intake_xarray Documentation

Release 0.3.1

Martin Durant

Mar 30, 2020

CONTENTS:

This package enables the set of data-loading methods from Xarray to be used within the Intake data access and cataloging system.

CHAPTER

ONE

QUICKSTART

`intake-xarray` provides quick and easy access to n dimensional data suitable for reading by xarray.

1.1 Installation

To use this plugin for intake, install with the following command:

```
conda install -c conda-forge intake-xarray
```

1.2 Usage

1.2.1 Inline use

After installation, the functions `intake.open_netcdf`, `intake.open_rasterio`, `intake.open_zarr`, `intake.open_xarray_image`, and `intake.open_opendap` will become available. They can be used to open data files as xarray objects.

1.2.2 Creating Catalog Entries

Catalog entries must specify `driver: netcdf`, `driver: rasterio`, `driver: zarr`, `driver: xarray_image`, or `driver: opendap` as appropriate.

The zarr and image plugins allow access to remote data stores (s3 and gcs), settings relevant to those should be passed in using the parameter `storage_options`.

1.2.3 Choosing a Driver

While all the drivers in the `intake-xarray` plugin yield `xarray` objects, they do not all accept the same file formats.

1.3 netcdf

Supports any file format that can be passed to xarray.open_dataset. this included .nc, .grib, and unauthenticated OPeNDAP URLs

1.4 opendap

Supports OPeNDAP URLs that require authentication.

1.5 zarr

Supports .zarr directories. See https://zarr.readthedocs.io/ for more information.

1.6 rasterio

Supports any file format supported by `rasterio.open` - most commonly geotiffs.

1.7 xarray_image

Supports any file format that can be passed to `scikit-image.io.imread` which includes all the common image formats (jpg, png, tif, . . .)

CHAPTER

TWO

API REFERENCE

intake_xarray.netcdf.NetCDFSource(urlpath[, ...])	Open a xarray file.
intake_xarray.opendap.OpenDapSource(urlpath, ...)	Open a OPeNDAP source.
intake_xarray.xzarr.ZarrSource(urlpath[, ...])	Open a xarray dataset.
intake_xarray.raster.RasterIOSource(urlpath, ...)	Open a xarray dataset via RasterIO.
intake_xarray.image.ImageSource(urlpath[, ...])	Open a xarray dataset from image files.

class intake_xarray.netcdf.**NetCDFSource**(*urlpath*, *chunks=None*, *concat_dim='concat_dim'*, *xarray_kwargs=None*, *metadata=None*, *path_as_pattern=True*, ***kwargs*)

Open a xarray file.

Parameters

urlpath [str, List[str]] Path to source file. May include glob "*" characters, format pattern strings, or list. Some examples:

- `{{ CATALOG_DIR }}/data/air.nc`
- `{{ CATALOG_DIR }}/data/*.nc`
- `{{ CATALOG_DIR }}/data/air_{year}.nc`

chunks [int or dict, optional] Chunks is used to load the new dataset into dask arrays. `chunks={}` loads the dataset with dask using a single chunk for all arrays.

concat_dim [str, optional] Name of dimension along which to concatenate the files. Can be new or pre-existing. Default is 'concat_dim'.

path_as_pattern [bool or str, optional] Whether to treat the path as a pattern (ie. `data_{field}.nc`) and create new coodinates in the output corresponding to pattern fields. If str, is treated as pattern to match on. Default is True.

Attributes

cache_dirs

classname

datashape

description

has_been_persisted

hvplot Returns a hvPlot object to provide a high-level plotting API.

is_persisted

path_as_pattern

pattern

plot Returns a hvPlot object to provide a high-level plotting API.

plots List custom associated quick-plots

urlpath

Methods

close(self)	Delete open file from memory
discover(self)	Open resource and populate the source attributes.
export(self, path, **kwargs)	Save this data for sharing with other people
persist(self[, ttl])	Save data from this source to local persistent storage
read(self)	Return a version of the xarray with all the data in memory
read_chunked(self)	Return xarray object (which will have chunks)
read_partition(self, i)	Fetch one chunk of data at tuple index i
to_dask(self)	Return xarray object where variables are dask arrays
to_spark(self)	Provide an equivalent data object in Apache Spark
yaml(self[, with_plugin])	Return YAML representation of this data-source

get_persisted	
set_cache_dir	

class intake_xarray.xzarr.**ZarrSource**(*urlpath*, *storage_options=None*, *metadata=None*, ***kwargs*)

Open a xarray dataset.

Parameters

urlpath: str Path to source. This can be a local directory or a remote data service (i.e., with a protocol specifier like 's3://').

storage_options: dict Parameters passed to the backend file-system

kwargs: Further parameters are passed to xr.open_zarr

Attributes

cache_dirs

classname

datashape

description

has_been_persisted

hvplot Returns a hvPlot object to provide a high-level plotting API.

is_persisted

plot Returns a hvPlot object to provide a high-level plotting API.

plots List custom associated quick-plots

Methods

close(self)	Delete open file from memory
discover(self)	Open resource and populate the source attributes.
export(self, path, **kwargs)	Save this data for sharing with other people
persist(self[, ttl])	Save data from this source to local persistent storage
read(self)	Return a version of the xarray with all the data in memory
read_chunked(self)	Return xarray object (which will have chunks)
read_partition(self, i)	Fetch one chunk of data at tuple index i
to_dask(self)	Return xarray object where variables are dask arrays
to_spark(self)	Provide an equivalent data object in Apache Spark
yaml(self[, with_plugin])	Return YAML representation of this data-source

get_persisted	
set_cache_dir	

close(*self*)
Delete open file from memory

class intake_xarray.raster.**RasterIOSource**(*urlpath*, *chunks*, *concat_dim='concat_dim'*, *xarray_kwargs=None*, *metadata=None*, *path_as_pattern=True*, ***kwargs*)

Open a xarray dataset via RasterIO.

This creates an xarray.array, not a dataset (i.e., there is exactly one variable).

See https://rasterio.readthedocs.io/en/latest/ for the file formats supported, particularly GeoTIFF, and http://xarray.pydata.org/en/stable/generated/xarray.open_rasterio.html#xarray.open_rasterio for possible extra arguments

Parameters

urlpath: str or iterable, location of data May be a local path, or remote path if including a protocol specifier such as `'s3://'`. May include glob wildcards or format pattern strings. Must be a format supported by rasterIO (normally GeoTiff). Some examples:

- `{{ CATALOG_DIR }}data/RGB.tif`
- `s3://data/*.tif`
- `s3://data/landsat8_band{band}.tif`
- `s3://data/{location}/landsat8_band{band}.tif`
- `{{ CATALOG_DIR }}data/landsat8_{start_date:%Y%m%d}_band{band}.tif`

chunks: int or dict Chunks is used to load the new dataset into dask arrays. `chunks={}` loads the dataset with dask using a single chunk for all arrays.

path_as_pattern: bool or str, optional Whether to treat the path as a pattern (ie. `data_{field}.tif`) and create new coodinates in the output corresponding to pattern fields. If str, is treated as pattern to match on. Default is True.

Attributes

cache_dirs

classname

datashape

description

has_been_persisted

hvplot Returns a hvPlot object to provide a high-level plotting API.

is_persisted

path_as_pattern

pattern

plot Returns a hvPlot object to provide a high-level plotting API.

plots List custom associated quick-plots

urlpath

Methods

`close`(self)	Delete open file from memory
`discover`(self)	Open resource and populate the source attributes.
`export`(self, path, **kwargs)	Save this data for sharing with other people
`persist`(self[, ttl])	Save data from this source to local persistent storage
`read`(self)	Return a version of the xarray with all the data in memory
`read_chunked`(self)	Return xarray object (which will have chunks)
`read_partition`(self, i)	Fetch one chunk of data at tuple index i
`to_dask`(self)	Return xarray object where variables are dask arrays
`to_spark`(self)	Provide an equivalent data object in Apache Spark
`yaml`(self[, with_plugin])	Return YAML representation of this data-source

get_persisted	
set_cache_dir	

class intake_xarray.image.**ImageSource**(*urlpath*, *chunks=None*, *concat_dim='concat_dim'*, *metadata=None*, *path_as_pattern=True*, *storage_options=None*, ***kwargs*)

Open a xarray dataset from image files.

This creates an xarray.DataArray or an xarray.Dataset. See http://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imread for the file formats supported.

NOTE: Although `skimage.io.imread` is used by default, any reader function which accepts a file object and outputs a numpy array can be used instead.

Parameters

urlpath [str or iterable, location of data] May be a local path, or remote path if including a protocol specifier such as `'s3://'`. May include glob wildcards or format pattern strings. Must be a format supported by `skimage.io.imread` or user-supplied `imread`. Some examples:

- `{{ CATALOG_DIR }}/data/RGB.tif`
- `s3://data/*.jpeg`
- `https://example.com/image.png`
- `s3://data/Images/{{ landuse }}/{{ '%02d' % id }}.tif`

chunks [int or dict] Chunks is used to load the new dataset into dask arrays. `chunks={}` loads the dataset with dask using a single chunk for all arrays.

path_as_pattern [bool or str, optional] Whether to treat the path as a pattern (ie. `data_{field}.tif`) and create new coodinates in the output corresponding to pattern fields. If str, is treated as pattern to match on. Default is True.

concat_dim [str or iterable] Dimension over which to concatenate. If iterable, all fields must be part of the the pattern.

imread [function (optional)] Optionally provide custom imread function. Function should expect a file object and produce a numpy array. Defaults to `skimage.io.imread`.

preprocess [function (optional)] Optionally provide custom function to preprocess the image. Function should expect a numpy array for a single image and return a numpy array.

coerce_shape [iterable of len 2 (optional)] Optionally coerce the shape of the height and width of the image by setting *coerce_shape* to desired shape.

Attributes

cache_dirs

classname

datashape

description

has_been_persisted

`hvplot` Returns a hvPlot object to provide a high-level plotting API.

is_persisted

path_as_pattern

pattern

`plot` Returns a hvPlot object to provide a high-level plotting API.

`plots` List custom associated quick-plots

urlpath

Methods

close(self)	Delete open file from memory
discover(self)	Open resource and populate the source attributes.
export(self, path, **kwargs)	Save this data for sharing with other people
persist(self[, ttl])	Save data from this source to local persistent storage
read(self)	Return a version of the xarray with all the data in memory
read_chunked(self)	Return xarray object (which will have chunks)
read_partition(self, i)	Fetch one chunk of data at tuple index i
to_dask(self)	Return xarray object where variables are dask arrays
to_spark(self)	Provide an equivalent data object in Apache Spark
yaml(self[, with_plugin])	Return YAML representation of this data-source

get_persisted	
set_cache_dir	

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX