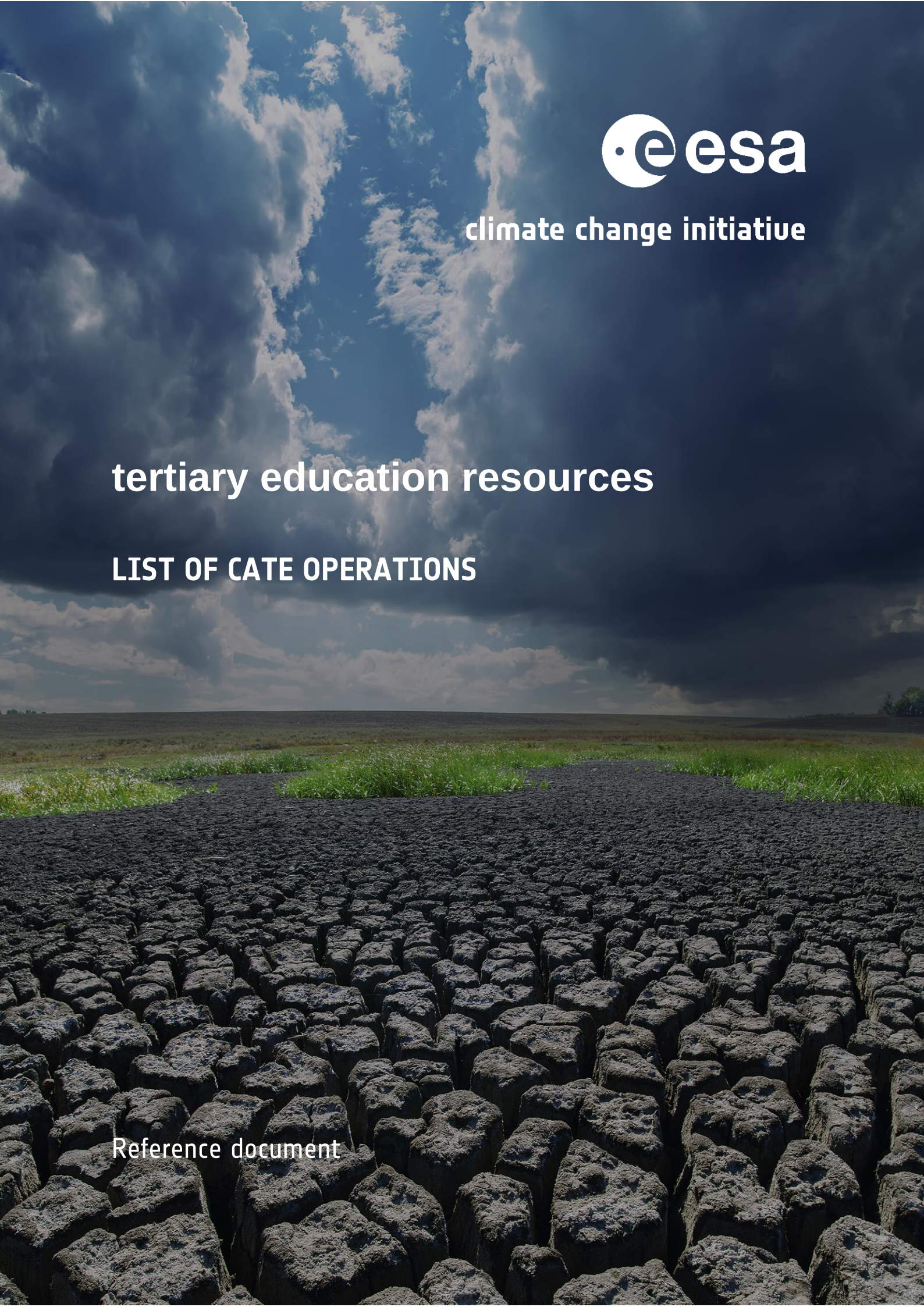

tertiary education resources

LIST OF CATE OPERATIONS

Reference document

climate change initiative– LIST OF CATE OPERATIONS
https://climate.esa.int/educate/

Developed by University of Twente (NL)

The ESA Climate Office welcomes feedback and comments
https://climate.esa.int/helpdesk/

Produced by the ESA Climate Office

LIST OF CATE OPERATIONS

Fast facts

Subjects: list of operations

Type: reference document

Complexity: medium

Lesson time required: 3 hours

Cost: none

Location: indoors

Includes the use of: internet, and the CCI Toolbox (CATE)

Keywords: CATE, climate change, essential climate variables, satellite

Brief description

This document lists all the operations for data pre-processing and analysis that are provided in CATE v. 2.15

Intended learning outcomes

After consulting this reference document, students will be able to:

find the sought operation

understand the purpose of CATE's operations, their inputs, required parameters and their outputs

Summary of activities

	Title	Description	Outcome	Requirements	Time
1	List of CATE Operations	The complete list of operations provided by CATE are listed	find the sought operation understand the purpose of CATE's operations, their inputs, required parameters and their outputs	desktop or laptop good to excellent internet connection	3 hours

Times given are for the main reading activities. They include time for exploring the operations in the CATE tool box Data Open Portal, but not experimenting with the Climate from Space application.

Health and safety

In all activities, we have assumed you will continue to follow your usual procedures relating to the use of common equipment (including electrical devices such as computers and readers), movement within the learning environment, trips and spills, first aid, and so on. Since the need for these is universal but the details of their implementation vary considerably, we have not itemised them every time. Instead, we have highlighted hazards particular to a given practical activity to inform your risk assessment.

All the activities involve the use of Climate analysis Toolbox (CATE) and online resources from the Open Data Portal. CATE connects to Open Data Portal and access huge data. Be cautious when using the option of Cashing Data Resources, this may requires huge space on your desktop. We provide sample data set to work with, which also requires considerable space on your hard drive. If you are not able – or do not wish – to use your desktop storage, then you can work with CATE Software-as-a-Service on the cloud. In this case you are reminded of your local Internet and data access safety rules.

CHAPTER ONE

Climate from Space

ESA satellites play an important role in monitoring climate change. Climate from Space (cfs.climate.esa.int) is an online resource that uses illustrated stories to summarise some of the ways in which our planet is changing and highlight the work of ESA scientists.

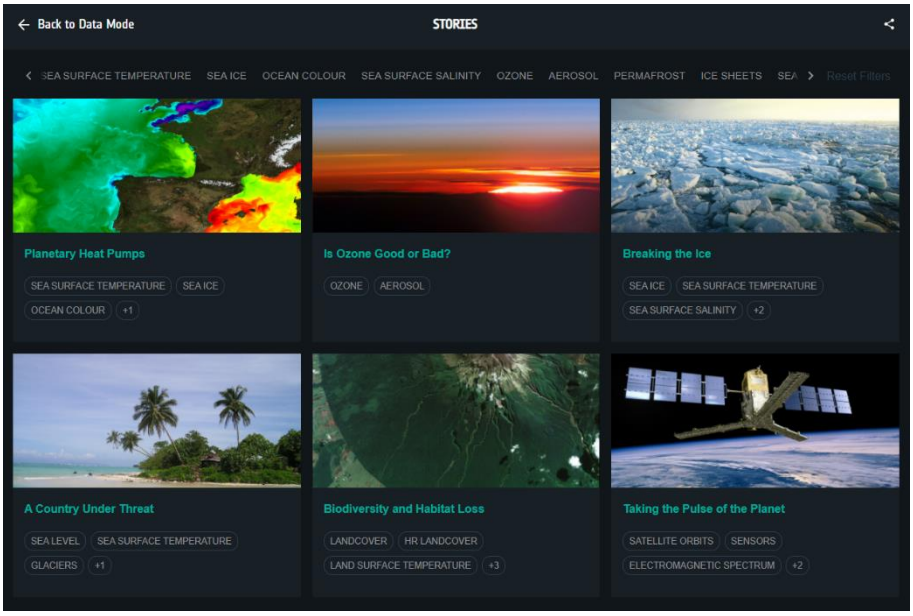

Figure 1: Stories in Climate from Space (ESA CCI)

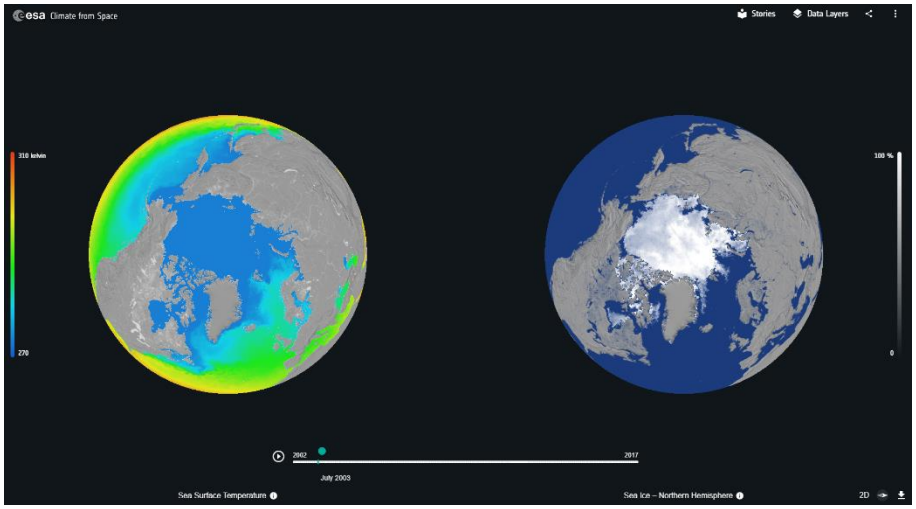

Figure 2: Exploring sea surface temperatures and sea-ice extent in the Climate from Space data viewer (ESA CCI)

ESA's Climate Change Initiative programme produces reliable global records of some key aspects of the climate known as essential climate variables (ECVs). The Climate from Space data viewer allows you to find out more about the impacts of climate change by exploring this data for yourself.

The Climate Analysis Toolbox for ESA (CATE), on the other hand, is a cloud-enabled computing environment to analyse, process and visualise climate data. With these tertiary training materials, a tutorial for CATE is provided together with a list of CATE operations.

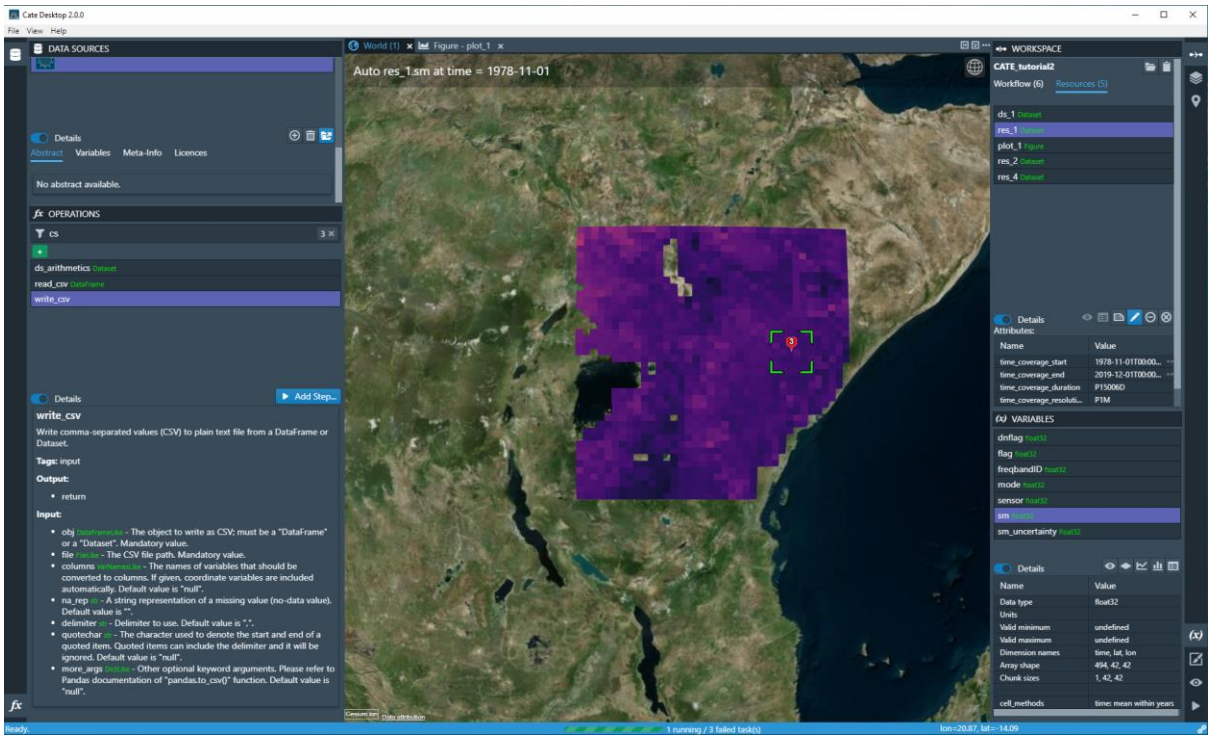

Figure 3: The CCI Toolbox (CATE).

CHAPTER TWO

Introduction to CATE

Background

The Climate Change Initiative (CCI) programme is the response from the European Space Agency (ESA) to the need for reliable and consistent climate data as formulated by the United Nations Framework Convention on Climate Change (UNFCCC). The programme consists of twenty-three parallel projects, each generating a climate data record for a well-defined essential climate variable (ECV) using the long-term global satellite data archives that ESA has established with its Member States over the past forty years. The CCI Open Data Portal and the Climate Analysis Toolbox for ESA (CATE) are the two main technical support projects within the programme. The CCI Open Data Portal (ODP) provides a single point of harmonised access to a subset of mature and validated data on Essential Climate Variables (ECVs). CATE provides tools that support visualisation, analysis and processing of ECVs and other climate data products.

Scope

CATE comprises four major software interfaces:

1. CATE Software-as-a-Service (SaaS) provides to users access to the CATE software without any installation and configuration. CATE SaaS also provides some computational resources free of charge, however service capacities might be throttled depending on the number of concurrent users logged into the system. In the future, this will be the recommended way to use CATE for most users.

2. CATE Desktop is a similar to SaaS in appearance and functionality but is a cross-platform desktop application for users who wish to use CATE predominantly with their local data sources. The application can also connect to the remote CATE services of CATE SaaS.

3. The CATE Command-Line Interface (CLI) from a local CATE installation can be used to access and process data through a command shell or console terminal.

Almost all of CATE functionality except interactive visualisation is accessible through its CLI when installed locally

4. The CATE Python API allows using CATE functions in Python programmes and may also be used to extend CATE.

The objective of this CCI Toolbox documentation is to provide to users with a full description of the **data processing operations** implemented in CATE.

CHAPTER THREE

CATE's Operations

Adjust attributes

adjust_spatial_atts

Adjust the global spatial attribute of the dataset by doing some introspection of the data set and adjusting the appropriate attributes accordingly. In case the determined attributes do not exist in the dataset, this will be added. For more information on the global attribute see Attribute Convention for Data Discovery http://wiki.esipfed.org/index.php/Attribute_Convention_for_Data_Discovery

Parameters

- **ds**– Dataset to adjust. Mandatory value.
- **allow_point**– Whether a dataset containing a single point is allowed. Default value is "false"
- **Returns** Adjusted dataset

adjust_temporal_atts

Adjust the global temporal attribute of the dataset by doing some introspection of the data set and adjusting the appropriate attributes accordingly. In case the determined attributes do not exist in the dataset, this will be added. If the attribute exist, but the dataset lacks a variable 'time', a new dimension 'time' of size on will be added and related coordinate variables 'time' and 'time_bnds' are added to the dataset. The dimension of all non-coordinate variables will be expanded by the new time dimension. For more information on the global attribute see Attribute Convention for Data Discovery http://wiki.esipfed.org/index.php/Attribute_Convention_for_Data_Discovery

Parameters

- **ds**– Dataset to adjust. Mandatory value.
- **Returns** Adjusted dataset

Anomaly calculation

anomaly_internal

Calculate anomaly using as reference data the mean of an optional region and time slice from the given dataset. If no time slice/spatial region is given, the operation will calculate anomaly using the mean of the whole dataset as the reference.

This is done for each data array in the dataset. :type monitor: Monitor :param ds: The dataset to calculate anomalies from :param time_range: Time range to use for reference data :param region: Spatial region to use for reference data :param monitor: a progress monitor. :return: The anomaly dataset

anomaly_external

Calculate anomaly with external reference data, for example, a climatology. The given reference dataset is expected to consist of 12 time slices, one for each month.

The returned dataset will contain the variable names found in both - the reference and the given dataset. Names found in the given dataset, but not in the reference, will be dropped from the resulting dataset. The calculated anomaly will be against the corresponding month of the reference data. E.g. January against January, etc.

In case spatial extents differ between the reference and the given dataset, the anomaly will be calculated on the intersection.

Parameters

- **ds**– The dataset to calculate anomalies from
- **file**– Path to reference data file
- **transform**– Apply the given transformation before calculating the anomaly. For supported operations see help on 'ds_arithmetics' operation.
- **monitor**(Monitor) – a progress monitor.
- **Returns** The anomaly dataset

Arithmetic

ds_arithmetics

Do arithmetic operations on the given dataset by providing a list of arithmetic operations and the corresponding constant. The operations will be applied to the dataset in the order in which they appear in the list. For example: 'log,+5,-2,/3,*2'

Currently supported arithmetic operations: log, log10, log2 ,log1p, exp,+,-,/,*

where: log - natural logarithm log10 - base 10 logarithm log2 - base 2 logarithm log1p - log(1+x) exp - the exponential

The operations will be applied element-wise to all arrays of the dataset.

Parameters

- **ds**– The dataset to which to apply arithmetic operations
- **op**– A comma separated list of arithmetic operations to apply
- **monitor**(Monitor) – a progress monitor.
- **Returns** The dataset with given arithmetic operations applied

Averaging

long_term_average

Create a 'mean over years' dataset by averaging the values of the given input dataset over all years. The output is a climatological dataset with the same resolution as the input dataset. E.g. a daily input dataset will create a daily climatology consisting of 365 days, a monthly input dataset will create a monthly climatology, etc.

Seasonal input datasets must have matching seasons over all years denoted by the same date each year. E.g., first date of each quarter. The output dataset will then be a seasonal climatology where each season is denoted with the same date as in the input dataset.

For further information on climatological datasets, see http://cfconventions.org/cf-conventions/v1.6.0/ cf-conventions.html#climatological-statistics

Parameters

- **ds**– A dataset to average
- **var**– If given, only these variables will be preserved in the resulting dataset
- **monitor**(Monitor) – A progress monitor
- **Returns** A climatological long term average dataset

temporal_aggregation

Perform aggregation of dataset according to the given method and output resolution. Note that the operation does not perform weighting. Depending on the combination of input and output resolutions, as well as aggregation method, the resulting dataset might yield unexpected results.

Resolution 'month' will result in a monthly dataset with each month denoted by its first date. Resolution 'season' will result in a dataset aggregated to DJF, MAM, JJA, SON seasons, each denoted by the first date of the season.

The operation also works with custom resolution strings, see: http://pandas.pydata.org/pandas-docs/ stable/timeseries.html#offset-aliases If custom_resolution is provided, it will override output_resolution.

Some examples: ‘QS-JUN’ produces an output dataset on a quarterly resolution where the year ends in 1st of June and each quarter is denoted by its first date ‘8MS’ produces an output dataset on an eight-month resolution where each period is denoted by the first date. Note that such periods will not be consistent over years. ‘8D’ produces a dataset on an eight day resolution

Parameters

- **ds**– Dataset to aggregate
- **method**– Aggregation method
- **output_resolution**– Desired temporal resolution of the output dataset
- **custom_resolution**– Custom temporal resolution, overrides output_resolution
- **Returns** Aggregated dataset

Complex operation

detect_outliers

Detect outliers in the given Dataset. When mask=True the input dataset should not contain nan values, otherwise all existing nan values will be marked as 'outliers' in the mask data array added to the output dataset.

Parameters

- **ds**– The dataset or dataframe for which to do outlier detection
- **var**– Variable or variables in the dataset to which to do outlier detection. Note that when multiple variables are selected, absolute threshold values might not make much sense. Wild cards can be used to select multiple variables matching a pattern.
- **threshold_low**– Values less or equal to this will be removed/masked
- **threshold_high**–Values greater or equal to this will be removed/masked
- **quantiles**– If True, threshold values are treated as quantiles, otherwise as absolute values.
- **mask**–If True, an ancillary variable containing flag values for outliers will be added to the dataset. Otherwise, outliers will be replaced with nan directly in the data variables.
- **monitor**– A progress monitor.
- **Returns** The dataset with outliers masked or replaced with nan

enso

Calculate the ENSO index, which is defined as a five month running mean of anomalies of monthly mean of SST data in a given region.

Parameters

- **ds**– A monthly SST dataset. Mandatory value.
- **var**– Date set variable to be used for index calculation
- **file**– path to the reference data file, e.g. climatology. A suitable reference dataset can be generated using the long_term_average operation
- **region**– Region for the index calculation, the dfault is the Nino3.4.. Defaut value is “n34”
- **custom_region**– if ‘custom’ is chosen as the ‘region’, this parameter has to be provided to set the desired region. Default value is “null”
- **Threshold** – if given, Boolean El Nino/La Nina time series will be calculated and added to the output data set, according to the given threshold. Where anomaly larger than the positive value of the threshold indicates El Nino and anomaly smaller than the negative of the given threshold indicate La Nina. Default value is Null.
- **Returns** A dataset that contains the index time series

enso_nino34

Calculate the nino34 index, which is defined as a five month running mean of anomalies of monthly mean of SST data in the Nino3.4 region [lon_min = -170, lat_min = -5; lon_max = -120, lat_max = 5]

Parameters

- **ds**– A monthly SST dataset. Mandatory value.
- **var**– Date set variable to be used for index calculation
- **file**– path to the reference data file, e.g. climatology. A suitable reference dataset can be generated using the long_term_average operation
- **Threshold** – if given, Boolean El Nino/La Nina time series will be calculated and added to the output data set, according to the given threshold. Where anomaly larger than the positive value of the threshold indicates El Nino and anomaly smaller than the negative of the given threshold indicate La Nina. Default value is Null.
- **Returns** A dataset that contains the index time series

oni

Calculate the ONI index, which is defined as a three month running mean of anomalies of monthly meanof SST data in the Nino3.4 region.

Parameters

- **ds**– A monthly SST dataset. Mandatory value.
- **var**– Date set variable to be used for index calculation
- **file**– path to the reference data file, e.g. climatology. A suitable reference dataset can be generated using the long_term_average operation
- **Threshold** – if given, Boolean El Nino/La Nina time series will be calculated and added to the output data set, according to the given threshold. Where anomaly larger than the positive value of the threshold indicates El Nino and anomaly smaller than the negative of the given threshold indicate La Nina. Default value is Null.
- **Returns** A dataset that contains the index time series

Coregistration

coregister

Perform coregistration of two datasets by resampling the replica dataset unto the grid of the master. If up-sampling has to be performed, this is achieved using interpolation, if down-sampling has to be performed, the pixels of the replica dataset are aggregated to form a coarser grid.

The returned dataset will contain the lat/lon intersection of provided master and replica datasets, resampled unto the master grid frequency.

This operation works on datasets whose spatial dimensions are defined on pixel-registered and equidistant in lat/lon coordinates grids. E.g., data points define the middle of a pixel and pixels have the same size across the dataset.

This operation will resample all variables in a dataset, as the lat/lon grid is defined per dataset. It works only if all variables in the dataset have lat and lon as dimensions.

For an overview of down-sampling/up-sampling methods used in this operation, please see https://github.com/ CAB-LAB/gridtools

Whether up-sampling or down-sampling has to be performed is determined automatically based on the relation- ship of the grids of the provided datasets.

Parameters

- **ds_master**– The dataset whose grid is used for resampling
- **ds_replica**– The dataset that will be resampled
- **method_us**– Interpolation method to use for upsampling.
- **method_ds**– Interpolation method to use for downsampling.
- **monitor**(Monitor) – a progress monitor.
- **Returns** The replica dataset resampled on the grid of the master

Correlation

pearson_correlation_scalar

Do product moment Pearson's correlation analysis.

Performs a simple correlation analysis on two data variables and returns a correlation coefficient and the corresponding p_value.

Positive correlation implies that as x grows, so does y. Negative correlation implies that as x increases, y decreases.

For more information how to interpret the results, see here, and here.

Parameters

- **ds_x**– The 'x' dataset
- **ds_y**– The 'y' dataset
- **var_x**– Dataset variable to use for correlation analysis in the 'variable' dataset
- **var_y**– Dataset variable to use for correlation analysis in the 'dependent' dataset
- **monitor**(Monitor) – a progress monitor.
- **Returns** Data frame {'corr_coef': correlation coefficient, 'p_value': probability value}

pearson_correlation

Do product moment Pearson's correlation analysis.

Perform Pearson correlation on two datasets and produce a lon/lat map of correlation coefficients and the corresponding p_values.

In case two 3D lon/lat/time datasets are provided, pixel by pixel correlation will be performed. It is also possible two pro Perform Pearson correlation analysis on two time/lat/lon datasets and produce a lat/lon map of correlation coefficients and p_values of underlying time series in the provided datasets.

The lat/lon definition of both datasets has to be the same. The length of the time dimension should be equal, but not necessarily have the same definition. E.g., it is possible to correlate different times of the same area.

There are 'x' and 'y' datasets. Positive correlations imply that as x grows, so y. Negative correlations imply that as x increases, y decreases.

For more information how to interpret the results, see here, and here.

Parameters

- **ds_x**– The 'x' dataset
- **ds_y**– The 'y' dataset
- **var_x**– Dataset variable to use for correlation analysis in the 'variable' dataset

- **var_y**– Dataset variable to use for correlation analysis in the ‘dependent’ dataset
- **monitor**(Monitor) – a progress monitor.
- **Returns** a dataset containing a map of correlation coefficients and p_values

Data Frame

data_frame_min

Select the first record of a data frame for which the given variable value is minimal.

Parameters

- **df**– The data frame or dataset.
- **var**– The variable.
- **Returns** A new, one-record data frame.

data_frame_max

Select the first record of a data frame for which the given variable value is maximal.

Parameters

- **df**– The data frame or dataset.
- **var**– The variable.
- **Returns** A new, one-record data frame.

data_frame_query

Select records from the given data frame where the given conditional query expression evaluates to “True”. If the data frame df contains a geometry column (a GeoDataFrame object), then the query expression query_expr can also contain geometric relationship tests, for example the expression "population > 100000 and @within ('-10, 34, 20, 60')"could be used on a data frame with the *population* and a *geometry* column to query for larger cities in West-Europe.

The geom argument may be a point "<lon>, <lat>"text string, a bounding box "<lon1>, <lat1>,

<lon2>, <lat2>"text, or any valid geometry WKT.

Parameters

- **df**– The data frame or dataset.
- **query_expr**– The conditional query expression.
- **Returns** A new data frame.

Input / Output

open_dataset

Open a dataset from a data source identified by ds_name.

Parameters

- **ds_name**– The name of data source. This parameter has been deprecated, please use *ds_id* instead.
- **ds_id**– The identifier for the data source.
- **time_range**– Optional time range of the requested dataset
- **region**– Optional spatial region of the requested dataset
- **var_names**– Optional names of variables of the requested dataset
- **normalise**– Whether to normalise the dataset's geo- and time-coding upon opening. See operation normalise.
- .
- **force_local**– Whether to make a local copy of remote data source if it's not present
- **local_ds_id**– Optional local identifier for newly created local copy of remote data source. Used only if force_local=True.
- **monitor**(Monitor) – A progress monitor
- **Returns** An new dataset instance.

save_dataset

Save a dataset to NetCDF file.

Parameters

- **ds**– The dataset
- **file**– File path
- **format** – NetCDF format flavour, one of 'NETCDF4', 'NETCDF4_CLASSIC', 'NETCDF3_64BIT', 'NETCDF3_CLASSIC'.
- **monitor**(Monitor) – a progress monitor.

read_object

Read a data object from a file.

Parameters

- **file**– The file path.

- **format**– Optional format name.
- **Returns** The data object.

write_object

Write a data object to a file.

Parameters

- **obj**– The object to write.
- **file**– The file path.
- **format**– Optional format name.
- **Returns** The data object.

read_text

Read a string object from a text file.

Parameters

- **file**– The text file path.
- **encoding**– Optional encoding, e.g. “utc-8”.
- **Returns** The string object.

write_text

Write an object as string to a text file.

Parameters

- **obj**– The data object.
- **file**– The text file path.
- **encoding**– Optional encoding, e.g. “utc-8”.

read_json

Read a data object from a JSON text file.

Parameters

- **file**– The JSON file path.
- **encoding**– Optional encoding, e.g. “utc-8”.
- **Returns** The data object.

write_json

Write a data object to a JSON text file. Note that the data object must be JSON-serializable.

Parameters

- **obj**– A JSON-serializable data object.
- **file**– The JSON file path.
- **encoding**– Optional encoding, e.g. “utf-8”.
- **indent**– indent used in the file, e.g. ” ” (two spaces).

read_csv

Read comma-separated values (CSV) from plain text file into a Pandas DataFrame.

Parameters

- **file**– The CSV file path.
- **delimiter**– Delimiter to use. If delimiter is None, will try to automatically determine this.
- **delim_whitespace**– Specifies whether or not whitespaces will be used as delimiter. If this option is set, nothing should be passed in for the delimiter parameter.
- **quotechar**– The character used to denote the start and end of a quoted item. Quoted items can include the delimiter and it will be ignored.
- **comment**– Indicates remainder of line should not be parsed. If found at the beginning of a line, the line will be ignored altogether. This parameter must be a single character.
- **index_col**– The name of the column that provides unique identifiers
- **more_args**– Other optional keyword arguments. Please refer to Pandas documentation of pandas.read_csv()function.
- **Returns** The DataFrame object.

read_geo_data_frame

Read a geo data frame from a file with a format such as ESRI Shapefile or GeoJSON.

Parameters

- **file**– Is either the absolute or relative path to the file to be opened.
- **more_args**– Other optional keyword arguments. Please refer to Python documentation of fiona.open()function.
- **Returns** A geopandas.GeoDataFrameobject

read_netcdf

Read a dataset from a netCDF 3/4 or HDF file.

Parameters

- **file**– The netCDF file path.

- **drop_variables**– List of variables to be dropped.
- **decode_cf**– Whether to decode CF attributes and coordinate variables.
- **normalise**– Whether to normalise the dataset's geo- and time-coding upon opening. See operation normalise.
- **decode_times** – Whether to decode time information (convert time coordinates to datetimeobjects).
- **engine**– Optional netCDF engine name.

write_netcdf3

Write a data object to a netCDF 3 file. Note that the data object must be netCDF-serializable.

Parameters

- **obj**– A netCDF-serializable data object.
- **file**– The netCDF file path.
- **engine**– Optional netCDF engine to be used

write_netcdf4

Write a data object to a netCDF 4 file. Note that the data object must be netCDF-serializable.

Parameters

- **obj**– A netCDF-serializable data object.
- **file**– The netCDF file path.
- **engine**– Optional netCDF engine to be used

Data visualisation

animate_map

Create a geographic map animation for the variable given by the data dataset "ds" and variable name 'var'. Creates an animation of the given variables from the given data set on a map with coastal lines. In case no variable name is given, the first encountered variables in the data set is animated. It is also possible to set extent of the animation. If no extent are given, a global animation is created. The following format for saving the animation are supported: html

Parameters

- **ds**– Dataset containing the variable to animate
- **var**– the variable 's name. Default value is ''null'
- **animate_dim**– Dimension to animate, if non is hether a dataset

containing a single point is allowed. Default value is “false”

- **interval**–Delay between frames in milliseconds. Defaults to 200.
- **true_range**– If True, calculates colormap and colorbar configuration parameters from the whole dataset. Can potentially take a lot of time. Defaults to False, in which case the colormap is calculated from the first frame.
- **indexers**– Optional indexers into data array of *var*. The *indexers* is a dictionary or a comma-separated string of key-value pairs that maps the variable's dimension names to constant labels. e.g. "layer=4".
- **region**– Region to animate
- **projection**– name of a global projection, see http://scitools.org.uk/cartopy/docs/v0.15/crs/projections.html
- **central_lon**– central longitude of the projection in degrees
- **title**: an optional title
- **contour_plot**– If true plot a filled contour plot of data, otherwise plots a pixelated colormesh
- **cmap_params**– optional additional colormap configuration parameters, e.g. "vmax=300, cmap='magma"'. For full reference refer to http://xarray.pydata.org/en/stable/generated/xarray.plot.contourf.html
- **plot_properties**–optional plot properties for Python matplotlib, e.g. "bins=512, range=(-1.5, 1.5)". For full reference refer to https://matplotlib.org/api/lines_api.html and https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.contourf.html
- **file**: path to a file in which to save the animation
- **monitor**– A progress monitor.
- **Return**–An animation in HTML format
- **Returns** Adjusted dataset

plot_map

Create a geographic map plot for the variable given by dataset *ds* and variable name *var*.

Plots the given variable from the given dataset on a map with coastal lines. In case no variable name is given, the first encountered variable in the dataset is plotted. In case no *time* is given, the first time slice is taken. It is also possible to set extents of the plot. If no extents are given, a global plot is created.

The plot can either be shown using pyplot functionality, or saved, if a path is given. The following file formats for saving the plot are supported: eps, jpeg, jpg, pdf, pgf, png, ps, raw, rgba, svg, svgz, tif, tiff

Parameters

- **ds**– the dataset containing the variable to plot
- **var**– the variable's name
- **indexers** – Optional indexers into data array of *var*. The *indexers* is a dictionary or a comma-separated string of key-value pairs that maps the variable's dimension names to constant labels. e.g. "layer=4".
- **region**– Region to plot
- **projection**– name of a global projection, see http://scitools.org.uk/cartopy/docs/v0.15/ crs/projections.html
- **central_lon**– central longitude of the projection in degrees
- **title**– an optional title
- **contour_plot**– If true plot a filled contour plot of data, otherwise plots a pixelated colormesh
- **properties**– optional plot properties for Python matplotlib, e.g. "bins=512, range=(- 1.5, +1.5)" For full reference refer to https://matplotlib.org/api/lines_api.html and https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.contourf.html
- **file**– path to a file in which to save the plot
- **Returns** a matplotlib figure object or None if in IPython mode

plot

Create a 1D/line or 2D/image plot of a variable given by dataset *ds* and variable name *var*.

Parameters

- **ds**– Dataset or Dataframe that contains the variable named by *var*.
- **var**– The name of the variable to plot
- **indexers** – Optional indexers into data array of *var*. The *indexers* is a dictionary or a comma-separated string of key-value pairs that maps the variable's dimension names to constant labels. e.g. "lat=12.4, time='2012-05-02'".
- **title**– an optional plot title
- **properties**– optional plot properties for Python matplotlib, e.g. "bins=512, range=(- 1.5, +1.5), label='Sea Surface Temperature'" For full reference refer to https://matplotlib.org/api/lines_api.html and https://matplotlib.org/devdocs/api/_as_gen/matplotlib.patches.Patch.html#matplotlib.patches.Patch
- **file**– path to a file in which to save the plot
- **Returns** a matplotlib figure object or None if in IPython mode

plot_data_frame

Plot a data frame. This is a wrapper of pandas.DataFrame.plot() function. For further documentation please see http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html

Resampling

resample_2d(*src*, *w*, *h*, *ds_method=54*, *us_method=11*, *fill_value=None*, *mode_rank=1*, *out=None*)

Resample a 2-D grid to a new resolution.

Parameters

- **src**– 2-D *ndarray*
- **w**– *int* New grid width
- **h**– *int* New grid height
- **ds_method**(int) – one of the *DS_* constants, optional Grid cell aggregation method for a possible downsampling
- **us_method**(int) – one of the *US_* constants, optional Grid cell interpolation method for a possible upsampling
- **fill_value**– *scalar*, optional If None, it is taken from **src** if it is a masked array, other- wise from *out* if it is a masked array, otherwise numpy's default value is used.
- **mode_rank** (int) – *scalar*, optional The rank of the frequency determined by the *ds_method* DS_MODE. One (the default) means most frequent value, zwo means second most frequent value, and so forth.
- **out**– 2-D *ndarray*, optional Alternate output array in which to place the result. The default is *None*; if provided, it must have the same shape as the expected output.
- **Returns** An resampled version of the *src* array.

downsample_2d

Down-sample a 2-D grid to a lower resolution by aggregating original grid cells.

Parameters

- **src**– 2-D *ndarray*
- **w**– *int* Grid width, which must be less than or equal to *src.shape[-1]*
- **h**– *int* Grid height, which must be less than or equal to *src.shape[-2]*
- **method**(int) – one of the *DS_* constants, optional Grid cell aggregation method
- **fill_value**– *scalar*, optional If None, it is taken from **src** if it is a

masked array, other- wise from *out* if it is a masked array, otherwise numpy's default value is used.

- **mode_rank**(int) – *scalar*, optional The rank of the frequency determined by the *method* DS_MODE. One (the default) means most frequent value, zwo means second most frequent value, and so forth.
- **out**– 2-D *ndarray*, optional Alternate output array in which to place the result. The default is *None*; if provided, it must have the same shape as the expected output.
- **Returns** A downsampled version of the *src* array.

upsample_2d

Up-sample a 2-D grid to a higher resolution by interpolating original grid cells.

Parameters

- **src**– 2-D *ndarray*
- **w**– *int* Grid width, which must be greater than or equal to *src.shape[-1]*
- **h**– *int* Grid height, which must be greater than or equal to *src.shape[-2]*
- **method**(int) – one of the *US_* constants, optional Grid cell interpolation method
- **fill_value**– *scalar*, optional If None, it is taken from **src** if it is a masked array, other- wise from *out* if it is a masked array, otherwise numpy's default value is used.
- **out**– 2-D *ndarray*, optional Alternate output array in which to place the result. The default is *None*; if provided, it must have the same shape as the expected output.
- **Returns** An upsampled version of the *src* array.

Subsetting

select_var

Filter the dataset, by leaving only the desired variables in it. The original dataset information, including original coordinates, is preserved.

Parameters

- **ds**– The dataset or dataframe from which to perform selection.
- **var**– One or more variable names to select and preserve in the dataset. All of these are valid 'var_name' 'var_name1,var_name2,var_name3' ['var_name1', 'var_name2']. One can also use wildcards when doing the selection. E.g., choosing 'var_name*' for selection will select all variables that start with 'var_name'. This can be used to select variables along with their auxiliary variables, to select all uncertainty variables, and so on.

- **Returns** A filtered dataset

subset_spatial

Do a spatial subset of the dataset

Parameters

- **ds**– Dataset to subset
- **region**– Spatial region to subset
- **mask**– Should values falling in the bounding box of the polygon but not the polygon itself be masked with NaN.
- **Returns** Subset dataset

subset_temporal

Do a temporal subset of the dataset.

Parameters

- **ds**– Dataset or dataframe to subset
- **time_range**– Time range to select
- **Returns** Subset dataset

subset_temporal_index

Do a temporal indices based subset

Parameters

- **ds**– Dataset or dataframe to subset
- **time_ind_min**– Minimum time index to select
- **time_ind_max**– Maximum time index to select
- **Returns** Subset dataset

Timeseries

tseries_point

Extract time-series from *ds* at given *lon*, *lat* position using interpolation *method* for each *var* given in a comma separated list of variables.

The operation returns a new timeseries dataset, that contains the point timeseries for all required variables with original variable meta-information preserved.

If a variable has more than three dimensions, the resulting timeseries variable will preserve all other dimensions except for lon/lat.

Parameters

- **ds**– The dataset from which to perform timeseries extraction.
- **point**– Point to extract, e.g. (lon,lat)
- **var**– Variable(s) for which to perform the timeseries selection if none is given, all variables in the dataset will be used.
- **method**– Interpolation method to use.
- **Returns** A timeseries dataset

tseries_mean

Extract spatial mean timeseries of the provided variables, return the dataset that in addition to all the informa- tion in the given dataset contains also timeseries data for the provided variables, following naming convention 'var_name1_ts_mean'

If a data variable with more dimensions than time/lat/lon is provided, the data will be reduced by taking the mean of all data values at a single time position resulting in one dimensional timeseries data variable.

Parameters

- **ds**– The dataset from which to perform timeseries extraction.
- **var**– Variables for which to perform timeseries extraction
- **calculate_std**– Whether to calculate std in addition to mean
- **std_suffix**– Std suffix to use for resulting datasets, if std is calculated.
- **monitor**(Monitor) – a progress monitor.
- **Returns** Dataset with timeseries variables

Misc

normalise

Normalise the geo- and time-coding upon opening the given dataset w.r.t. to a common (CF-compatible) con- vention used within CATE. This will maximise the compatibility of a dataset for usage with CATE's operations.

That is, * variables named "latitude" will be renamed to "lat"; * variables named "longitude" or "long" will be renamed to"lon";

Then, for equi-rectangular grids, * Remove 2D "lat" and "lon" variables; * Two new 1D coordinate variables "lat" and "lon" will be generated from original 2D forms.

Finally, it will be ensured that a "time" coordinate variable will be of type *datetime*.

Parameters

- **ds –** The dataset to normalise
- **Returns** The normalised dataset, or the original dataset, if it is already "normal".

sel

Return a new dataset with each array indexed by tick labels along the

specified dimension(s). This is a wrapper for the xarray.sel()function.

For documentation refer to xarray documentation at
http://xarray.pydata.org/en/stable/generated/xarray.Dataset.sel.html#xarray.Dataset.sel

Parameters

- **ds**– The dataset from which to select.
- **point**– Optional geographic point given by longitude and latitude
- **time**– Optional time
- **indexers** – Keyword arguments with names matching dimensions and values given by scalars, slices or arrays of tick labels. For dimensions with multi-index, the indexer may also be a dict-like object with keys matching index level names.
- **method**– Method to use for inexact matches: * None: only exact matches * pad/ ffill: propagate last valid index value forward * backfill / bfill: propagate next valid index value backward * nearest(default): use nearest valid index value
- **Returns** A new Dataset with the same contents as this dataset, except each variable and dimension is indexed by the appropriate indexers. In general, each variable's data will be a view of the variable's data in this dataset.

from_dataframe

Convert the given dataframe to an xarray dataset.

Parameters

- **df –** Dataframe to convert
- **Returns** A dataset created from the given dataframe

identity

Return the given value. This operation can be useful to create constant resources to be used as input for other operations.

Parameters

- **value –** An arbitrary (Python) value.

literal

Return the given value. This operation can be useful to create constant resources to be used as input for other operations.

Parameters

- **value –** An arbitrary (Python) literal.

pandas_fillna

Return a new dataframe with NaN values filled according to the given value or method.

Parameters

- **df**– The dataframe to fill
- **value**– Value to fill
- **method**– Method according to which to fill NaN. ffill/pad will propagate the last valid observation to the next valid observation. backfill/bfill will propagate the next valid observation back to the last valid observation.
- **limit**– Maximum number of NaN values to forward/backward fill.
- **Returns** A dataframe with nan values filled with the given value or according to the given method.

INDEX