LaggedAnalyser Documention

`class composites.LaggedAnalyser(event, variables=None, name=None, is_categorical=None)`

Analysis of lagged composites defined with respect to a categorical event series

Arguments:

event

An xarray.DataArray with one dimension taking on categorical values, each defining a class of event (or non-event).

Optional arguments

variables, name, is_categorical

Arguments for adding variables to the LaggedAnalyser. Identical behaviour to calling add_variables directly.

`add_variable(variables, name=None, is_categorical=None, overwrite=False, join_type='outer')`

Adds an additional variable to LaggedAnalyser.variables.

Arguments

variables

An xarray.DataArray, xarray.Dataset or dictionary of xarray.DataArrays, containing data to be composited with respect to *event*. One of the coordinates of *variables* should have the same name as the coordinate of *events*. Stored internally as an xarray.Dataset. If a dictionary is passed, the DataArrays are joined according to the method 'outer'.

Optional Arguments

name

A string. If *variables* is a single xarray.DataArray then *name* will be used as the name of the array in the LaggedAnalyser.variables DataArray. Otherwise ignored.

is_categorical

An integer, if *variables* is an xarray.DataArray, or else a dictionary of integers with keys corresponding to DataArrays in the xarray.Dataset/dictionary. 0 indicates that the variable is continuous, and 1 indicates that it is categorical. Note that continuous and categorical variables are by default composited differently (see LaggedAnalyser.compute_composites). Default assumption is all DataArrays are continuous, unless a DataAarray contains an 'is_categorical' key in its DataArray.attrs, in which case this value is used.

overwrite

A boolean. If False then attempts to assign a variable who's name is already in *LaggedAnalyser.variables* will result in a ValueError

join_type

A string setting the rules for how differences in the coordinate indices of different variables are handled: "outer": use the union of object indexes "inner": use the intersection of object indexes

"left": use indexes from the pre-existing *LaggedAnalyser.variables* with each dimension

"right": use indexes from the new *variables* with each dimension

"exact": instead of aligning, raise ValueError when indexes to be aligned are not equal

"override": if indexes are of same size, rewrite indexes to be those of the pre-existing *LaggedAnalyser.variables*. Indexes for the same dimension must have the same size in all objects.

align(***kwargs*)

A convenience wrapper equivalent to calling xr.align(LaggedAnalyser.variables,LaggedAnalyser.event,**kwargs).

lagged_variables(*t*)

A convenience function that retrieves variables at lag *t* from the *LaggedAnalyser*

lag_variables(*offsets, offset_unit='days', offset_dim='time', mode='any', overwrite=False*)

Produces time lags of *LaggedAnalyser.variables* which can be used to produce lagged composites.

Arguments

offsets

An iterable of integers which represent time lags at which to lag *LaggedAnalyser.variables* in the units specified by *offset_unit*. Positive offsets denote variables *preceding* the event.

Optional arguments

offset_unit

A string, defining the units of *offsets*. Valid options are weeks, days, hours, minutes, seconds, milliseconds, and microseconds.

offset_dim

A string, defining the coordinate of *LaggedAnalyser.variables* along which offsets are to be calculated.

mode

One of 'any', 'past', or 'future'. If 'past' or 'future' is used then only positive or negative lags are valid, respectively.

overwrite

A boolean. If False, then attempts to produce a lag which already exist will raise a ValueError.

dropna(*dim, event=True, variables=False, **kwargs*)

A convenience wrapper to xarray.DataArray.dropna/xarray.Dataset.dropna

compute_composites(*dim='time', lag_vals='all', as_anomaly=False, con_func=<function mean_ds>, cat_func=<function cat_occ_ds>, inplace=True*)

Partitions *LaggedAnalyser.variables*, and time-lagged equivalents, into subsets depending on the value of *LaggedAnalyser.event*, and then computes a bulk summary metric for each.

Optional arguments

dim

A string, the coordinate along which to compute composites.

lag_vals

Either 'All', or a list of integers, denoting the time lags for which composites should be computed.

as_anomaly

A Boolean, defining whether composites should be given as absolute values or differences from the unpartitioned value.

con_func

The summary metric to use for continuous variables. Defaults to a standard mean average. If None, then continuous variables will be ignored

cat_func

The summary metric to use for categorical variables. Defaults to the occurrence probability of each categorical value. If None, then continuous variables will be ignored *inplace*

A boolean, defining whether the composite should be stored in *LaggedAnalyser.composites*

returns

An xarray.Dataset like *LaggedAnalyser.variables* but summarised according to *con_func* and *cat_func*, and with an additional coordinate *index_val*, which indexes over the values taken by *LaggedAnalyser.event*.

`aggregate_variables(dim='time', lag_vals='all', con_func=<function mean_ds>, cat_func=<function cat_occ_ds>)`

Calculates a summary metric from *LaggedAnalyser.variables* at all points where *LaggedAnalyser.event* is defined, regardless of its value.

Optional arguments

dim

A string, the name of the shared coordinate between *LaggedAnalyser.variables* and *LaggedAnalyser.event*.

lag_vals

'all' or a iterable of integers, specifying for which lag values to compute the summary metric.

con_func

The summary metric to use for continuous variables. Defaults to a standard mean average. If None, then continuous variables will be ignored

cat_func

The summary metric to use for categorical variables. Defaults to the occurrence probability of each categorical value. If None, then continuous variables will be ignored

returns

An xarray.Dataset like *LaggedAnalyser.variables* but summarised according to *con_func* and *cat_func*.

`add_derived_composite(name, func, composite_vars, as_anomaly=False)`

Applies *func* to one or multiple composites to calculate composites of derived quantities, and additionally, stores *func* to allow derived bootstrap composites to be calculated. For linear quantities, where Ex[f(x)]==f(Ex[x]), then this can minimise redundant memory use.

Arguments

name

A string, providing the name of the new variable to add.

func

A callable which must take 1 or more xarray.DataArrays as inputs

composite_vars

An iterable of strings, of the same length as the number of arguments taken by *func*. Each string must be the name of a variable in *LaggedAnalyser.variables* which will be passed into *func* in order.

Optional arguments

as_anomaly

A boolean. Whether anomaly composites or full composites should be passed in to func.

`compute_bootstraps(bootnum, dim='time', con_func=<function mean_ds>, cat_func=<function cat_occ_ds>, lag=0, synth_mode='markov', data_vars=None)`

Computes composites from synthetic event indices, which can be used to assess whether composites are insignificant.

Arguments

bootnum

An integer, the number of bootstrapped composites to compute

Optional arguments

dim

A string, the name of the shared coordinate between *LaggedAnalyser.variables* and *LaggedAnalyser.event*.

con_func

The summary metric to use for continuous variables. Defaults to a standard mean average. If None, then continuous variables will be ignored

cat_func

The summary metric to use for categorical variables. Defaults to the occurrence probability of each categorical value. If None, then continuous variables will be ignored

lag

An integer, specifying which lagged variables to use for the bootstraps. i.e. bootstraps for lag=90 will be from a completely different season than those for lag=0.

synth_mode

A string, specifying how synthetic event indices are to be computed. Valid options are: "random": categorical values are randomly chosen with the same probability of occurrence as those found in *LaggedAnalyser.event*, but with no autocorrelation. 'markov': A first order Markov chain is fitted to *LaggedAnalyser.event*, producing some autocorrelation and state dependence in the synthetic series. Generally a better approximation than "random" and so should normally be used.

data_vars

An iterable of strings, specifying for which variables bootstraps should be computed.

returns

An xarray.Dataset like *LaggedAnalyser.variables* but summarised according to *con_func* and *cat_func*, and with a new coordinate 'bootnum' of length *bootnum*.

`get_significance(bootstraps, comp, p, data_vars=None, hb_correction=False)`

Computes whether a composite is significant with respect to a given distribution of bootstrapped composites.

Arguments

bootstraps

An xarray.Dataset with a coordinate 'bootnum'

comp

An xarray Dataset of the same shape as *bootstraps* but without a 'bootnum' coordinate. Missing or additional variables are allowed, and are simply ignored.

p

A float, specifying the p-value of the 2-sided significance test (values in the range 0 to 1).

Optional arguments

data_vars

An iterable of strings, specifying for which variables significance should be computed.

hb_correction

A Boolean, specifying whether a Holm-Bonferroni correction should be applied to *p*, in order to reduce the family-wide error rate. Note that this correction is currently only applied to each variable in *comp* independently, and so will have no impact on scalar variables.

returns

An xarray.Dataset like *comp* but with boolean data, specifying whether each feature of each variable passed the significance test.

`bootstrap_significance(bootnum, p, dim='time', synth_mode='markov', reuse_lag0_boots=False, data_vars=None, hb_correction=False)`

A wrapper around *compute_bootstraps* and *test_significance*, that calculates bootstraps and applies a significance test to a number of time lagged composites simulataneously.

Arguments

bootnum

An integer, the number of bootstrapped composites to compute

p

A float, specifying the p-value of the 2-sided significance test (values in the range 0 to 1).

Optional arguments

dim

A string, the name of the shared coordinate between *LaggedAnalyser.variables* and *LaggedAnalyser.event*.

synth_mode

A string, specifying how synthetic event indices are to be computed. Valid options are: “random”: categorical values are randomly chosen with the same probability of occurrence as those found in *LaggedAnalyser.event*, but with no autocorrelation. ‘markov’: A first order Markov chain is fitted to *LaggedAnalyser.event*, producing some autocorrelation and state dependence in the synthetic series. Generally a better approximation than “random” and so should normally be used.

reuse_lag0_boots

A Boolean. If True, bootstraps are only computed for lag=0, and then used as a null distribution to assess all lagged composites. For variables which are approximately stationary across the lag timescale, then this is a good approximation and can increase performance. However if used incorrectly, it may lead to ‘significant composites’ which simply reflect the seasonal cycle. if False, separate bootstraps are computed for all time lags.

data_vars

An iterable of strings, specifying for which variables significance should be computed.

hb_correction

A Boolean, specifying whether a Holm-Bonferroni correction should be applied to *p*, in order to reduce the family-wide error rate. Note that this correction is currently only applied to each variable in *comp* independently, and so will have no impact on scalar variables.

returns

An xarray.Dataset like *LaggedAnalyser.variables* but with the *dim* dimension summarised according to *con_func* and *cat_func*, an additional *lag* coordinate, and with boolean data specifying whether each feature of each variable passed the significance test.

`compute_variable_mask(min_amp=0, min_dim_extents={}, data_vars=None, drop_allnan_composites=False, skip_sig_test=False, inplace=True, overwrite=False)`

Computes a Boolean mask to filter out features of variables with little relation to the event index. This is done through a flexible combination of statistical significance testing, applying a minimum amplitude threshold, and requiring unmasked features to span a particular spatial or temporal scale.

Optional arguments

min_amp

A float, specifying a minimum absolute amplitude below which composite anomalies are masked. *min_amp* is intepreted as a fraction of the standard deviation of each variable, calculated along the shared coordinate between *LaggedAnalyser.event* and *LaggedAnalyser.variables*, at all points where *LaggedAnalyser.event* is defined.

min_dim_extents

A dictionary in the format {str:int} where keys specify coordinate names in *LaggedAnalyser.composites*, and values specify a minimum length threshold for each coordinate. Unmasked features (as defined by the amplitude test and/or the significant test) for a variable are masked anyway if they are not part of a contiguous sequence of unmasked anomalies that meet the minimum length threshold along each dimension. E.g. *min_dim_extents* ={'lat':5,'lon':5} would mean any unmasked features in a field variable that did not span 5 grid points in both the latitudinal and longitudinal direction, would be masked.

data_vars

An iterable of strings, specifying for which variables a mask should be computed.

drop_allnan_composites

A boolean, specifying whether variables which are masked at all coordinate values should be dropped FROM WHAT THOUGH? DOUBLE CHECK.

skip_sig_test

If True, no signifance test is applied, and filtering is done solely based on amplitude and extent.

inplace

A boolean, specifying whether the output should be saved to *LaggedAnalyser.composite_masks*

overwrite

A boolean, which if *inplace*=True, specifies whether overwriting an existing *LaggedAnalyser.composite_masks* is allowed.

returns

An xarray.Dataset like *LaggedAnalyser.variables* but with the *dim* dimension summarised according to *con_func* and *cat_func*, an additional *lag* coordinate, and with boolean data specifying whether each feature of each variable is masked or not.

generate_variable_index(*data_var, mask, dim='time', sel={}, inplace=True, name=None*)

Computes standardised scalar indices from masked composites, by taking a dot product between corresponding variable and composite DataArrays.

Arguments

data_var

A string, specifying which variable should be used to compute the index.

mask

A Dataset with boolean entries. Only features of data_var where mask=True will be included in the dot product.

Optional arguments

dim

A string, specifying the name of the shared dimension between *LaggedAnalyser.variables* and LaggedAnalyser.event*

sel

A dictionary whose keys should be coordinate names, and whose elements should be valid coordinate values, or ranges of coordinate values. Specifies how data_var should be subsetted prior to dot product computation.

inplace

A boolean, specifying whether the new index should be added to *LaggedAnalyser.variables*

name

A string, specifying the name of the new index

returns

An xr.DataArray