

---

# **causalcp Documentation**

***Release 0.2.0***

**Juan L Gamella**

**May 12, 2021**



**CONTENTS:**

<b>1</b>	<b>Navigating this documentation</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Versioning</b>	<b>7</b>
<b>4</b>	<b>License</b>	<b>9</b>
<b>5</b>	<b>Feedback</b>	<b>11</b>
5.1	causalicp.fit . . . . .	11
5.2	causalicp.Result . . . . .	14
	<b>Index</b>	<b>17</b>



This is a Python implementation of the Invariant Causal Prediction (ICP) algorithm from the 2016 [paper](#) “*Causal inference using invariant prediction: identification and confidence intervals*” by Jonas Peters, Peter Bühlmann and Nicolai Meinshausen.

Currently, only the faster Method II of the paper (t-test + F-test on residuals) is implemented. For more details, see Section 3.1.2 of the paper.



## NAVIGATING THIS DOCUMENTATION

To run the algorithm, see the function `causalicp.fit()`. The results of the computation are reported through the `causalicp.Result` class.





## INSTALLATION

You can clone this repo or install the python package via pip:

```
pip install causalicp
```

The code has been written with an emphasis on readability and on keeping the dependency footprint to a minimum; to this end, the only dependencies outside the standard library are `numpy`, `scipy` and `termcolor`.



## VERSIONING

The package is still at its infancy and its API is subject to change. However, this will be done with care: non backward-compatible changes to the API are reflected by a change to the minor or major version number,

*e.g. code written using `causalcp==0.1.2` will run with `causalcp==0.1.3`, but may not run with `causalcp==0.2.0`.*



## **LICENSE**

The implementation is open-source and shared under a BSD 3-Clause License. You can find the source code in the [GitHub repository](#).



## FEEDBACK

Feedback is most welcome! You can add an issue in the [repository](#) or send an [email](#).

## 5.1 causalicp.fit

To run the algorithm, the function `causalicp.fit()` is provided. The result of the computation is given in a `causalicp.Result` object containing the estimate, accepted sets, p-values, etc.

`causalicp.fit(data, target, alpha=0.05, sets=None, precompute=True, verbose=False, color=True)`

Run Invariant Causal Prediction on data from different experimental settings.

### Parameters

- **data** (*numpy.ndarray or list of array-like*) – The data from all experimental settings. Each element of the list/array is a 2-dimensional array with a sample from a different setting, where columns correspond to variables and rows to observations (data-points). The data also contains the response variable, which is specified with the *target* parameter.
- **target** (*int*) – The index of the response or target variable of interest.
- **alpha** (*float, default=0.05*) – The level of the test procedure, taken from  $[0,1]$ . Defaults to 0.05.
- **sets** (*list of set or None, default=None*) – The sets for which ICP will test invariance. An error is raised if a set is not a subset of  $\{0, \dots, p-1\}$  or it contains the target, where  $p$  is the total number of variables (including the target). If *None* all possible subsets of predictors will be considered.
- **precompute** (*bool, default=True*) – Whether to precompute the sample covariance matrix to speed up linear regression during the testing of each predictor set. For large sample sizes this drastically reduces the overall execution time, but it may result in numerical instabilities for highly correlated data. If set to *False*, for each set of predictors the regression is done using an iterative least-squares solver on the raw data.
- **verbose** (*bool, default=False*) – If ICP should run in verbose mode, i.e. displaying information about completion and the result of tests.
- **color** (*bool, default=True*) – If the output produced when *verbose=True* should be color encoded (not recommended if your terminal does not support ANSI color formatting), see [termcolor](#).

### Raises

- **ValueError** : – If the value of some of the parameters is not appropriate, e.g. *alpha* is negative, *data* contains samples with different number of variables, or *sets* contains invalid sets.

- **TypeError** : – If the type of some of the parameters was not expected (see examples below).

**Returns** **result** – A `causalicp.Result` object containing the result of running ICP, i.e. estimate, accepted sets, p-values, etc.

**Return type** `causalicp.Result`

## Example

Using interventional from a linear-gaussian SCM (generated using `sampler`)

```
>>> data = [np.array([[0.46274901, -0.19975643, 0.76993618, 2.65949677],
...                  [0.3749258, -0.98625196, -0.1806925, 1.23991796],
...                  [-0.39597772, -1.79540294, -0.39718702, -1.31775062],
...                  [2.39332284, -3.22549743, 0.15317657, 1.60679175],
...                  [-0.56982823, 0.5084231, 0.41380479, 1.19607095]]),
...         np.array([[1.45648798, 8.29977262, 1.05992289, 7.49191164],
...                  [-1.35654212, 13.59077259, -1.14624494, 5.76580633],
...                  [-0.48800913, 11.15112687, 0.48421499, 7.20695569],
...                  [2.74901219, 8.82465628, 1.49619723, 12.48016441],
...                  [5.35033726, 12.91847915, 1.69812062, 19.40468998]]),
...         np.array([[-11.73619893, -6.87502658, -6.71775898, -28.2782561],
...                  [-16.24118216, -11.26774231, -9.22041168, -42.09076079],
...                  [-14.85266731, -11.02688079, -8.71264951, -40.37471919],
...                  [-16.08519052, -11.73497156, -10.58198058, -42.55646184],
...                  [-17.07817707, -11.29005529, -10.04063011, -45.01702447]])]
```

Running ICP for the response variable 3, at a significance level of 0.05.

```
>>> import causalicp as icp
>>> result = icp.fit(data, 3, alpha=0.05, precompute=True, verbose=True,
↳color=False)
Tested sets and their p-values:
set() rejected : 6.8529852769059795e-06
{0} rejected : 0.043550405609324994
{1} rejected : 6.10963528362226e-06
{2} rejected : 0.009731028782704005
{0, 1} accepted : 0.9107055098714101
{0, 2} rejected : 0.004160395025223608
{1, 2} accepted : 1
{0, 1, 2} accepted : 1
Estimated parental set: {1}
```

Obtaining the estimate, accepted sets, etc

```
>>> result.estimate
{1}
```

```
>>> result.accepted_sets
[{0, 1}, {1, 2}, {0, 1, 2}]
```

```
>>> result.rejected_sets
[set(), {0}, {1}, {2}, {0, 2}]
```



```
>>> result.pvalues
{0: 1, 1: 0.043550405609324994, 2: 0.9107055098714101, 3: nan}
```

```
>>> result.conf_intervals
array([[0.          , 0.37617783, 0.          ,      nan],
       [2.3531227 , 0.89116407, 4.25277329,      nan]])
```

### Examples of exceptions

A *TypeError* is raised for parameters of the wrong type, and *ValueError* if they are not valid. For example, if *alpha* is not a float between 0 and 1,

```
>>> icp.fit(data, 3, alpha = 1)
Traceback (most recent call last):
...
TypeError: alpha must be a float, not <class 'int'>.
```

```
>>> icp.fit(data, 3, alpha = -0.1)
Traceback (most recent call last):
...
ValueError: alpha must be in [0,1].
```

```
>>> icp.fit(data, 3, alpha = 1.1)
Traceback (most recent call last):
...
ValueError: alpha must be in [0,1].
```

if the target is not an integer within range,

```
>>> icp.fit(data, 3.0)
Traceback (most recent call last):
...
TypeError: target must be an int, not <class 'float'>.
```

```
>>> icp.fit(data, 5)
Traceback (most recent call last):
...
ValueError: target must be an integer in [0, p-1].
```

if *sets* is of the wrong type or contains an invalid set,

```
>>> icp.fit(data, 3, sets = [{2}, {1,3}])
Traceback (most recent call last):
...
ValueError: Set {1, 3} in sets is not valid: it must be a subset of {0,...,p-1} -
↪ {target}.
```

```
>>> icp.fit(data, 3, sets = ({2}, {0,1}))
Traceback (most recent call last):
...
TypeError: sets must be a list of set, not <class 'tuple'>.
```

```
>>> icp.fit(data, 3, sets = [(2,), (0,1)])
Traceback (most recent call last):
...
TypeError: sets must be a list of set, not of <class 'tuple'>.
```

if *precompute*, *verbose* or *color* are not of type *bool*,

```
>>> icp.fit(data, 3, precompute=1)
Traceback (most recent call last):
...
TypeError: precompute must be bool, not <class 'int'>.
```

```
>>> icp.fit(data, 3, verbose=1)
Traceback (most recent call last):
...
TypeError: verbose must be bool, not <class 'int'>.
```

```
>>> icp.fit(data, 3, color=1)
Traceback (most recent call last):
...
TypeError: color must be bool, not <class 'int'>.
```

or if the samples from each experimental setting have different numbers of variables,

```
>>> data = [[[0.01, 0.02],[0.03,0.04]], [[0.01],[0.03]]]
>>> icp.fit(data, 3)
Traceback (most recent call last):
...
ValueError: The samples from each setting have a different number of variables: [2,
↪ 1].
```

## 5.2 causalicp.Result

The result of running `causalicp.fit()` is returned in a `causalicp.Result` object, which contains the estimate, accepted sets, p-values, etc.

**class** `causalicp.Result`(*target*, *data*, *estimate*, *accepted*, *rejected*, *conf\_intervals*, *set\_pvalues*, *set\_coefs*)

The result of running Invariant Causal Prediction, produced as output of `causalicp.fit()`.

**p**

The total number of variables in the data (including the response/target).

**Type** int

**target**

The index of the response/target.

**Type** int

**estimate**

The estimated parental set returned by ICP, or *None* if all sets of predictors were rejected.

**Type** set or None

**accepted\_sets**

A list containing the accepted sets of predictors.

**Type** list of set

**rejected\_sets**

A list containing the rejected sets of predictors.

**Type** list of set

**pvalues**

A dictionary containing the p-value for the causal effect of each individual predictor. The target/response is included in the dictionary and has value *nan*.

**Type** dict of (int, float)

**conf\_intervals**

A  $2 \times p$  array of floats representing the confidence interval for the causal effect of each variable. Each column corresponds to a variable, and the first and second row correspond to the lower and upper limit of the interval, respectively. The column corresponding to the target/response is set to *nan*.

**Type** numpy.ndarray or None

**Example**

```
>>> import causalicp as icp
>>> result = icp.fit(data, 3)
```

```
>>> result.p
4
```

```
>>> result.target
3
```

```
>>> result.estimate
set()
```

```
>>> result.accepted_sets
[{1}, {2}, {0, 1}, {1, 2}, {0, 1, 2}]
```

```
>>> result.rejected_sets
[set(), {0}, {0, 2}]
```

```
>>> result.pvalues
{0: 1, 1: 0.187430598304751, 2: 1, 3: nan}
```

```
>>> result.conf_intervals
array([[0., 0., 0., nan],
       [2.37257655, 1.95012059, 5.88760917, nan]])
```

When all sets are rejected (e.g. there is a model violation), the estimate and confidence intervals are set to *None*:

```
>>> result = icp.fit(data_bad_model, 3)
>>> result.estimate
>>> result.conf_intervals
```

And the individual p-value for the causal effect of each variable is set to 1:

```
>>> result.pvalues
{0: 1, 1: 1, 2: 1, 3: nan}
```

## INDEX

### A

`accepted_sets` (*causalicp.Result* attribute), [14](#)

### C

`conf_intervals` (*causalicp.Result* attribute), [15](#)

### E

`estimate` (*causalicp.Result* attribute), [14](#)

### F

`fit()` (*in module causalicp*), [11](#)

### P

`p` (*causalicp.Result* attribute), [14](#)

`pvalues` (*causalicp.Result* attribute), [15](#)

### R

`rejected_sets` (*causalicp.Result* attribute), [15](#)

`Result` (*class in causalicp*), [14](#)

### T

`target` (*causalicp.Result* attribute), [14](#)