Stratified Randomized Experiments (Python™ Edition)

The `sreg` package offers a toolkit for estimating average treatment effects (ATEs) in stratified randomized experiments. The package is designed to accommodate scenarios with multiple treatments and cluster-level treatment assignments, and accomodates optimal linear covariate adjustment based on baseline observable characteristics. The package computes estimators and standard errors based on Bugni, Canay, Shaikh (2018); Bugni, Canay, Shaikh, Tabord-Meehan (2023); and Jiang, Linton, Tang, Zhang (2023).

Dependencies: `numpy`, `pandas`, `scipy`

Authors

- Juri Trifonov jutrifonov@uchicago.edu
- Yuehao Bai yuehao.bai@usc.edu
- Azeem Shaikh amshaikh@uchicago.edu
- Max Tabord-Meehan maxtm@uchicago.edu

Supplementary files

- Sketch of the derivation of the ATE variance estimator under cluster-level treatment assignment: Download PDF
- Expressions for the multiple treatment case (with and without clusters): Download PDF

Installation

The official released version can be installed from `PyPI`.

```
pip install sreg
```

The latest development version can be installed from `GitHub`.

```
pip install git+https://github.com/jutrifonov/sreg.py
```

The function `sreg()`

Estimates the ATE(s) and the corresponding standard error(s) for a (collection of) treatment(s) relative to a control.

Syntax

```
sreg(Y, S = None, D = None, G_id = None, Ng = None, X = None, HC1 = True)
```

Arguments

- **`Y (float)`** - a `numpy.array` of the observed outcomes;
- **`S (int)`** - a `numpy.array` of strata indicators $\{0, 1, 2, \ldots\}$; if `None` then the estimation is performed assuming no stratification;
- **`D (int)`** - a `numpy.array` of treatments indexed by $\{0, 1, 2, \ldots\}$, where `D = 0` denotes the control;
- **`G_id (int)`** - a `numpy.array` of cluster indicators; if `None` then estimation is performed assuming treatment is assigned at the individual level;
- **`Ng (int)`** - a `numpy.array` of cluster sizes; if `None` then `Ng` is assumed to be equal to the number of available observations in every cluster;
- **`X (DataFrame)`** - a `pandas.DataFrame` with columns representing the covariate values for every observation; if `None` then the estimator without linear adjustments is applied [^*];
- **`HC1 (bool)`** - a `True/False` logical argument indicating whether the small sample correction should be applied to the variance estimator.

  [^*]: *Note: sreg cannot use individual-level covariates for covariate adjustment in cluster-randomized experiments. Any individual-level covariates will be aggregated to their cluster-level averages.*

Data Structure

Here we provide an example of a data frame that can be used with `sreg`.

Y	S	D	G_id	Ng	X1	X2
-0.57773576	2	0	1	10	1.5597899	0.03023334
1.69495638	2	0	1	10	1.5597899	0.03023334
2.02033740	4	2	2	30	0.8747419	-0.77090031
1.22020493	4	2	2	30	0.8747419	-0.77090031
1.64466086	4	2	2	30	0.8747419	-0.77090031
-0.32365109	4	2	2	30	0.8747419	-0.77090031
2.21008191	4	2	2	30	0.8747419	-0.77090031
-2.25064316	4	2	2	30	0.8747419	-0.77090031
0.37962312	4	2	2	30	0.8747419	-0.77090031

Summary

`sreg` prints a *"Stata-style"* table containing the ATE estimates, corresponding standard errors, t-statistics, p-values, 95% asymptotic confidence intervals, and significance indicators for different levels α. The example of the printed output is provided below.

```
Saturated Model Estimation Results under CAR with clusters
Observations: 24680
Clusters: 1000
Number of treatments: 2
Number of strata: 5
---
Coefficients:
     Tau   As.se   T-stat  P-value  CI.left(95%)  CI.right(95%)
Significance
-0.03836 0.09008 -0.42585  0.67021      -0.21491        0.13819
 0.80719 0.09096  8.87453  0.00000       0.62892        0.98546
***
---
Signif. codes:  0 `***` 0.001 `**` 0.01 `*` 0.05 `.` 0.1 ` ` 1
```

Return Value

Returns an object of class `Sreg` that is a dictionary containing the following elements:

- **`tau_hat`** - a numpy array of shape $(1, |A|)$ containing the ATE estimates, where $|A|$ represents the number of treatments;
- **`se_rob`** - a numpy array of shape $(1, |A|)$ containing the standard error estimates, where $|A|$ represents the number of treatments;
- **`t_stat`** - a numpy array of shape $(1, |A|)$ containing the t-statistics, where $|A|$ represents the number of treatments;
- **`p_value`** - a numpy array of shape $(1, |A|)$ containing the corresponding p-values, where $|A|$ represents the number of treatments;
- **`CI_left`** - a numpy array of shape $(1, |A|)$ containing the left bounds of the 95% as. confidence interval;
- **`CI_right`** - a numpy array of shape $(1, |A|)$ containing the right bounds of the 95% as. confidence interval;
- **`data`** - the original data provided, stored as a `pandas DataFrame` with the columns `[Y, S, D, G_id, Ng, X]`;
- **`lin_adj`** - a `pandas DataFrame` representing the covariates that were used in implementing linear adjustments.

Empirical Example

Here, we provide the empirical application example using the data from (Chong et al., 2016), who studied the effect of iron deficiency anemia on school-age children's educational attainment and cognitive ability in Peru. The example replicates the empirical illustration from (Bugni et al., 2019). For replication purposes, the data is included in the package and can be accessed by running `AEJapp()`.

```
from sreg import sreg, sreg_rgen, AEJapp
```

We can upload the `AEJapp` dataset to the `Python` session via `AEJapp` function:

```
data = AEJapp()
```

It is pretty straightforward to prepare the data to fit the package syntax:

```
Y = data['gradesq34']
D = data['treatment']
S = data['class_level']
pills = data['pills_taken']
age = data['age_months']
data_clean = pd.DataFrame({'Y': Y, 'D': D, 'S': S, 'pills': pills,
'age': age})

data_clean['D'] = data_clean['D'].apply(lambda x: 0 if x == 3 else x)

Y = data_clean['Y']
D = data_clean['D']
S = data_clean['S']
pills = data_clean['pills']
age = data_clean['age']
X = data_clean[['pills', 'age']]
```

We can take a look at the frequency table of `D` and `S`:

```
contingency_table = pd.crosstab(data_clean['D'], data_clean['S'])
print(contingency_table)
S   1   2   3   4   5
D
0  15  19  16  12  10
1  16  19  15  10  10
2  17  20  15  11  10
```

Now, it is straightforward to replicate the results from (Bugni et al, 2019) using `sreg`:

```
result = sreg(Y = Y, S = S, D = D, G_id = None, Ng = None, X = None, HC1
= True)
print(result)
```

```
Saturated Model Estimation Results under CAR
Observations: 215
Number of treatments: 2
Number of strata: 5
---
Coefficients:
     Tau   As.se   T-stat  P-value  CI.left(95%)  CI.right(95%)
Significance
-0.05113 0.20645 -0.24766  0.80440      -0.45577        0.35351
 0.40903 0.20651  1.98065  0.04763       0.00427        0.81379
*
---
Signif. codes:  0 `***` 0.001 `**` 0.01 `*` 0.05 `.` 0.1 ` ` 1
```

Besides that, `sreg` allows adding linear adjustments (covariates) to the estimation procedure:

```
result = sreg(Y = Y, S = S, D = D, G_id = None, Ng = None, X = X, HC1 =
True)
print(result)
Saturated Model Estimation Results under CAR with linear adjustments
Observations: 215
Number of treatments: 2
Number of strata: 5
Covariates used in linear adjustments: pills, age
---
Coefficients:
     Tau   As.se   T-stat  P-value  CI.left(95%)  CI.right(95%)
Significance
-0.02862 0.17964 -0.15929  0.87344      -0.38071        0.32348
 0.34609 0.18362  1.88477  0.05946      -0.01381        0.70598
.
---
Signif. codes:  0 `***` 0.001 `**` 0.01 `*` 0.05 `.` 0.1 ` ` 1
```

The function `sreg_rgen()`

Generates the observed outcomes, treatment assignments, strata indicators, cluster indicators, cluster sizes, and covariates for estimating the treatment effect following the stratified block randomization design under covariate-adaptive randomization (CAR).

Syntax

```
sreg_rgen(n, Nmax = 50, n_strata = 5,
          tau_vec = [0], gamma_vec = [0.4, 0.2, 1],
          cluster = True, is_cov = True)
```

Arguments

- **`n (int)`** - The total number of observations in the sample;
- **`Nmax (int)`** - The maximum size of generated clusters (maximum number of observations in a cluster);
- **`n_strata (int)`** - An `integer` specifying the number of strata;
- **`tau_vec (list of float)`** - A `list` of treatment effects of length |A|, where |A| represents the number of treatments;
- **`gamma_vec (list of float)`** - A `list` of three parameters corresponding to covariates;
- **`cluster (bool)`** - A `boolean` indicating whether the data generation process (DGP) should use cluster-level treatment assignment (`True`) or individual-level treatment assignment (`False`);
- **`is.cov (bool)`** - A `boolean` indicating whether the DGP should include covariates (`True`) or not (`False`).

Return Value

`pd.DataFrame`: A `DataFrame` with `n` observations containing the generated values of the following variables:

- **`Y (pd.Series of float)`** - A numeric Series of length `n` representing the observed outcomes;
- **`S (pd.Series of int)`** - A numeric Series of length `n` representing the strata indicators;
- **`D (pd.Series of int)`** - A numeric Series of length `n` representing the treatment assignments, indexed by ${0, 1, 2, ...}$, where `D = 0` denotes the control group;
- **`G_id (pd.Series of int)`** - A numeric Series of length `n` representing the cluster indicators;
- **`Ng (pd.DataFrame)`** - A numeric Series of length `n` representing the cluster indicators;
- **`X (pd.DataFrame)`** - A `DataFrame` with columns representing the covariate values for every observation.

Example

```
from sreg import sreg_rgen
data = sreg_rgen(n = 1000, tau_vec = [0, 0.8], cluster = False, is_cov =
True)
print(data)
            Y  S  D        X1        X2
0    4.689501  1  0  7.120830  3.792420
1    3.629002  3  2  3.234888  1.674029
2    0.739461  3  1  4.822114  1.165004
3    0.292031  4  0  4.360900  1.171521
4    0.755504  5  1  6.417946  0.176026
..        ... .. ..       ...       ...
```

```
995  1.732812  4  0  3.695054  0.866644
996  2.529121  4  0  5.449032  1.639192
997  2.174121  3  1  4.929872  0.566262
998  2.649385  3  0  3.535942  1.995133
999  4.868684  2  2  7.111149  1.646865
```

References

Bugni, F. A., Canay, I. A., and Shaikh, A. M. (2018). Inference Under Covariate-Adaptive Randomization. *Journal of the American Statistical Association*, 113(524), 1784–1796, doi:10.1080/01621459.2017.1375934.

Bugni, F., Canay, I., Shaikh, A., and Tabord-Meehan, M. (2024+). Inference for Cluster Randomized Experiments with Non-ignorable Cluster Sizes. *Forthcoming in the Journal of Political Economy: Microeconomics*, doi:10.48550/arXiv.2204.08356.

Jiang, L., Linton, O. B., Tang, H., and Zhang, Y. (2023+). Improving Estimation Efficiency via Regression-Adjustment in Covariate-Adaptive Randomizations with Imperfect Compliance. *Forthcoming in Review of Economics and Statistics*, doi:10.48550/arXiv.2204.08356.