intake_astro Documentation

Release 0.1.1

Martin Durant

Sep 18, 2018

Contents:

Intake_astro allows the loading of data from local and remote FITS files into either arrays or dataframes. This can be used in conjunction with Dask for parallel and out-of-core processing of large data-sets.

CHAPTER 1

Quickstart

`intake-astro` provides quick and easy access to tabular or array data stored in the astronomical FITS binary format.

Although the plugin uses astropy under the hood, it provides extra facility for remote files and partitioned access.

1.1 Installation

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

```
conda install -c intake intake-astro
```

1.2 Usage

1.2.1 Ad-hoc

After installation, the functions `intake.open_fits_array` and `intake.open_fits_table` will become available. They can be used to load data from local or remote data

```
import intake
source = intake.open_fits_array('/data/fits/set*.fits', ext=1)
darr = source.to_dask()  # for parallel access,
arr = source.read()      # to read into memory
wcs = source.wcs         # WCS will be set from first file, if possible
```

In this case, "parallel access" will mean one partition per input file, but partitioning within files is also possible (only recommended for uncompressed input).

1.2.2 Creating Catalog Entries

To use, catalog entries must specify `driver: `` with one of the two plugins available here, ``fits_table`, `fits_array`. The data source specs will have the same parameters as the equivalent open functions. In the following example, the files might happen to be stored on amazon S3, to be accesses anonymously.

```
sources:
  some_astro_arr:
    driver: fits_array
    args:
      url: s3://mybucket/fits/*.fits
      ext: 0
      storage_options:
        anon: true
```

1.2.3 Using a Catalog

Assuming the existence of catalogs with blocks such as that above, the data-sets can be accessed with the usual intake pattern, i.e., the methods `discover()`, `read()`, etc.

As with other array-type plugins, the input to `read_partition()` for the fits_array plugin is generally a tuple of int.

CHAPTER 2

API Reference

intake_astro.FITSTableSource(url[, ...])	ext,	Read FITS tabular data into dataframes
intake_astro.FITSArraySource(url[, ...])	ext,	Read one of more local or remote FITS files using Intake

class intake_astro.**FITSTableSource**(*url*, *ext=0*, *chunksize=None*, *storage_options=None*, *metadata=None*)

Read FITS tabular data into dataframes

For one or more FITS files, which can be local or remote, with support for partitioning within files.

Parameters

url: str or list of str files to load. Can include protocol specifiers and/or glob characters

ext: str or int Extension to load. Normally 0 or 1.

chunksize: int or None For partitioning within files, use this many rows per partition. This is very inefficient for compressed files, and for remote files, will require at least touching each file to discover the number of rows, before even starting to read the data. Cannot be used with FITS tables with a "heap", i.e., containing variable- length arrays.

storage_options: dict or None Additional keyword arguments to pass to the storage back-end.

metadata: Arbitrary information to associate with this source.

After reading the schema, the source will have attributes:

''header'' - the full FITS header of one of the files as a dict,

''dtype'' - a numpy-like list of field/dtype string pairs,

''shape'' - where the number of rows will only be known if using

partitioning or for a single file input.

Attributes

cache_dirs

datashape

description

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

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

Methods

close()	Close open resources corresponding to this data source.
discover()	Open resource and populate the source attributes.
read()	Load entire dataset into a container and return it
read_chunked()	Return iterator over container fragments of data source
read_partition(i)	Return a (offset_tuple, container) corresponding to i-th partition.
to_dask()	Return a dask container for this data source
yaml([with_plugin])	Return YAML representation of this data-source

set_cache_dir	

read()
Load entire dataset into a container and return it

read_chunked()
Return iterator over container fragments of data source

read_partition(*i*)
Return a (offset_tuple, container) corresponding to i-th partition.

Offset tuple is of same length as shape.

By default, assumes i should be an integer between zero and npartitions; override for more complex indexing schemes.

to_dask()
Return a dask container for this data source

class intake_astro.**FITSArraySource**(*url*, *ext=0*, *chunks=None*, *storage_options=None*, *metadata=None*)
Read one of more local or remote FITS files using Intake

At initialisation (when something calls `._get_schema()`), the header of the first file will be read and a delayed array constructed. The properties `header, dtype, shape, wcs` will be populated from that header, and no check is made to ensure that all files are compatible.

Parameters

url: str or list of str Location of the data file(s). May include glob characters; may include protocol specifiers.

ext: int or str or tuple Extension to probe. By default, is primary extension. Can either be an integer referring to sequence number, or an extension name. If a tuple like ('SCI', 2), get the second extension named 'SCI'.

chunks: None or tuple of int size of blocks to use within each file; must specify all axes, if using. If None, each file is one partition. Do not use chunks for compressed data, and only use contiguous chunks for remote data.

storage_options: dics Parameters to pass on to storage backend

Attributes

cache_dirs

datashape

description

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

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

Methods

close()	Close open resources corresponding to this data source.
discover()	Open resource and populate the source attributes.
read()	Load entire dataset into a container and return it
read_chunked()	Return iterator over container fragments of data source
read_partition(i)	Return a (offset_tuple, container) corresponding to i-th partition.
to_dask()	Return a dask container for this data source
yaml([with_plugin])	Return YAML representation of this data-source

set_cache_dir	

read()
Load entire dataset into a container and return it

read_chunked()
Return iterator over container fragments of data source

read_partition(*i*)
Return a (offset_tuple, container) corresponding to i-th partition.

Offset tuple is of same length as shape.

By default, assumes i should be an integer between zero and npartitions; override for more complex indexing schemes.

to_dask()
Return a dask container for this data source

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index