Tigramite Documentation

Release 3.0

Jakob Runge

May 15, 2017

CONTENTS:

Tigramite is a causal time series analysis python package. With flexibly adaptable scripts it allows to reconstruct graphical models (conditional independence graphs) from discrete or continuously-valued time series based on a causal discovery algorithm and create high-quality plots of the results.

tigramite.pcmci.PCMCI(dataframe, cond_ind_test)	PCMCI: causal discovery for time series datasets.
tigramite.independence_tests.CondIndTest([...])	Base class of conditional independence tests.
tigramite.independence_tests.ParCorr(**kwargs)	Partial correlation test.
tigramite.independence_tests.GPACE([...])	GPACE conditional independence test.
tigramite.independence_tests.CMIknn([knn])	Conditional mutual information test based on nearest-neighbor estimator.
tigramite.independence_tests.CMIsymb(**kwargs)	Conditional mutual information test based on discrete estimator.
tigramite.data_processing	Tigramite data processing functions.
tigramite.plotting	Tigramite plotting package.

CHAPTER

ONE

TIGRAMITE.PCMCI: PCMCI

class tigramite.pcmci.**PCMCI**(*dataframe*, *cond_ind_test*, *selected_variables=None*, *var_names=None*, *verbosity=0*)

PCMCI: causal discovery for time series datasets.

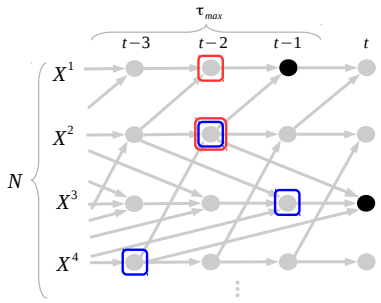
The implementation is based on Algorithms 1 and 2 in[1]

PCMCI is a 2-step causal discovery method for large-scale time series datasets. The first step is a condition-selection followed by the MCI conditional independence test.

PCMCI allows:

- different conditional independence test statistics adapted to continuously-valued or discrete data, and different assumptions about linear or nonlinear dependencies
- easy parallelization
- masked time series data
- false discovery control and confidence interval estimation

Notes



In our framework, the dependency structure of a set of time series variables is represented in a *time series graph* as shown in the Figure. The nodes of a time series graph are defined as the variables at different times and a link exists if two lagged variables are *not* conditionally independent given the past of the whole process. Assuming stationarity, the links are repeated in time. The parents $\mathcal{P}$ of a variable are defined as the set of all nodes with a link towards it.

Our causal discovery technique to estimate the time series graph is based on a two-step procedure:

1. Condition-selection: For all N variables: Estimate *superset* of parents :math:‘ ilde{mathcal{P}}(X^j_t)‘ with iterative PC1algorithm.

[1] J. Runge, D. Sejdinovic, S. Flaxman (2017): Detecting causal associations in large nonlinear time series datasets, https://arxiv.org/abs/1702.07007

2.*Momentary conditional independence* test

$$extMCI: \quad X^i_{t-au} \perp X^j_t \mid \mathcal{P}(X^j_t) \setminus \{X^i_{t-au}\}, \mathcal{P}(X^i_{t-au}),$$

The main free parameters of this method (in addition to free parameters of the conditional independence test statistic) are the maximum time delay :math:‘ au_{max}‘ and the significance threshold in the PC1 step $lpha$. The maximum time delay depends on the application and should be chosen according to the maximum physical time lag expected in the complex system. $lpha$ should not be seen as a significance test level in PC1 since the iterative hypothesis tests do not allow for a precise confidence level. $lpha$ rather takes the role of a regularization parameter in model-selection techniques. The conditioning sets $\mathcal{P}$ estimated with PC1 should include the true parents and at the same time be small in size to reduce the estimation dimension of the MCI test and improve its power. But the first demand is typically more important. If a list of values is given or `pc_alpha=None`, this parameter is optimized using model selection

Further minor parameters are discussed in[1].

Parameters

- **dataframe** (*data object*) – This can either be the tigramite dataframe object or a pandas data frame. It must have the attributes dataframe.values yielding a numpy array of shape (observations T, variables N) and optionally a mask of the same shape.
- **cond_ind_test** (*conditional independence test object*) – This can be ParCorr or other classes from the tigramite package or an external test passed as a callable. This test can be based on the class tigramite.independence_tests.CondIndTest. If a callable is passed, it must have the signature:

```
class CondIndTest():
    # with attributes
    # * measure : str
    #   name of the test
    # * use_mask : bool
    #   whether the mask should be used

    # and functions
    # * run_test(X, Y, Z, tau_max) : where X,Y,Z are of the form
    #   X = [(var, -tau)]  for non-negative integers var and tau
    #   specifying the variable and time lag
    #   return (test statistic value, p-value)
    # * set_dataframe(dataframe) : set dataframe object

    # optionally also

    # * get_model_selection_criterion(j, parents) : required if
    #   pc_alpha parameter is to be optimized. Here j is the
    #   variable index and parents a list [(var, -tau), ...]
    #   return score for model selection
    # * get_confidence(X, Y, Z, tau_max) : required for
    #   return_confidence=True
    #   estimate confidence interval after run_test was called
    #   return (lower bound, upper bound)
```

- **selected_variables** (*list of integers, optional (default: range(N))*) – Specify to estimate parents only for selected variables. If None is passed, parents are estimated for all variables.
- **var_names** (*list of strings, optional (default: range(N))*) – Names of variables, must match the number of variables. If None is passed, variables are enumerated as [0, 1, ...]

- **verbosity** (*int, optional (default: 0)*) – Verbose levels 0, 1, ...

all_parents
dictionary – Dictionary of form {0:[(0, -1), (3, -2), ...], 1:[], ...} containing the conditioning-parents estimated with PC algorithm.

estimates
dictionary – Dictionary of form estimates[j][(i, -tau)] = {'min_val': float, 'max_pval': float,...} containing the minimum test statistic value and maximum p-value for each link estimated in the PC algorithm.

iterations
dictionary – Dictionary containing further information on algorithm steps.

N
int – Number of variables.

T
int – Time series sample length.

References

get_corrected_pvalues(*tau_max*, *p_matrix*, *fdr_method='fdr_bh'*, *exclude_contemporaneous=True*)
Returns p-values corrected for multiple testing.

Wrapper around statsmodels.sandbox.stats.multicomp.multipletests. Corrections is performed either among all links if exclude_contemporaneous==False, or only among lagged links.

Parameters

- **tau_max** (*int*) – Maximum time lag.
- **p_matrix** (*array-like*) – Matrix of p-values. Must be of shape (N, N, tau_max + 1).
- **fdr_method** (*str, optional (default: 'fdr_bh')*) – Correction method, default is Benjamini-Hochberg False Discovery Rate method.
- **exclude_contemporaneous** (*bool, optional (default: True)*) – Whether to include contemporaneous links in correction.

Returns **q_matrix** – Matrix of shape (N, N, tau_max + 1) containing corrected p-values.

Return type array-like

get_lagged_dependencies(*selected_links=None*, *tau_min=0*, *tau_max=1*, *parents=None*, *max_conds_py=None*, *max_conds_px=None*)
Returns matrix of lagged dependence measure values.

Parameters

- **selected_links** (*dict or None*) – Dictionary of form {0:[(0, -1), (3, -2), ...], 1:[], ...} specifying whether only selected links should be tested. If None is passed, all links are tested
- **tau_min** (*int, default: 1*) – Minimum time lag to test. Useful for multi-step ahead predictions.
- **tau_max** (*int, default: 1*) – Maximum time lag. Must be larger or equal to tau_min.
- **parents** (*dict or None*) – Dictionary of form {0:[(0, -1), (3, -2), ...], 1:[], ...} specifying the conditions for each variable. If None is passed, no conditions are used.

- **max_conds_py** (*int or None*) – Maximum number of conditions of Y to use. If None is passed, this number is unrestricted.
- **max_conds_px** (*int or None*) – Maximum number of conditions of Z to use. If None is passed, this number is unrestricted.

Returns **val_matrix** – The matrix of shape (N, N, tau_max+1) containing the lagged dependencies.

Return type array

run_mci(*selected_links=None*, *tau_min=0*, *tau_max=1*, *parents=None*, *max_conds_py=None*, *max_conds_px=None*)

MCI conditional independence tests.

Implements the MCI test (Algorithm 2 in[1]). Returns the matrices of test statistic values, p-values, and confidence intervals.

Parameters

- **selected_links** (*dict or None*) – Dictionary of form {0:all_parents (3, -2), ...], 1:[], ...} specifying whether only selected links should be tested. If None is passed, all links are tested
- **tau_min** (*int, default: 1*) – Minimum time lag to test. Useful for multi-step ahead predictions.
- **tau_max** (*int, default: 1*) – Maximum time lag. Must be larger or equal to tau_min.
- **parents** (*dict or None*) – Dictionary of form {0:[(0, -1), (3, -2), ...], 1:[], ...} specifying the conditions for each variable. If None is passed, no conditions are used.
- **max_conds_py** (*int or None*) – Maximum number of conditions of Y to use. If None is passed, this number is unrestricted.
- **max_conds_px** (*int or None*) – Maximum number of conditions of Z to use. If None is passed, this number is unrestricted.

Returns **(val_matrix, p_matrix, conf_matrix)** – The matrices val_matrix and p_matrix are of shape (N, N, tau_max+1) and the matrix conf_matrix is of shape (N, N, tau_max+1, 2)

Return type tuple of arrays

run_pc_stable(*selected_links=None*, *tau_min=1*, *tau_max=1*, *save_iterations=False*, *pc_alpha=0.1*, *max_conds_dim=None*, *max_combinations=1*)

PC algorithm for estimating parents of all variables.

Parents are made available as self.all_parents

Parameters

- **selected_links** (*dict or None*) – Dictionary of form {0:[(0, -1), (3, -2), ...], 1:[], ...} specifying whether only selected links should be tested. If None is passed, all links are tested
- **tau_min** (*int, default: 1*) – Minimum time lag to test. Useful for multi-step ahead predictions.
- **tau_max** (*int, default: 1*) – Maximum time lag. Must be larger or equal to tau_min.
- **save_iterations** (*bool, default: False*) – Whether to save iteration step results such as conditions used.

- **pc_alpha** (*float or list of floats, default: 0.1*) – Significance level in algorithm. If a list is passed, then the pc_alpha level is optimized for every variable across the given pc_alpha values using the score computed in cond_ind_test.get_model_selection_criterion()
- **max_conds_dim** (*int or None*) – Maximum number of conditions to test. If None is passed, this number is unrestricted.
- **max_combinations** (*int, default: 1*) – Maximum number of combinations of conditions of current cardinality to test. Defaults to 1 for PC_1 algorithm. For original PC algorithm a larger number, such as 10, can be used.

Returns **all_parents** – Dictionary of form {0:[(0, -1), (3, -2), ...], 1:[], ...} containing estimated parents.

Return type dict

run_pcmci(*selected_links=None*, *tau_min=1*, *tau_max=1*, *save_iterations=False*, *pc_alpha=0.05*, *max_conds_dim=None*, *max_combinations=1*, *max_conds_py=None*, *max_conds_px=None*, *fdr_method='none'*)

Run full PCMCI causal discovery for time series datasets.

Wrapper around PC-algorithm function and MCI function.

Parameters

- **selected_links** (*list or None*) – List of form [(0, -1), (3, -2), ...] specifying whether only selected links should be tested. If None is passed, all links are tested
- **tau_min** (*int, default: 1*) – Minimum time lag to test. Useful for multi-step ahead predictions.
- **tau_max** (*int, default: 1*) – Maximum time lag. Must be larger or equal to tau_min.
- **save_iterations** (*bool, default: False*) – Whether to save iteration step results such as conditions used.
- **pc_alpha** (*float, default: 0.1*) – Significance level in algorithm.
- **max_conds_dim** (*int or None*) – Maximum number of conditions to test. If None is passed, this number is unrestricted.
- **max_combinations** (*int, default: 1*) – Maximum number of combinations of conditions of current cardinality to test. Defaults to 1 for PC_1 algorithm. For original PC algorithm a larger number, such as 10, can be used.
- **max_conds_py** (*int or None*) – Maximum number of conditions of Y to use. If None is passed, this number is unrestricted.
- **max_conds_px** (*int or None*) – Maximum number of conditions of Z to use. If None is passed, this number is unrestricted.
- **fdr_method** (*str, optional (default: 'fdr_bh')*) – Correction method, default is Benjamini-Hochberg False Discovery Rate method.

Returns **(val_matrix, p_matrix, conf_matrix)** – The matrices val_matrix and p_matrix are of shape (N, N, tau_max+1) and the matrix conf_matrix is of shape (N, N, tau_max+1, 2)

Return type tuple of arrays

CHAPTER

TWO

TIGRAMITE.INDEPENDENCE_TESTS: CONDITIONAL INDEPENDENCE TESTS

Base class:

class tigramite.independence_tests.**CondIndTest**(*use_mask=False*, *mask_type=None*, *significance='analytic'*, *fixed_thres=0.1*, *sig_samples=100*, *sig_blocklength=None*, *confidence=False*, *conf_lev=0.9*, *conf_samples=100*, *conf_blocklength=None*, *recycle_residuals=False*, *verbosity=0*)

Base class of conditional independence tests.

Provides useful general functions for different independence tests such as shuffle significance testing and bootstrap confidence estimation. Also handles masked samples. Other test classes can inherit from this class.

Parameters

- **use_mask** (*bool, optional (default: False)*) – Whether a supplied mask should be used.
- **mask_type** (*list including all or some of the strings 'x', 'y', 'z',*) – optional (default: None) Marks for which variables in the dependence measure I(X; Y | Z) the samples should be masked. If None, ['x', 'y', 'z'] is used, which excludes masked samples in X, Y, and Z and can be used for missing values. Explained in[1].
- **significance**(*str, optional (default: 'analytic')*) – Type of significance test to use. In this package 'analytic', 'fixed_thres' and 'shuffle_test' are available.
- **fixed_thres** (*float, optional (default: 0.1)*) – If significance is 'fixed_thres', this specifies the threshold for the absolute value of the dependence measure.
- **sig_samples** (*int, optional (default: 100)*) – Number of samples for shuffle significance test.
- **sig_blocklength** (*int, optional (default: None)*) – Block length for block-shuffle significance test. If None, the block length is determined from the decay of the autocovariance as explained in[1].
- **confidence** (*False or str, optional (default: False)*) – Specify type of confidence estimation. If False, numpy.nan is returned. 'bootstrap' can be used with any test, for ParCorr also 'analytic' is implemented.
- **conf_lev**(*float, optional (default: 0.9)*) – Two-sided confidence interval.
- **conf_samples**(*int, optional (default: 100)*) – Number of samples for bootstrap.

- **conf_blocklength** (*int, optional (default: None)*) – Block length for block-bootstrap. If None, the block length is determined from the decay of the autocovariance as explained in[1].
- **recycle_residuals** (*bool, optional (default: False)*) – Specifies whether residuals should be stored. This may be faster, but can cost considerable memory.
- **verbosity** (*int, optional (default: 0)*) – Level of verbosity.

get_bootstrap_confidence(*array*, *xyz*, *dependence_measure*, *conf_samples=100*, *conf_blocklength=None*, *conf_lev=0.95*, *verbosity=0*)

Perform bootstrap confidence interval estimation.

With conf_blocklength > 1 or None a block-bootstrap is performed.

Parameters

- **array** (*array-like*) – data array with X, Y, Z in rows and observations in columns
- **xyz** (*array of ints*) – XYZ identifier array of shape (dim,).
- **dependence_measure** (*object*) – Dependence measure function must be of form dependence_measure(array, xyz) and return a numeric value
- **conf_lev** (*float, optional (default: 0.9)*) – Two-sided confidence interval.
- **conf_samples** (*int, optional (default: 100)*) – Number of samples for bootstrap.
- **conf_blocklength** (*int, optional (default: None)*) – Block length for block-bootstrap. If None, the block length is determined from the decay of the autocovariance as explained in[1].
- **verbosity** (*int, optional (default: 0)*) – Level of verbosity.

Returns **(conf_lower, conf_upper)** – Upper and lower confidence bound of confidence interval.

Return type Tuple of floats

get_confidence(*X*, *Y*, *Z=None*, *tau_max=0*)

Perform confidence interval estimation.

Calls the dependence measure and confidence test functions. The child classes can specify a function get_dependence_measure and get_analytic_confidence or get_bootstrap_confidence. If confidence is False, (numpy.nan, numpy.nan) is returned.

Parameters

- **Y, Z** (*X,*) – X,Y,Z are of the form [(var, -tau)], where var specifies the variable index and tau the time lag.
- **tau_max** (*int, optional (default: 0)*) – Maximum time lag. This may be used to make sure that estimates for different lags in X, Z, all have the same sample size.

Returns **(conf_lower, conf_upper)** – Upper and lower confidence bound of confidence interval.

Return type Tuple of floats

get_fixed_thres_significance(*value*, *fixed_thres*)

Returns signficance for thresholding test.

Returns 0 if numpy.abs(value) is smaller than fixed_thres and 1 else.

Parameters

- **value** (*number*) – Value of test statistic for unshuffled estimate.
- **fixed_thres** (*number*) – Fixed threshold, is made positive.

Returns **pval** – Returns 0 if numpy.abs(value) is smaller than fixed_thres and 1 else.

Return type bool

get_measure(*X*, *Y*, *Z=None*, *tau_max=0*)
Estimate dependence measure.

Calls the dependence measure function. The child classes must specify a function get_dependence_measure.

Parameters

- **Y[, Z]** (*X,*) – X,Y,Z are of the form [(var, -tau)], where var specifies the variable index and tau the time lag.
- **tau_max** (*int, optional (default: 0)*) – Maximum time lag. This may be used to make sure that estimates for different lags in X, Z, all have the same sample size.

Returns **val** – The test statistic value.

Return type float

get_shuffle_significance(*array*, *xyz*, *value*)
Returns p-value for shuffle significance test.

For residual-based test statistics only the residuals are shuffled.

Parameters

- **array** (*array-like*) – data array with X, Y, Z in rows and observations in columns
- **xyz** (*array of ints*) – XYZ identifier array of shape (dim,).
- **value** (*number*) – Value of test statistic for unshuffled estimate.

Returns **pval** – p-value

Return type float

run_test(*X*, *Y*, *Z=None*, *tau_max=0*)
Perform conditional independence test.

Calls the dependence measure and signficicance test functions. The child classes must specify a function get_dependence_measure and either or both functions get_analytic_significance and get_shuffle_significance. If recycle_residuals is True, also _get_single_residuals must be available.

Parameters

- **Y, Z** (*X,*) – X,Y,Z are of the form [(var, -tau)], where var specifies the variable index and tau the time lag.
- **tau_max** (*int, optional (default: 0)*) – Maximum time lag. This may be used to make sure that estimates for different lags in X, Z, all have the same sample size.

Returns **val, pval** – The test statistic value and the p-value. These are also made in the class as self.val and self.pval.

Return type Tuple of floats

set_dataframe(*dataframe*)
Initialize dataframe.

Parameters **dataframe** (*data object*) – This can either be the tigramite dataframe object or a pandas data frame. It must have the attributes dataframe.values yielding a numpy array of shape (observations T, variables N) and optionally a mask of the same shape.

Test statistics:

class tigramite.independence_tests.**ParCorr**(***kwargs*)
Partial correlation test.

Partial correlation is estimated as described in[1].

Parameters ****kwargs** – Arguments passed on to Parent class CondIndTest.

get_analytic_confidence(*value*, *df*, *conf_lev*)
Returns analytic confidence interval for correlation coefficient.

Based on Student's t-distribution.

Parameters

- **value** (*float*) – Test statistic value.
- **df** (*int*) – degrees of freedom of the test, given by T - dim
- **conf_lev** (*float*) – Confidence interval, eg, 0.9

Returns **(conf_lower, conf_upper)** – Upper and lower confidence bound of confidence interval.

Return type Tuple of floats

get_analytic_significance(*value*, *df*)
Returns analytic p-value from Student's t-test for the Pearson correlation coefficient.

Assumes two-sided correlation. If the degrees of freedom are less than 1, numpy.nan is returned.

Parameters

- **value** (*float*) – Test statistic value.
- **df** (*int*) – degrees of freedom of the test, given by T - dim

Returns **pval** – P-value.

Return type float or numpy.nan

get_dependence_measure(*array*, *xyz*)
Return partial correlation.

Estimated as the Pearson correlation of the residuals of a linear OLS regression.

Parameters

- **array** (*array-like*) – data array with X, Y, Z in rows and observations in columns
- **xyz** (*array of ints*) – XYZ identifier array of shape (dim,).

Returns **val** – Partial correlation coefficient.

Return type float

get_model_selection_criterion(*j*, *parents*, *tau_max=0*)
Returns Akaike's Information criterion modulo constants.

Fits a linear model of the parents to variable j and returns the score. I used to determine optimal hyperparameters in PCMCI, in particular the pc_alpha value.

Parameters

- **j** (*int*) – Index of target variable in data array.

- **parents** (*list*) – List of form [(0, -1), (3, -2), ...] containing parents.
- **tau_max** (*int, optional (default: 0)*) – Maximum time lag. This may be used to make sure that estimates for different lags in X, Z, all have the same sample size.
- **Returns** –
- **score** (*float*) – Model score.

class tigramite.independence_tests.**GPACE**(*null_dist_filename=None*, *gp_version='new'*, *gp_kernel=None*, *gp_alpha=None*, *gp_restarts=None*, *ace_version='acepack'*, ***kwargs*)

GPACE conditional independence test.

GPACE is based on a Gaussian process (GP) regression and a maximal correlation test on the residuals. GP is estimated with scikit-learn and allows to flexibly specify kernels and hyperparameters or let them be optimized automatically. The maximal correlation test is implemented with the ACE estimator either from a pure python implementation (slow) or, if rpy is available, using the R-package 'acepack'. Here the null distribution is not analytically available, but can be precomputed with the script 'generate_gpace_nulldist.py' which generates a *.npz file containing the null distribution for different sample sizes.

Notes

As described in[1], GPACE is based on a Gaussian process (GP) regression and a maximal correlation test on the residuals. To test $X \perp Y|Z$, first Z is regressed out from X and Y assuming the model

$$\begin{aligned} X &= f_X(Z) + \epsilon_X \\ Y &= f_Y(Z) + \epsilon_Y \\ \epsilon_{X,Y} &\sim \mathcal{N}(0, \sigma^2) \end{aligned}$$

using GP regression. Here σ^2 corresponds to the gp_alpha parameter. Then the residuals are transformed to uniform marginals yielding r_X, r_Y and their dependency is tested with

$$\max_{g,h} \rho\left(g(r_X), h(r_Y)\right)$$

where g, h yielding maximal correlation are obtained using the Alternating Conditional Expectation (ACE) algorithm. The null distribution of the maximal correlation can be pre-computed.

Parameters

- **null_dist_filename** (*str, otional (default: None)*) – Path to file containing null distribution. If None is passed, the default filename generated by the script "generate_gpace_nulldist.py" is used.
- **gp_version** (*{'new', 'old'}, optional (default: 'new')*) – The older GP version from scikit-learn 0.17 was used for the numerical simulations in[1]. The newer version from scikit-learn 0.19 is faster and allows more flexibility regarding kernels etc.
- **gp_kernel** (*kernel object, optional (default: None)*) – Only available for gp_version='new'. Can be any scikit-learn kernel object available in gaussian_process.kernels. The kernel specifies the covariance function of the GP. If None is passed, the kernel '1.0 * RBF(1.0)' is used as default. Note that the kernel's hyperparameters are optimized during fitting.
- **gp_alpha** (*float or array-like, optional (default: None)*) – Only available for gp_version='new'. Value added to the diagonal of the kernel matrix during

fitting. Larger values correspond to increased noise level in the observations and reduce potential numerical issues during fitting. If an array is passed, it must have the same number of entries as the data used for fitting and is used as datapoint-dependent noise level. Note that this is equivalent to adding a WhiteKernel with c=alpha. If None is passed, gp_alpha=1 is used.

- **gp_restarts** (*int, optional (default: None)*) – Only available for gp_version='new'. The number of restarts of the optimizer for finding the kernel's parameters which maximize the log- marginal likelihood. The first run of the optimizer is performed from the kernel's initial parameters, the remaining ones (if any) from thetas sampled log-uniform randomly from the space of allowed theta-values. If greater than 0, all bounds must be finite. If None is passed, n_restarts_optimizer=0 is used, implying that one run is performed.
- **ace_version** (*{'python', 'acepack'}*) – Estimator for ACE estimator of maximal correlation to use. 'python' loads the very slow pure python version available from https://pypi.python.org/pypi/ace/0.3. 'acepack' loads the much faster version from the R-package acepack. This requires the R-interface rpy2 to be installed and acepack needs to be installed in R beforehand. Note that both versions 'python' and 'acepack' may result in different results. In[1] the acepack version was used.
- ****kwargs** – Arguments passed on to parent class CondIndTest.

get_analytic_confidence(*value*, *df*, *conf_lev*)
Placeholder function, not available.

get_analytic_significance(*value*, *df*)
Returns p-value for the maximal correlation coefficient.

The null distribution is loaded and the entry for the nearest available degrees of freedom (df) is used. If it is different by more than 1% from the actual sample size, an error is raised. Then the null distribution has to be generated with the script "generate_gpace_nulldist.py". The maximal correlation coefficient is one-sided. If the degrees of freedom are less than 1, numpy.nan is returned.

Parameters

- **value** (*float*) – Test statistic value.
- **df** (*int*) – degrees of freedom of the test, given by T - dim

Returns **pval** – P-value.

Return type float or numpy.nan

get_dependence_measure(*array*, *xyz*)
Return GPACE measure.

Estimated as the maximal correlation of the residuals of a GP regression.

Parameters

- **array** (*array-like*) – data array with X, Y, Z in rows and observations in columns
- **xyz** (*array of ints*) – XYZ identifier array of shape (dim,).

Returns **val** – GPACE test statistic.

Return type float

get_model_selection_criterion(*j*, *parents*, *tau_max=0*)
Returns log marginal likelihood for GP regression.

Fits a GP model of the parents to variable j and returns the negative log marginal likelihood as a model selection score. Is used to determine optimal hyperparameters in PCMCI, in particular the pc_alpha value.

Parameters

- **j** (*int*) – Index of target variable in data array.
- **parents** (*list*) – List of form [(0, -1), (3, -2), ...] containing parents.
- **tau_max** (*int, optional (default: 0)*) – Maximum time lag. This may be used to make sure that estimates for different lags in X, Z, all have the same sample size.
- **Returns** –
- **score** (*float*) – Model score.

class tigramite.independence_tests.**CMIknn**(*knn=100*, ***kwargs*)

Conditional mutual information test based on nearest-neighbor estimator.

Conditional mutual information is the most general dependency measure coming from an information-theoretic framework. It makes no assumptions about the parametric form of the dependencies by directly estimating the underlying joint density. The test here is based on the estimator in S. Frenzel and B. Pompe, Phys. Rev. Lett. 99, 204101 (2007), combined with a shuffle test to generate the distribution under the null hypothesis of independence. The knn-estimator is suitable only for variables taking a continuous range of values. For discrete variables use the CMIsymb class.

Notes

CMI is given by

$$I(X;Y|Z) = \int p(z) \iint p(x,y|z) \log \frac{p(x,y|z)}{p(x|z)\cdot p(y|z)}\, dxdydz$$

Its knn-estimator is given by

$$\widehat{I}(X;Y|Z) = \psi(k) + \frac{1}{T}\sum_{t=1}^{T}\left[\psi(k_{Z,t}) - \psi(k_{XZ,t}) - \psi(k_{YZ,t})\right]$$

where ψ is the Digamma function. This estimator has as a parameter the number of nearest-neighbors k which determines the size of hyper-cubes around each (high-dimensional) sample point. Then $k_{Z,}, k_{XZ}, k_{YZ}$ are the numbers of neighbors in the respective subspaces.

k can be viewed as a density smoothing parameter (although it is data-adaptive unlike fixed-bandwidth estimators). For large k, the underlying dependencies are more smoothed and CMI has a larger bias, but lower variance, which is more important for significance testing. Note that the estimated CMI values can be slightly negative while CMI is a non- negative quantity.

This class requires the scipy.spatial.cKDTree package and the tigramite cython module.

Parameters

- **knn** (*int, optional (default: 100)*) – Number of nearest-neighbors which determines the size of hyper-cubes around each (high-dimensional) sample point.
- ****kwargs** – Arguments passed on to parent class CondIndTest.

get_analytic_confidence(*value*, *df*, *conf_lev*)

Placeholder function, not available.

get_analytic_significance(*value*, *df*)

Placeholder function, not available.

get_dependence_measure(*array*, *xyz*)

Returns CMI estimate as described in Frenzel and Pompe PRL (2007).

Parameters

- **array** (*array-like*) – data array with X, Y, Z in rows and observations in columns
- **xyz** (*array of ints*) – XYZ identifier array of shape (dim,).

Returns **val** – Conditional mutual information estimate.

Return type float

get_model_selection_criterion(*j*, *parents*, *tau_max=0*)
Placeholder function, not available.

class tigramite.independence_tests.**CMIsymb**(***kwargs*)
Conditional mutual information test based on discrete estimator.

Conditional mutual information is the most general dependency measure coming from an information-theoretic framework. It makes no assumptions about the parametric form of the dependencies by directly estimating the underlying joint density. The test here is based on directly estimating the joint distribution assuming symbolic input, combined with a shuffle test to generate the distribution under the null hypothesis of independence. The knn-estimator is suitable only for discrete variables. For continuous variables, either pre-process the data using the functions in data_processing or use the CMIknn class.

Notes

CMI and its estimator are given by

$$I(X;Y|Z) = \sum p(z) \sum \sum p(x,y|z) \log \frac{p(x,y|z)}{p(x|z) \cdot p(y|z)} \, dxdydz$$

Parameters ****kwargs** – Arguments passed on to parent class CondIndTest.

get_analytic_confidence(*value*, *df*, *conf_lev*)
Placeholder function, not available.

get_analytic_significance(*value*, *df*)
Placeholder function, not available.

get_dependence_measure(*array*, *xyz*)
Returns CMI estimate based on bincount histogram.

Parameters

- **array** (*array-like*) – data array with X, Y, Z in rows and observations in columns
- **xyz** (*array of ints*) – XYZ identifier array of shape (dim,).

Returns **val** – Conditional mutual information estimate.

Return type float

get_model_selection_criterion(*j*, *parents*, *tau_max=0*)
Placeholder function, not available.

CHAPTER

THREE

TIGRAMITE.DATA_PROCESSING: DATA PROCESSING FUNCTIONS

Tigramite data processing functions.

class tigramite.data_processing.**DataFrame**(*data*, *mask=None*)

Data object containing time series array and optional mask.

Alternatively, a panda dataframe can be used.

Parameters

- **data** (*array-like*) – Numpy array of shape (observations T, variables N)
- **mask** (*array-like, optional (default: None)*) – Optional mask array, must be of same shape as data

self.**data**

array-like – Numpy array of shape (observations T, variables N)

mask

array-like, optional (default: None) – Optional mask array, must be of same shape as data

tigramite.data_processing.**lowhighpass_filter**(*data*, *cutperiod*, *pass_periods='low'*)

Butterworth low- or high pass filter.

This function applies a linear filter twice, once forward and once backwards. The combined filter has linear phase.

Parameters

- **data** (*array*) – Data array of shape (time, variables).
- **cutperiod** (*int*) – Period of cutoff.
- **pass_periods** (*str, optional (default: 'low')*) – Either 'low' or 'high' to act as a low- or high-pass filter

Returns **data** – Filtered data array.

Return type array

tigramite.data_processing.**ordinal_patt_array**(*array*, *array_mask*, *dim=2*, *step=1*, *weights=False*, *verbosity=0*)

Returns symbolified array of ordinal patterns.

Each data vector (X_t, ..., X_t+(dim-1)*step) is converted to its rank vector. E.g., (0.2, -.6, 1.2) –> (1,0,2) which is then assigned to a unique integer (see Article). There are faculty(dim) possible rank vectors.

Note that the symb_array is step*(dim-1) shorter than the original array!

Reference: B. Pompe and J. Runge (2011). Momentary information transfer as a coupling measure of time series. Phys. Rev. E, 83(5), 1-12. doi:10.1103/PhysRevE.83.051122

Parameters

- **array** (*array-like*) – Data array of shape (time, variables).
- **array_mask** (*bool array*) – Data mask where True labels masked samples.
- **dim** (*int, optional (default: 2)*) – Pattern dimension
- **step** (*int, optional (default: 1)*) – Delay of pattern embedding vector.
- **weights** (*bool, optional (default: False)*) – Whether to return array of variances of embedding vectors as weights.
- **verbosity** (*int, optional (default: 0)*) – Level of verbosity.

Returns **patt, patt_mask [, patt_time]** – Tuple of converted pattern array and new length

Return type tuple of arrays

tigramite.data_processing.**quantile_bin_array**(*data*, *bins=6*)

Returns symbolified array with equal-quantile binning.

Parameters

- **data** (*array*) – Data array of shape (time, variables).
- **bins** (*int, optional (default: 6)*) – Number of bins.

Returns **symb_array** – Converted data of integer type.

Return type array

tigramite.data_processing.**smooth**(*data*, *smooth_width*, *kernel='gaussian'*, *mask=None*, *residuals=False*)

Returns either smoothed time series or its residuals.

the difference between the original and the smoothed time series (=residuals) of a kernel smoothing with gaussian (smoothing kernel width = twice the sigma!) or heaviside window, equivalent to a running mean.

Assumes data of shape (T, N) or (T,) :rtype: array :returns: smoothed/residual data

Parameters

- **data** (*array*) – Data array of shape (time, variables).
- **smooth_width** (*float*) – Window width of smoothing, 2*sigma for a gaussian.
- **kernel** (*str, optional (default: 'gaussian')*) – Smoothing kernel, 'gaussian' or 'heaviside' for a running mean.
- **mask** (*bool array, optional (default: None)*) – Data mask where True labels masked samples.
- **residuals** (*bool, optional (default: False)*) – True if residuals should be returned instead of smoothed data.

Returns **data** – Smoothed/residual data.

Return type array-like

tigramite.data_processing.**time_bin_with_mask**(*data*, *time_bin_length*, *sample_selector=None*)

Returns time binned data where only about non-masked values is averaged.

Parameters

- **data** (*array*) – Data array of shape (time, variables).
- **time_bin_length** (*int*) – Length of time bin.

- **mask** (*bool array, optional (default: None)*) – Data mask where True labels masked samples.

Returns **(bindata, T)** – Tuple of time-binned data array and new length of array.

Return type tuple of array and int

tigramite.data_processing.**var_process**(*parents_neighbors_coeffs*, *T=1000*, *use='inv_inno_cov'*, *verbosity=0*)

Returns a vector-autoregressive process with correlated innovations.

Wrapper around var_network with possibly more user-friendly input options.

Parameters

- **parents_neighbors_coeffs** (*dict*) – Dictionary of format {..., j:[(var1, lag1), (var2, lag2), ...], ...} for all variables where vars must be in [0..N-1] and lags <= 0 with number of variables N. If lag=0, a nonzero value in the covariance matrix (or its inverse) is implied. These should be the same for (i, j) and (j, i).
- **use** (*str, optional (default: 'inv_inno_cov')*) – Specifier, either 'inno_cov' or 'inv_inno_cov'.
- **T** (*int, optional (default: 1000)*) – Sample size.
- **verbosity** (*int, optional (default: 0)*) – Level of verbosity.

Returns **X** – Array of realization.

Return type array-like

tigramite.data_processing.**weighted_avg_and_std**(*values*, *axis*, *weights*)

Returns the weighted average and standard deviation.

Parameters

- **values** (*array*) – Data array of shape (time, variables).
- **axis** (*int*) – Axis to average/std about
- **weights** (*array*) – Weight array of shape (time, variables).

Returns **(average, std)** – Tuple of weighted average and standard deviation along axis.

Return type tuple of arrays

CHAPTER

FOUR

TIGRAMITE.PLOTTING: PLOTTING FUNCTIONS

Tigramite plotting package.

tigramite.plotting.**plot_graph**(*val_matrix*, *var_names=None*, *fig_ax=None*, *sig_thres=None*, *link_matrix=None*, *save_name=None*, *link_colorbar_label='MCI'*, *node_colorbar_label='auto-MCI'*, *link_width=None*, *node_pos=None*, *arrow_linewidth=30.0*, *vmin_edges=-1*, *vmax_edges=1.0*, *edge_ticks=0.4*, *cmap_edges='RdBu_r'*, *vmin_nodes=0*, *vmax_nodes=1.0*, *node_ticks=0.4*, *cmap_nodes='OrRd'*, *node_size=20*, *arrowhead_size=20*, *curved_radius=0.2*, *label_fontsize=10*, *alpha=1.0*, *node_label_size=10*, *link_label_fontsize=6*, *network_lower_bound=0.2*)

Creates a network plot.

This is still in beta. The network is defined either from True values in link_matrix, or from thresholding the val_matrix with sig_thres. Nodes denote variables, straight links contemporaneous dependencies and curved arrows lagged dependencies. The node color denotes the maximal absolute auto-dependency and the link color the value at the lag with maximal absolute cross-dependency. The link label lists the lags with significant dependency in order of absolute magnitude. The network can also be plotted over a map drawn before on the same axis. Then the node positions can be supplied in appropriate axis coordinates via node_pos.

Parameters

- **val_matrix** (*array_like*) – Matrix of shape (N, N, tau_max+1) containing test statistic values.
- **var_names** (*list, optional (default: None)*) – List of variable names. If None, range(N) is used.
- **fig_ax** (*tuple of figure and axis object, optional (default: None)*) – Figure and axes instance. If None they are created.
- **sig_thres** (*array-like, optional (default: None)*) – Matrix of significance thresholds. Must be of same shape as val_matrix. Either sig_thres or link_matrix has to be provided.
- **link_matrix** (*bool array-like, optional (default: None)*) – Matrix of significant links. Must be of same shape as val_matrix. Either sig_thres or link_matrix has to be provided.
- **save_name** (*str, optional (default: None)*) – Name of figure file to save figure. If None, figure is shown in window.
- **link_colorbar_label** (*str, optional (default: 'MCI')*) – Test statistic label.

- **node_colorbar_label** (*str, optional (default: 'auto-MCI')*) – Test statistic label for auto-dependencies.
- **link_width** (*array-like, optional (default: None)*) – Array of val_matrix.shape specifying relative link width with maximum given by arrow_linewidth. If None, all links have same width.
- **node_pos** (*dictionary, optional (default: None)*) – Dictionary of node positions in axis coordinates of form node_pos = {'x':array of shape (N,), 'y':array of shape(N)}. These coordinates could have been transformed before for basemap plots.
- **arrow_linewidth** (*float, optional (default: 30)*) – Linewidth.
- **vmin_edges** (*float, optional (default: -1)*) – Link colorbar scale lower bound.
- **vmax_edges** (*float, optional (default: 1)*) – Link colorbar scale upper bound.
- **edge_ticks** (*float, optional (default: 0.4)*) – Link tick mark interval.
- **cmap_edges** (*str, optional (default: 'RdBu_r')*) – Colormap for links.
- **vmin_nodes** (*float, optional (default: 0)*) – Node colorbar scale lower bound.
- **vmax_nodes** (*float, optional (default: 1)*) – Node colorbar scale upper bound.
- **node_ticks** (*float, optional (default: 0.4)*) – Node tick mark interval.
- **cmap_nodes** (*str, optional (default: 'OrRd')*) – Colormap for links.
- **node_size** (*int, optional (default: 20)*) – Node size.
- **arrowhead_size** (*int, optional (default: 20)*) – Size of link arrow head. Passed on to FancyArrowPatch object.
- **float, optional (default** (*curved_radius,*) – Curvature of links. Passed on to FancyArrowPatch object.
- **label_fontsize** (*int, optional (default: 10)*) – Fontsize of colorbar labels.
- **alpha** (*float, optional (default: 1.)*) – Opacity.
- **node_label_size** (*int, optional (default: 10)*) – Fontsize of node labels.
- **link_label_fontsize** (*int, optional (default: 6)*) – Fontsize of link labels.
- **network_lower_bound** (*float, optional (default: 0.2)*) – Fraction of vertical space below graph plot.

tigramite.plotting.**plot_time_series_graph**(*val_matrix*, *var_names=None*, *fig_ax=None*, *sig_thres=None*, *link_matrix=None*, *link_colorbar_label='MCI'*, *save_name=None*, *link_width=None*, *arrow_linewidth=20.0*, *vmin_edges=-1*, *vmax_edges=1.0*, *edge_ticks=0.4*, *cmap_edges='RdBu_r'*, *order=None*, *node_size=10*, *arrowhead_size=20*, *curved_radius=0.2*, *label_fontsize=10*, *alpha=1.0*, *node_label_size=10*, *label_space_left=0.1*, *label_space_top=0.0*, *network_lower_bound=0.2*)

Creates a time series graph.

This is still in beta. The time series graph's links are colored by val_matrix.

Parameters

- **val_matrix** (*array_like*) – Matrix of shape (N, N, tau_max+1) containing test statistic values.
- **var_names** (*list, optional (default: None)*) – List of variable names. If None, range(N) is used.
- **fig_ax** (*tuple of figure and axis object, optional (default: None)*) – Figure and axes instance. If None they are created.
- **sig_thres** (*array-like, optional (default: None)*) – Matrix of significance thresholds. Must be of same shape as val_matrix. Either sig_thres or link_matrix has to be provided.
- **link_matrix** (*bool array-like, optional (default: None)*) – Matrix of significant links. Must be of same shape as val_matrix. Either sig_thres or link_matrix has to be provided.
- **save_name** (*str, optional (default: None)*) – Name of figure file to save figure. If None, figure is shown in window.
- **link_colorbar_label** (*str, optional (default: 'MCI')*) – Test statistic label.
- **link_width** (*array-like, optional (default: None)*) – Array of val_matrix.shape specifying relative link width with maximum given by arrow_linewidth. If None, all links have same width.
- **order** (*list, optional (default: None)*) –
- **of variables from top to bottom.** (*order*) –
- **arrow_linewidth** (*float, optional (default: 30)*) – Linewidth.
- **vmin_edges** (*float, optional (default: -1)*) – Link colorbar scale lower bound.
- **vmax_edges** (*float, optional (default: 1)*) – Link colorbar scale upper bound.
- **edge_ticks** (*float, optional (default: 0.4)*) – Link tick mark interval.
- **cmap_edges** (*str, optional (default: 'RdBu_r')*) – Colormap for links.
- **node_size** (*int, optional (default: 20)*) – Node size.
- **arrowhead_size** (*int, optional (default: 20)*) – Size of link arrow head. Passed on to FancyArrowPatch object.

- **float, optional (default** (*curved_radius,*) – Curvature of links. Passed on to FancyArrowPatch object.
- **label_fontsize** (*int, optional (default: 10)*) – Fontsize of colorbar labels.
- **alpha** (*float, optional (default: 1.)*) – Opacity.
- **node_label_size** (*int, optional (default: 10)*) – Fontsize of node labels.
- **link_label_fontsize** (*int, optional (default: 6)*) – Fontsize of link labels.
- **label_space_left** (*float, optional (default: 0.1)*) – Fraction of horizontal figure space to allocate left of plot for labels.
- **label_space_top** (*float, optional (default: 0.)*) – Fraction of vertical figure space to allocate top of plot for labels.
- **network_lower_bound** (*float, optional (default: 0.2)*) – Fraction of vertical space below graph plot.

tigramite.plotting.**plot_timeseries**(*data*, *datatime=None*, *var_names=None*, *save_name=None*, *fig_axes=None*, *figsize=None*, *var_units=None*, *time_label='time'*, *use_mask=False*, *mask=None*, *grey_masked_samples=False*, *data_linewidth=1.0*, *skip_ticks_data_x=1*, *skip_ticks_data_y=2*, *label_fontsize=8*)

Create and save figure of stacked panels with time series.

Parameters

- **data** (*array-like*) – Data series array of shape (T, N).
- **datatime** (*array-like, optional (default: None)*) – Timelabel array. If None, range(T) is used.
- **var_names** (*list, optional (default: None)*) – List of variable names. If None, range(N) is used.
- **save_name** (*str, optional (default: None)*) – Name of figure file to save figure. If None, figure is shown in window.
- **fig_axes** (*subplots instance, optional (default: None)*) – Figure and axes instance. If None they are created as fig, axes = pyplot.subplots(N,...)
- **figsize** (*tuple of floats, optional (default: None)*) – Figure size if new figure is created. If None, default pyplot figsize is used.
- **var_units** (*list of str, optional (default: None)*) – Units of variables.
- **time_label** (*str, optional (default: '')*) – Label of time axis.
- **use_mask** (*bool, optional (default: False)*) – Whether to use masked data.
- **mask** (*array-like, optional (default: None)*) – Data mask where True labels masked samples.
- **grey_masked_samples** (*bool, optional (default: False)*) – Whether to mark masked samples by grey fills ('fill') or grey data ('data').
- **data_linewidth** (*float, optional (default: 1.)*) – Linewidth.

- **skip_ticks_data_x** (*int, optional (default: 1)*) – Skip every other tickmark.
- **skip_ticks_data_y** (*int, optional (default: 2)*) – Skip every other tickmark.
- **label_fontsize** (*int, optional (default: 10)*) – Fontsize of variable labels.

class tigramite.plotting.**setup_matrix**(*N*, *tau_max*, *var_names=None*, *figsize=None*, *minimum=-1*, *maximum=1*, *label_space_left=0.1*, *label_space_top=0.05*, *legend_width=0.15*, *legend_fontsize=10*, *x_base=1.0*, *y_base=0.4*, *plot_gridlines=False*, *lag_units=''*, *label_fontsize=10*)

Create matrix of lag function panels.

Class to setup figure object. The function add_lagfuncs(...) allows to plot the val_matrix of shape (N, N, tau_max+1). Multiple lagfunctions can be overlaid for comparison.

Parameters

- **N** (*int*) – Number of variables
- **tau_max** (*int*) – Maximum time lag.
- **var_names** (*list, optional (default: None)*) – List of variable names. If None, range(N) is used.
- **figsize** (*tuple of floats, optional (default: None)*) – Figure size if new figure is created. If None, default pyplot figsize is used.
- **minimum** (*int, optional (default: -1)*) – Lower y-axis limit.
- **maximum** (*int, optional (default: 1)*) – Upper y-axis limit.
- **label_space_left** (*float, optional (default: 0.1)*) – Fraction of horizontal figure space to allocate left of plot for labels.
- **label_space_top** (*float, optional (default: 0.05)*) – Fraction of vertical figure space to allocate top of plot for labels.
- **legend_width** (*float, optional (default: 0.15)*) – Fraction of horizontal figure space to allocate right of plot for legend.
- **x_base** (*float, optional (default: 1.)*) – x-tick intervals to show.
- **y_base** (*float, optional (default: .4)*) – y-tick intervals to show.
- **plot_gridlines** (*bool, optional (default: False)*) – Whether to show a grid.
- **lag_units** (*str, optional (default: '')*) –
- **label_fontsize** (*int, optional (default: 10)*) – Fontsize of variable labels.

add_lagfuncs(*val_matrix*, *sig_thres=None*, *conf_matrix=None*, *color='black'*, *label=None*, *two_sided_thres=True*, *marker='.'*, *markersize=5*, *alpha=1.0*)

Add lag function plot from val_matrix array.

Parameters

- **val_matrix** (*array_like*) – Matrix of shape (N, N, tau_max+1) containing test statistic values.

- **sig_thres** (*array-like, optional (default: None)*) – Matrix of significance thresholds. Must be of same shape as val_matrix.
- **conf_matrix** (*array-like, optional (default: None)*) – Matrix of shape (, N, tau_max+1, 2) containing confidence bounds.
- **color** (*str, optional (default: 'black')*) – Line color.
- **label** (*str*) – Test statistic label.
- **two_sided_thres** (*bool, optional (default: True)*) – Whether to draw sig_thres for pos. and neg. values.
- **marker** (*matplotlib marker symbol, optional (default: '.')*) – Marker.
- **markersize** (*int, optional (default: 5)*) – Marker size.
- **alpha** (*float, optional (default: 1.)*) – Opacity.

savefig(*name=None*)

Save matrix figure.

Parameters **name** (*str, optional (default: None)*) – File name. If None, figure is shown in window.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX