
. scpi, dfname("python_scdata") name(simplex) u_missp

marker stored_results}**Stored results**

scpi stores the following in **e()**:

Scalars

e(KMI)	number of covariates used for adjustment
e(I)	number of treated units

Macros

e(outcomevar)	name of outcome variable
e(features)	name of features
e(constant)	logical indicating the presence of a common constant across features
e(anticipation)	logical indicating the extent of anticipation effects
e(donors)	donor units for each treated unit
e(cointegrated_data)	logical indicating cointegration
e(p)	type of norm of the weights used in constrained estimation
e(dir)	direction of the constraint on the norm of the weights
e(name)	name of constraint used in estimation
e(u_missp)	a logical indicating whether the model has been treated as misspecified or not
e(u_order)	order of the polynomial in the predictors used to estimate conditional moments of the pseudo-residuals
e(u_lags)	lags of the predictors used to estimate conditional moments of the pseudo-residuals
e(u_sigma)	estimator of the conditional variance-covariance matrix

of the pseudo-residuals

e(u_alpha)	confidence level for in-sample uncertainty
e(e_method)	method used to quantify out-of-sample uncertainty
e(e_order)	order of the polynomial in the predictors used to estimate conditional moments of the out-of-sample error
e(e_lags)	order of the polynomial in the predictors used to estimate conditional moments of the out-of-sample error
e(e_alpha)	confidence level for out-of-sample uncertainty

Matrices

e(A)	pre-treatment features of the treated unit
e(B)	pre-treatment features of the control units
e(C)	covariates used for adjustment
e(P)	covariates used to predict the out-of-sample series for the synthetic unit
e(pred)	predicted values of the features of the treated unit
e(res)	residuals **e(A)** - **e(pred)**
e(w)	weights of the controls
e(r)	coefficients of the covariates used for adjustment
e(beta)	stacked version of **e(w)** and **e(r)**
e(Y_pre)	pre-treatment outcome of the treated unit (only returned if one treated unit)
e(Y_post)	post-treatment outcome of the treated unit (only returned if one treated unit)
e(Y_pre_fit)	estimate pre-treatment outcome of the treated unit
e(Y_post_fit)	estimated post-treatment outcome of the treated unit
e(T0)	number of pre-treatment periods per feature for each treated unit
e(M)	number of features for each treated unit
e(KM)	number of covariates used for adjustment for each treated unit
e(J)	number of donors for each treated unit
e(T1)	number of post-treatment periods for each treated unit
e(Qstar)	regularized constraint on the norm
e(rho)	estimated regularizing parameter that imposes sparsity on the estimated vector of weights
e(CI_in_sample)	prediction intervals taking only in-sample uncertainty into account
e(CI_all_gaussian)	prediction intervals taking in- and out-of-sample uncertainty into account
e(CI_all_ls)	prediction intervals taking in- and out-of-sample uncertainty into account
e(CI_all_qreg)	prediction intervals taking in- and out-of-sample uncertainty into account
e(u_mean)	estimated conditional mean of the pseudo-residuals
e(u_var)	estimated conditional variance-covariance of the pseudo-residuals
e(e_mean)	estimated conditional mean of the out-of-sample error
e(e_var)	estimated conditional variance of the out-of-sample error
e(failed_sims)	percentage of failed simulations per post-treatment period to estimate lower and upper bounds.

References

Abadie, A. 2021. Using synthetic controls: Feasibility, data requirements, and methodological aspects. *Journal of Economic Literature*, 59(2), 391–425.

Cattaneo, M. D., Feng, Y., and Titiunik, R. 2021. Prediction Intervals for Synthetic Sontrol Methods. *Journal of the American Statistical Association*, 116(536), 1865–1880.

Cattaneo, M. D., Feng, Y., Palomba F., and Titiunik, R. 2022. scpi: Uncertainty Quantification for Synthetic Control Estimators, *arXiv*:2202.05984.

Cattaneo, M. D., Feng, Y., Palomba F., and Titiunik, R. 2022. Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption, *arXiv*:2210.05026.

Authors

Matias D. Cattaneo, Princeton University, Princeton, NJ. cattaneo@princeton.edu.

Yingjie Feng, Tsinghua University, Beijing, China. fengyj@sem.tsinghua.edu.cn.

Filippo Palomba, Princeton University, Princeton, NJ. fpalomba@princeton.edu.

Rocio Titiunik, Princeton University, Princeton, NJ. titiunik@princeton.edu.