grids

Release 0.15

May 01, 2022

Contents

Tools for extracting time series subsets from n-dimensional arrays in NetCDF, GRIB, HDF, and GeoTIFF formats. Time series can be extracted for:

1. Points - by specifying the coordinates of the point in terms of the dimensions of the array
2. Ranges or Bounding Boxes - by specifying the minimum and maximum coordinates for each dimension
3. Spatial data - if the rasters are spatial data and the appropriate dimensions are specified

CHAPTER 1

Citing Grids

If you use Grids in a project, please cite

- Our journal article at MDPI Water. doi: 10.3390/w13152066
- The source code through Zenodo. doi: 10.5281/zenodo.5225437

CHAPTER 2

Installation

```
pip install grids
```

Some of the dependencies for grids depend on system libraries and binaries which are not installed using a pip install. The easiest solution is to conda install the dependency whose system dependencies you need e.g. cfgrib or rasterio. You should not need to do this often.

```
# example conda install to get system dependencies
conda install -c conda-forge cfgrib rasterio netcdf4
pip install grids
```

CHAPTER 3

Interactive Demo

View a live demo python notebook using Google Colaboratory and GitHub Gists.

Find a copy of the notebook on GitHub Gists.

CHAPTER 4

TimeSeries Class Documentation

class grids.**TimeSeries**(*files: list*, *var: str*, *dim_order: tuple*, ***kwargs*)

Creates a time series of values from arrays contained in netCDF, grib, hdf, or geotiff formats. Values in the series are extracted by specifying coordinates of a point, range of coordinates, a spatial data file, or computing statistics for the entire array.

Parameters

- **files** (*list*) – A list (even if len==1) of either absolute file paths to netcdf, grib, hdf5, or geotiff files or urls to an OPeNDAP service (but beware the data transfer speed bottleneck)
- **var** (*str or int or list or tuple*) – The name of the variable(s) to query as they are stored in the file (e.g. often 'temp' or 'T' instead of Temperature) or the band number if you are using grib files *and* you specify the engine as pygrib. If the var is contained in a group, include the group name as a unix style path e.g. 'group_name/var'.
- **dim_order** (*tuple*) – A tuple of the names of the dimensions/coordinate variables for all of the variables listed in the "var" parameter, listed in order. If the coordinate variables are contained in a group, include the group name as a unix style path e.g. 'group_name/coord_var'.

Keyword Arguments

- **t_var** (*str*) – Name of the time dimension/coordinate variable if it is used in the data. Grids will attempt to guess if it is not "time" and you do not specify one with this argument.
- **x_var** (*str*) – Name of the x dimension/coordinate variable if it is used in the data. Grids will attempt to guess if it is not specified and not a standard name.
- **y_var** (*str*) – Name of the y dimension/coordinate variable if it is used in the data. Grids will attempt to guess if it is not specified and not a standard name.
- **engine** (*str*) – the python package used to power the file reading. Defaults to best for the type of input data. The options include 'xarray', 'opendap', 'auth-opendap', 'netcdf4', 'cfgrib', 'pygrib', 'h5py', 'rasterio'
- **xr_kwargs** (*dict*) – A dictionary of kwargs that you might need when opening complex grib files with xarray

- **user** (*str*) – a username used for authenticating remote datasets, if required by your remote data source
- **pswd** (*str*) – a password used for authenticating remote datasets, if required by your remote data source
- **session** (*requests.Session*) – a requests Session object preloaded with credentials/tokens for authentication
- **stats** (*str or tuple*) – How to reduce arrays of values to a single value in the series dataframe. Provide a list of strings (e.g. [‘mean’, ‘max’]), or a comma separated string (e.g. ‘mean,max,min’). Options include: mean, median, max, min, sum, std, or a percentile (e.g. ‘25%’). Or, provide ‘all’ which is interpreted as (mean, median, max, min, sum, std,). Or, provide ‘box’ or ‘boxplot’ which is interpreted as (max, 75%, median, mean, 25%, min,). Or, provide ‘values’ to get a flat list of all non-null values in the query so you can compute other stats.
- **fill_value** (*int*) – The value used for filling no_data/null values in the variable’s array. Default: -9999.0
- **interp_units** (*bool*) – If your data conforms to the CF NetCDF standard for time data, choose True to convert the values in the time variable to datetime strings in the pandas output. The units string for the time variable of each file is checked separately unless you specify it in the unit_str parameter.
- **origin_format** (*str*) – A datetime.strptime string for extracting the origin time from the units string.
- **unit_str** (*str*) – a CF Standard conforming string indicating how the spacing and origin of the time values. This is helpful if your files do not contain a units string. Only specify this if ALL files that you query use the same units string. Usually this looks like “step_size since YYYY-MM-DD HH:MM:SS” such as “days since 2000-01-01 00:00:00”.
- **strp_filename** (*str*) – A datetime.strptime string for extracting datetimes from patterns in file names. Only for datasets which contain 1 time step per file.

point()
Extracts a time series of values at a point for a given coordinate pair

multipoint()
Extracts a time series of values for several points given a series of coordinate values

bound()
Extracts a time series of values with a bounding box for each requested statistic

range()
Alias for TimeSeries.bound()

shape()
Extracts a time series of values on a line or within a polygon for each requested statistic

bound(*min_coords: tuple*, *max_coords: tuple*, *stats: str = None*) → pandas.core.frame.DataFrame

Parameters

- **min_coords** (*tuple*) – a tuple containing minimum coordinates of a bounding box range- coordinates given in order of the dimensions of the source arrays.
- **max_coords** (*tuple*) – a tuple containing maximum coordinates of a bounding box range- coordinates given in order of the dimensions of the source arrays.
- **stats** (*str or tuple*) – How to reduce arrays of values to a single value for the series. See class docstring.

Returns pandas.DataFrame with an index, a datetime column, and a column named for each statistic specified

multipoint(**coords*, *labels: list = None*) → pandas.core.frame.DataFrame

Extracts a time series at many points for a given series of coordinate values. Each point should have the same time coordinate and different coordinates for each other dimension.

Parameters

- **coords** (*int or float or None*) – a list of coordinate tuples or a 2D numpy array. Each coordinate pair in the list should provide a coordinate value (integer or float) for each dimension of the array, e.g. len(coordinate_pair) == len(dim_order). See TimeSeries.point for more explanation.
- **labels** (*list*) – an optional list of strings which label each of the coordinates provided. len(labels) should be equal to len(coords)

Returns pandas.DataFrame with an index, a column named datetime, and a column named values.

point(**coords*) → pandas.core.frame.DataFrame

Extracts a time series at a point for a given series of coordinate values

Parameters coords (*int or float or None*) – provide a coordinate value (integer or float) for each dimension of the array which you are creating a time series for. You need to provide exactly the same number of coordinates as there are dimensions

Returns pandas.DataFrame with an index, a column named datetime, and a column named values.

range(*min_coordinates: tuple*, *max_coordinates: tuple*, *stats: str = None*) → pandas.core.frame.DataFrame

Alias for TimeSeries.bound(). Refer to documentation for the bound method.

shape(*mask: str*, *time_range: tuple = (None, None)*, *behavior: str = 'dissolve'*, *label_attr: str = None*, *feature: str = None*, *stats: str = None*) → pandas.core.frame.DataFrame

Applicable only to source data with exactly 2 spatial dimensions, x and y, and a time dimension.

Parameters

- **mask** (*str*) – path to any spatial polygon file, e.g. shapefile or geojson, which can be read by gpd.
- **time_range** – a tuple of the min and max time range to query a time series for
- **behavior** (*str*) – determines how the vector data is used to mask the arrays. Options are: dissolve, features - dissolve: treats all features as if they were 1 feature and masks the entire set of polygons in 1 grid - features: treats each feature as a separate entity, must specify an attribute shared by each feature with unique values for each feature used to label the resulting series
- **label_attr** – The name of the attribute in the vector data features to label the several outputs
- **feature** – A value of the label_attr attribute for 1 or more features found in the provided shapefile
- **stats** (*str or tuple*) – How to reduce arrays of values to a single value for the series. See class docstring.

Returns pandas.DataFrame with an index, a datetime column, and a column named for each statistic specified

CHAPTER 5

Handling Time values

Datetime values are extracted in one of 4 ways (controlled by the `interp_units`, `units_str`, `origin_format`, and `strp_format` parameters), in this order of preference:

1. **When `interp_units` is True, interpret the time variable values as datetimes using time's units attribute.** Override the file's units attribute or provide a missing one with the `units_str` kwarg and the `origin_format` kwarg if the date doesn't use YYYY-MM-DD HH:MM:SS format.
2. When a pattern is specified with `strp_filename`, a datetime extracted from the filename is applied to all values coming from that dataset.
3. If a time variable exists, its numerical values are used without further interpretation.
4. The string file name is used if there is no time variable and no other options were provided.

CHAPTER 6

Speed Test Results

engine	point	range	shape	count	point/file	range/file	shape/file
xarray	55.589837	58.575156	60.256757	864	0.0643	0.0678	0.0697
netCDF4	33.357278	33.29196	38.594516	864	0.0386	0.0385	0.0447
h5py	10.5311	10.254075	14.19886	864	0.0122	0.0119	0.0164
cfgrib	121.737391	121.199744	121.702608	500	0.2435	0.2424	0.2434
rasterio	1.780844	1.79053	3.597246	480	0.0037	0.0037	0.0075

Python Module Index

Index