daops Documentation

Release 0.8.1

Elle Smith

May 12, 2022

CONTENTS:

CHAPTER

ONE

QUICK GUIDE

1.1 daops - data-aware operations

The `daops` library (pronounced "day-ops") provides a python interface to a set of operations suitable for working with climate simulation outputs. It is typically used with ESGF data sets that are described in NetCDF files. `daops` is unique in that it accesses a store of *fixes* defined for datasets that are irregular when compared with others in their *population*.

When a `daops` operation, such as `subset`, is requested, the library will look up a database of known fixes before performing and calculations or transformations. The data will be loaded and *fixed* using the xarray library before the any actual operations are sent to its sister library clisops.

- Free software: BSD
- Documentation: https://daops.readthedocs.io

1.1.1 Features

The package has the following features:

- Ability to run *data-reduction* operations on large climate data sets.
- Knowledge of irregularities/anomalies in some climate data sets.
- Ability to apply *fixes* to those data sets before operating on them. This process is called *normalisation* of the data sets.

1.2 Credits

This package was created with `Cookiecutter` and the `cedadev/cookiecutter-pypackage` project template.

- Cookiecutter: https://github.com/audreyr/cookiecutter
- cookiecutter-pypackage: https://github.com/cedadev/cookiecutter-pypackage

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install daops, run this command in your terminal:

```
$ pip install daops
```

This is the preferred method to install daops, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for daops can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/roocs/daops
```

Create Conda environment named *daops*:

```
$ conda env create -f environment.yml
$ source activate daops
```

Install daops in development mode:

```
$ pip install -r requirements.txt
$ pip install -r requirements_dev.txt
$ python setup.py develop
```

Run tests:

```
$ pytest -v tests/
```

CHAPTER

THREE

USAGE

To use daops in a project:

```
import daops
```

For information on the configuration options available in daops, see: https://roocs-utils.readthedocs.io/en/latest/configuration.html#daops

CHAPTER

FOUR

API

4.1 Subset operation

daops.ops.subset.**subset**(*collection*, *time=None*, *area=None*, *level=None*, *time_components=None*, *output_dir=None*, *output_type='netcdf'*, *split_method='time:auto'*, *file_namer='standard'*, *apply_fixes=True*)

Subset input dataset according to parameters. Can be subsetted by level, area and time.

Parameters

- **collection** (*Collection of datasets to process, sequence or string of*) – comma-separated dataset identifiers.
- **time** (*Time interval (defined by start/end) or time series (a sequence of*) – datetime values) to subset over. Datetimes are typically provided as strings.
- **area** (*Area to subset over, sequence or string of comma separated lat and lon*) – bounds. Must contain 4 values.
- **level** (*Level interval (defined by start/end) or level series (a sequence of*) – values) to subset over. Levels are typically provided as integers or floats.
- **time_compoonents** (*Time components to filter on: year, month, day, hour, minute, second*)
- **output_dir** (*str or path like object describing output directory for output files.*)
- **output_type** (*{"netcdf", "nc", "zarr", "xarray"}*)
- **split_method** (*{"time:auto"}*)
- **file_namer** (*{"standard", "simple"}*)
- **apply_fixes** (*Boolean. If True fixes will be applied to datasets if needed. Default is True.*)

Returns **List of outputs in the selected type** (*a list of xarray Datasets or file paths.*)

Examples

collection: ("cmip6.ukesm1.r1.gn.tasmax.v20200101",)
time: ("1999-01-01T00:00:00", "2100-12-30T00:00:00")
area: (-5.,49.,10.,65)
level: (1000.,)
time_components: {"month": ["dec", "jan", "feb"]}
output_type: "netcdf"
output_dir: "/cache/wps/procs/req0111"

split_method: “time:auto”
file_namer: “standard”
apply_fixes: True

4.2 Average operation

daops.ops.average.**average_over_dims**(*collection*, *dims=None*, *ignore_undetected_dims=False*, *output_dir=None*, *output_type='netcdf'*, *split_method='time:auto'*, *file_namer='standard'*, *apply_fixes=True*)

Average input dataset according over indicated dimensions. Can be averaged over multiple dimensions.

Parameters

- **collection** (*Collection of datasets to process, sequence or string of comma separated dataset identifiers.*)
- **dims** (*list of dims to average over or None.*)
- **ignore_undetected_dims** (*Boolean. If False exception will be raised if requested dims do not exist in the dataset*)
- **If True missing dims will be ignored.**
- **output_dir** (*str or path like object describing output directory for output files.*)
- **output_type** (*{“netcdf”, “nc”, “zarr”, “xarray”}*)
- **split_method** (*{“time:auto”}*)
- **file_namer** (*{“standard”, “simple”}*)
- **apply_fixes** (*Boolean. If True fixes will be applied to datasets if needed. Default is True.*)

Returns **List of outputs in the selected type** (*a list of xarray Datasets or file paths.*)

Examples

collection: (“cmip6.ukesm1.r1.gn.tasmax.v20200101”,)
dims: [“time”, “lat”]
ignore_undetected_dims: (-5.,49.,10.,65)
output_type: “netcdf”
output_dir: “/cache/wps/procs/req0111”
split_method: “time:auto”
file_namer: “standard”
apply_fixes: True

4.3 Utilities

daops.utils.consolidate.**consolidate**(*collection*, ***kwargs*)

Finds the file paths relating to each input dataset. If a time range has been supplied then only the files relating to this time range are recorded.

Parameters

- **collection** – (roocs_utils.CollectionParameter) The collection of datasets to process.
- **kwargs** – Arguments of the operation taking place e.g. subset, average, or re-grid.

Returns An ordered dictionary of each dataset from the collection argument and the file paths relating to it.

daops.utils.consolidate.**get_files_matching_time_range**(*time_param*, *file_paths*)

Using the settings in *time_param*, examine each file to see if it contains years that are in the requested range.

The *time_param* can have three types:

1. type: "interval": - defined with "start_time" and "end_time"
2. type: "series": - defined with a list of "time_values"
3. type: "none": - undefined

It attempts to filter out files that do not match the selected year. For any file that we cannot do this with, the file will be read by xarray.

Parameters

- **time_param** (*TimeParameter*) – time parameter of requested date/times
- **file_paths** (*list*) – list of file paths

Returns *file_paths (list)* – filtered list of file paths

daops.utils.consolidate.**get_year**(*value*, *default*)

Gets a year from a datetime string. Defaults to the value of *default* if not defined.

daops.utils.consolidate.**get_years_from_file**(*fpath*)

Attempts to extract years from a file. First by examining the file name. If that doesn't work then it reads the file contents and looks at the time axis.

Returns a set of years.

daops.utils.consolidate.**to_year**(*time_string*)

Returns the year in a time string as an integer.

class daops.utils.core.**Characterised**(*dset*)

Bases: daops.utils.base_lookup.Lookup

Characterisation lookup class to look up whether a dataset has been characterised.

lookup_characterisation()

Attempts to find datasets in the characterisation store. Returns True if they exist in the store, returns False if not.

daops.utils.core.**is_characterised**(*collection*, *require_all=False*)

Takes in a collection (an individual data reference or a sequence of them). Returns an ordered dictionary of a collection of ids with a boolean value for each stating whether the dataset has been characterised.

If *require_all* is True: return a single Boolean value.

Parameters

- **collection** – one or more data references
- **require_all** – Boolean to require that all must be characterised

Returns Ordered Dictionary OR Boolean (if *require_all* is True)

daops.utils.core.**open_dataset**(*ds_id*, *file_paths*, *apply_fixes=True*)

Opens an xarray Dataset and applies fixes if requested. Fixes are applied to the data either before or after the dataset is opened. Whether a fix is a 'pre-processor' or 'post-processor' is defined in the fix itself.

Parameters

- **ds_id** – Dataset identifier in the form of a drs id e.g. cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.r1i1p1.latest.zostoga
- **file_paths** – (list) The file paths corresponding to the ds id.
- **apply_fixes** – Boolean. If True fixes will be applied to datasets if needed. Default is True.

Returns xarray Dataset with fixes applied to the data.

class daops.utils.base_lookup.**Lookup**(*dset*)

Bases: object

Base class used for looking up datasets in the elasticsearch indexes.

convert_to_ds_id()

Converts the input dataset to a drs id form to use with the elasticsearch index.

class daops.utils.fixer.**Fixer**(*dset*)

Bases: daops.utils.base_lookup.Lookup

Fixer class to look up fixes to apply to input dataset from the elastic search index. Gathers fixes into pre and post processors. Pre-process fixes are chained together to allow them to be executed with one call.

class daops.utils.fixer.**FuncChainer**(*funcs*)

Bases: object

Chains functions together to allow them to be executed in one call.

class daops.utils.normalise.**ResultSet**(*inputs=None*)

Bases: object

A class to hold the results from an operation e.g. subset

add(*dset*, *result*)

Adds outputs to an ordered dictionary with the ds id as the key. If the output is a file path this is also added to the file_paths variable so a list of file paths can be accessed independently.

daops.utils.normalise.**normalise**(*collection*, *apply_fixes=True*)

Takes file paths and opens and fixes the dataset they make up.

Parameters

- **collection** – Ordered dictionary of ds ids and their related file paths.
- **apply_fixes** – Boolean. If True fixes will be applied to datasets if needed. Default is True.

Returns An ordered dictionary of ds ids and their fixed xarray Dataset.

4.4 Data Utilities

daops.data_utils.attr_utils.**add_global_attrs_if_needed**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet
- **operands** – sequence of arguments

Returns Xarray Dataset

Add a global attribute if it doesn't already exist.

daops.data_utils.attr_utils.**edit_global_attrs**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet
- **operands** – sequence of arguments

Returns Xarray DataArray

Change the gloabl attributes.

daops.data_utils.attr_utils.**edit_var_attrs**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet
- **operands** – sequence of arguments

Returns Xarray Dataset

Change the attributes of a variable.

daops.data_utils.attr_utils.**remove_coord_attr**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet
- **operands** – sequence of arguments

Returns Xarray Dataset

Remove coordinate attribute that is added by xarray, for specified variables.

daops.data_utils.coord_utils.**add_coord**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet
- **operands** – sequence of arguments

Returns Xarray DataArray

Add a coordinate.

daops.data_utils.coord_utils.**add_scalar_coord**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet

- **operands** – sequence of arguments

Returns Xarray Dataset

Add a scalar coordinate.

daops.data_utils.coord_utils.**squeeze_dims**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray Dataset
- **operands** – (dict) Arguments for fix. Dims (list) to remove.

Returns Xarray Dataset

daops.data_utils.var_utils.**add_data_var**(*ds_id*, *ds*, ***operands*)

Parameters

- **ds** – Xarray DataSet
- **operands** – sequence of arguments

Returns Xarray Dataset

Add a data variable.

4.5 Processor

daops.processor.**dispatch**(*operation*, *dset*, ***kwargs*)

daops.processor.**process**(*operation*, *dset*, *mode='serial'*, ***kwargs*)

Runs the processing operation on the dataset in the correct mode (in series or parallel).

CHAPTER

FIVE

EXAMPLES

{

“cells”: [

{ “cell_type”: “code”, “execution_count”: 1, “metadata”: {}, “outputs”: [], “source”: [

“from daops.ops.subset import subsetn”, “n”, “# remove previosuly created example filen”, “import osn”, “if os.path.exists("./output_001.nc"):n”, ” os.remove("./output_001.nc")”

]

}, {

“cell_type”: “markdown”, “metadata”: {}, “source”: [

“## Subsetn”, “n”, “Daops has a subsetting operation that calls `clisops.ops.subset.subset` from the `clisops` library. n”, “n”, “Before making the call to the subset operation, `daops` will look up a database of known fixes. If there are any fixes for the requested dataset then the data will be loaded and fixed using the `xarray` library and the subsetting operation is then carried out by `clisops`.”

]

}, {

“cell_type”: “markdown”, “metadata”: {}, “source”: [

“### Results of subset and applying a fixn”, “n”, “The results of the subsetting operation in daops are returned as an ordered dictionary of the input dataset id and the output in the chosen format (xarray dataset, netcdf file paths, zarr file paths)n”, “n”, “The example below requires a fix so the elasticsearch index has been consulted.n”, “n”, “It also demostrates the results of the operation “

]

}, {

“cell_type”: “code”, “execution_count”: 2, “metadata”: {}, “outputs”: [

{ “name”: “stdout”, “output_type”: “stream”, “text”: [

“2020-12-16 12:20:42,474 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/utils/consolidate.py - INFO - Testing 1 files in time range: . . . n”, “2020-12-16 12:20:42,507 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/utils/consolidate.py - INFO - File 0: badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/zostoga_Omon_
210012.ncn”, “2020-12-16 12:20:42,766 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/utils/consolidate.py - INFO - Kept 1 filesn”, “2020-12-16 12:20:42,767 - /srv/conda/envs/notebook/lib/python3.7/site-

packages/daops/utils/normalise.py - INFO - Working on datasets: OrderedDict([('badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/*.nc', ['badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/zostoga_Omon
210012.nc'])])n", "2020-12-16 12:20:43,550 - elasticsearch - INFO - GET https://elasticsearch.ceda.ac.uk:443/roocs-fix/_doc/f34d45e4f7f5e187f64021b685adc447 [status:200 request:0.782s]n", "2020-12-16 12:20:43,566 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/utils/core.py - INFO - Running post-processing function: squeeze_dimsn", "2020-12-16 12:20:43,569 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/processor.py - INFO - Running subset [serial]: on Dataset with args: {'time': Time period to subset overn", " start time: 1955-01-01T00:00:00n", " end time: 2013-12-30T00:00:00, 'area': Area to subset over:n", " None, 'level': Level range to subset overn", " first_level: Nonen", " last_level: None, 'output_type': 'xarray', 'output_dir': None, 'split_method': 'time:auto', 'file_namer': 'standard'}n", "2020-12-16 12:20:43,597 - /srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/ops/subset.py - INFO - Processing subset for times: ('2006-01-16', '2013-12-16')n", "2020-12-16 12:20:43,599 - /srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/utils/output_utils.py - INFO - fmt_method=None, output_type=xarrayn", "2020-12-16 12:20:43,600 - /srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/utils/output_utils.py - INFO - Returning output as <class 'xarray.core.dataset.Dataset'>n"

]

}, {

"name": "stderr", "output_type": "stream", "text": [

"/srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/ops/subset.py:34: UserWarning: "start_date" not found within input date time range. Defaulting to minimum time step in xarray object.n", " result = subset_time(ds, **kwargs)n", "/srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/ops/subset.py:34: UserWarning: "end_date" has been nudged to nearest valid time step in xarray object.n", " result = subset_time(ds, **kwargs)n"

]

}, {

"data": {

"text/plain": ["OrderedDict([('badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p
" [<xarray.Dataset>n", " Dimensions: (bnds: 2, time: 96)n", " Coordinates:n", " lev float64 0.0n", " * time (time) object 2006-01-16 12:00:00 … 2013-12-16 12:00:00n", " Dimensions without coordinates: bndsn", " Data variables:n", " lev_bnds (bnds) float64 dask.array<chunksize=(2,), meta=np.ndarray>n", " time_bnds (time, bnds) object dask.array<chunksize=(96, 2), meta=np.ndarray>n", " zostoga (time) float32 dask.array<chunksize=(96,), meta=np.ndarray>n", " Attributes:n", " institution: INM (Institute for Numerical Mathematics, Moscow…n", " institute_id: INMn", " experiment_id: rcp45n", " source: inmcm4 (2009)n", " model_id: inmcm4n", " forcing: N/An", " parent_experiment_id: historicaln", " branch_time: 56940.0n", " contact:

```
                Evgeny Volodin, volodin@inm.ras.ru,INM RAS, Gubki...n", " history:
                Mon Mar 9 11:49:38 2020: ncks -d lev,,,8 -v zost...n", " comment: no
                commentsn", " references: Volodin, Diansky, Gusev 2010. Climate model
                INMCM...n", " initialization_method: 1n", " physics_version: 1n", "
                tracking_id: e16ae391-db18-4e82-b2b8-46ff24aeec77n", " product: outputn",
                " experiment: RCP4.5n", " frequency: monn", " creation_date: 2010-
                11-19T08:18:56Zn", " Conventions: CF-1.4n", " project_id: CMIP5n", "
                table_id: Table Omon (12 May 2010) f2afe576fb73a3a11aaa3cc8...n", " title:
                inmcm4 model output prepared for CMIP5 RCP4.5n", " parent_experiment:
                Historicaln", " modeling_realm: oceann", " realization: 1n", " cmor_version:
                2.0.0n", " NCO: 4.7.3])])"

            ]

        }, "execution_count": 2, "metadata": {}, "output_type": "execute_result"

    }

], "source": [

    "# An example of subsetting a dataset that requires a
    fix - the elasticsearch index is consulted.n", "n", "ds =
    "badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/*.nc"n",
    "result = subset(n", " ds,n", " time=("1955-01-01T00:00:00", "2013-12-
    30T00:00:00"),n", " output_dir=None,n", " output_type="xarray",n", " )n", "n",
    "result._results"

]

}, {

"cell_type": "markdown", "metadata": {}, "source": [

    "### File paths of outputn", "n", "If output as file paths, it is also possible to access
    just the output file paths from the results object.n", "This is demonstrated below."

]

}, {

"cell_type": "code", "execution_count": 3, "metadata": {}, "outputs": [

    { "name": "stdout", "output_type": "stream", "text": [

        "2020-12-16 12:20:43,627 - /srv/conda/envs/notebook/lib/python3.7/site-
        packages/daops/utils/consolidate.py - INFO - Test-
        ing 1 files in time range: ...n", "2020-12-16
        12:20:43,649 - /srv/conda/envs/notebook/lib/python3.7/site-
        packages/daops/utils/consolidate.py - INFO - File 0:
        badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/zostoga_Omo
        210012.ncn", "2020-12-16 12:20:43,973 -
        /srv/conda/envs/notebook/lib/python3.7/site-
        packages/daops/utils/consolidate.py - INFO - Kept 1 filesn", "2020-
        12-16 12:20:43,975 - /srv/conda/envs/notebook/lib/python3.7/site-
        packages/daops/utils/normalise.py - INFO - Working on datasets: Ordered-
        Dict([('badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/*.nc',
        ['badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/zostoga_Om
        210012.nc'])])n", "2020-12-16 12:20:44,431 - elasticsearch -
        INFO - GET https://elasticsearch.ceda.ac.uk:443/roocs-fix/_doc/
        f34d45e4f7f5e187f64021b685adc447 [status:200 request:0.454s]n",
```

```
            "2020-12-16 12:20:44,445 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/utils/core.py - INFO - Running post-processing function: squeeze_dimsn", "2020-12-16 12:20:44,447 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/processor.py - INFO - Running subset [serial]: on Dataset with args: {'time': Time period to subset overn", " start time: 1955-01-01T00:00:00n", " end time: 2013-12-30T00:00:00, 'area': Area to subset over:n", " None, 'level': Level range to subset overn", " first_level: Nonen", " last_level: None, 'output_type': 'netcdf', 'output_dir': '.', 'split_method': 'time:auto', 'file_namer': 'simple'}n", "2020-12-16 12:20:44,463 - /srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/ops/subset.py - INFO - Processing subset for times: ('2006-01-16', '2013-12-16')n", "2020-12-16 12:20:44,464 - /srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/utils/output_utils.py - INFO - fmt_method=to_netcdf, output_type=netcdfn", "2020-12-16 12:20:44,535 - /srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/utils/output_utils.py - INFO - Wrote output file: ./output_001.ncn", "ouptut file paths = ['./output_001.nc']n"
        ]
    }, {
        "name": "stderr", "output_type": "stream", "text": [
            "/srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/ops/subset.py:34: UserWarning: "start_date" not found within input date time range. Defaulting to minimum time step in xarray object.n", " result = subset_time(ds, **kwargs)n", "/srv/conda/envs/notebook/lib/python3.7/site-packages/clisops/ops/subset.py:34: UserWarning: "end_date" has been nudged to nearest valid time step in xarray object.n", " result = subset_time(ds, **kwargs)n"
        ]
    }
], "source": [
    "# An example of subsetting a dataset that requires a fix - the elasticsearch index is consulted.n", "n", "ds = "badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/*.nc"n", "result = subset(n", " ds,n", " time=("1955-01-01T00:00:00", "2013-12-30T00:00:00"),n", " output_dir=".",n", " output_type="netcdf",n", " file_namer="simple"n", " )n", "n", "print("ouptut file paths = ", result.file_uris)"
]
}, {
    "cell_type": "markdown", "metadata": {}, "source": [
        "### Checks implemented by daopsn", "n", "Daops will check that files exist in the requested time range"
    ]
}, {
    "cell_type": "code", "execution_count": 4, "metadata": {}, "outputs": [
```

{ “name”: “stdout”, “output_type”: “stream”, “text”: [

“2020-12-16 12:20:44,553 - /srv/conda/envs/notebook/lib/python3.7/site-packages/daops/utils/consolidate.py - INFO - Testing 0 files in time range: …n”, “no files to openn”

]

}

], “source”: [

“ds = "/badc/cmip5/data/cmip5/output1/INM/inmcm4/rcp45/mon/ocean/Omon/r1i1p1/latest/zostoga/*.nc"n”, “n”, “try:n”, ” result = subset(n”, ” ds,n”, ” time=("1955-01-01T00:00:00", "1990-12-30T00:00:00"),n”, ” output_dir=None,n”, ” output_type="xarray",n”, ”)n”, “n”, “except Exception as exc:n”, ” print(exc)”

]

}

], “metadata”: {

“kernelspec”: { “display_name”: “Python 3”, “language”: “python”, “name”: “python3”

}, “language_info”: {

“codemirror_mode”: { “name”: “ipython”, “version”: 3

}, “file_extension”: “.py”, “mimetype”: “text/x-python”, “name”: “python”, “nbconvert_exporter”: “python”, “pygments_lexer”: “ipython3”, “version”: “3.7.8”

}

}, “nbformat”: 4, “nbformat_minor”: 4

}

CHAPTER

SIX

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/roocs/daops/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

6.1.4 Write Documentation

daops could always use more documentation, whether as part of the official daops docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/roocs/daops/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *daops* for local development.

1. Fork the *daops* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your-name/daops.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
# For virtualenv environments:
$ mkvirtualenv daops

# For Anaconda/Miniconda environments:
$ conda create -n daops python=3.6

$ cd daops/
$ pip install -e .
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally!

5. When you're done making changes, check that you verify your changes with *black* and run the tests, including testing other Python versions with *tox*:

```
# For virtualenv environments:
$ pip install black pytest tox

# For Anaconda/Miniconda environments:
$ conda install -c conda-forge black pytest tox

$ black daops tests
$ python setup.py test
$ tox
```

6. Before committing your changes, we ask that you install *pre-commit* in your virtualenv. *Pre-commit* runs git hooks that ensure that your code resembles that of the project and catches and corrects any small errors or inconsistencies when you *git commit*:

```
# For virtualenv environments:
$ pip install pre-commit

# For Anaconda/Miniconda environments:
$ conda install -c conda-forge pre_commit

$ pre-commit install
```

7. Commit your changes and push your branch to GitHub:

```
$ git add *

$ git commit -m "Your detailed description of your changes."
# `pre-commit` will run checks at this point:
# if no errors are found, changes will be committed.
# if errors are found, modifications will be mades. Simply `git commit` again.

$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

6.3 Logging

`daops` uses the loguru library as its primary logging engine. In order to integrate this kind of logging in processes, we can use their logger:

The mechanism for enabling log reporting in scripts/notebooks using `loguru` is as follows:

For convenience, a preset logger configuration can be enabled via *daops.enable_logging()*.

6.4 Pull Request Guidelines

Before you submit a pull request, please follow these guidelines:

1. Open an *issue* on our GitHub repository with your issue that you'd like to fix or feature that you'd like to implement.
2. Perform the changes, commit and push them either to new a branch within roocs/daops or to your personal fork of daops.

> **Warning:** Try to keep your contributions within the scope of the issue that you are addressing. While it might be tempting to fix other aspects of the library as it comes up, it's better to simply to flag the problems in case others are already working on it.
>
> Consider adding a "**# TODO:**" comment if the need arises.

3. Pull requests should raise test coverage for the daops library. Code coverage is an indicator of how extensively tested the library is. If you are adding a new set of functions, they **must be tested** and **coverage percentage should not significantly decrease.**

4. If the pull request adds functionality, your functions should include docstring explanations. So long as the docstrings are syntactically correct, sphinx-autodoc will be able to automatically parse the information. Please ensure that the docstrings adhere to one of the following standards:

   - numpydoc
   - reStructuredText (ReST)

5. The pull request should work for Python 3.6, 3.7 and 3.8 as well as raise test coverage. Pull requests are also checked for documentation build status and for PEP8 compliance.

   The build statuses and build errors for pull requests can be found at: https://travis-ci.org/roocs/daops/pull_requests

Warning: PEP8 and Black is strongly enforced. Ensure that your changes pass **Flake8** and **Black** tests prior to pushing your final commits to your branch. Code formatting errors are treated as build errors and will block your pull request from being accepted.

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Ag Stephens ag.stephens@stfc.ac.uk

7.2 Co-Developers

- Eleanor Smith eleanor.smith@stfc.ac.uk
- Carsten Ehbrecht ehbrecht@dkrz.de

7.3 Contributors

- Trevor James Smith smith.trevorj@ouranos.ca

CHAPTER

EIGHT

VERSION HISTORY

8.1 v0.8.1 (2022-05-12)

8.1.1 Bug fixes

- Fix inconsistent bounds in metadata after subset operation by updating to `clisops` 0.9.1 (#94).

8.1.2 Other Changes

- Merged pre-commit autoupdate (#93).
- Updated logging using `loguru` (#92).

8.2 v0.8.0 (2022-04-13)

8.2.1 Breaking Changes

- `clisops`>=0.9.0 required.

8.2.2 New Features

- Added *AverageTime* operation (#90).
- Added support for decadal fixes (#89, #87, #84, #83).

8.3 v0.7.0 (2021-10-28)

8.3.1 Breaking Changes

- `clisops`>=0.7.0 and `roocs-utils`>=0.5.0 required.
- `fsspec` no longer constrained to `fsspec` < 0.9
- Python 3.7 required.
- `time` input for `time` in `ops.subset.subset` but now be one of [<class 'roocs_utils.parameter.param_utils.Interval'>, <class 'roocs_utils.parameter.param_utils.Series'>, <class 'NoneType'>, <class 'str'>].

- `level` input for `level` in `ops.subset.subset` but now be one of [<class 'roocs_utils.parameter.param_utils.Interval'>, <class 'roocs_utils.parameter.param_utils.Series'>, <class 'NoneType'>, <class 'str'>].

8.3.2 New Features

- `time_components` argument added to `ops.subset.subset` to allowing subsetting by time components such as year, month, day etc.

8.3.3 Other Changes

- Python 3.6 no longer tested in GitHub actions.

8.4 v0.6.0 (2021-05-19)

8.4.1 Breaking Changes

- intake, fsspec<0.9 and aiohttp are new dependencies in order to use the intake catalog search functionality.
- `clisops`>=0.6.4 and `roocs-utils`>=0.4.2 required.

8.4.2 New Features

- Intake catalog search functionality added. In use in `utils.consolidate`: if the catalog is used for the project, then consolidate will find the files within the time range specified using the intake catalog, rather than opening xarray datasets.
- `intake_catalog_url` has been added to `etc/roocs.ini`

8.5 v0.5.0 (2021-02-26)

8.5.1 New Features

- `average_over_dims` added in `daops.ops.average`.

8.5.2 Other Changes

- Refactoring of daops.ops.subset to use a base `Operation` class in `daops.ops.base`

8.6 v0.4.0 (2021-02-23)

8.6.1 Breaking Changes

- In `daops.utils.normalise` `ResultSet().file_paths` has been changed to `file_uris` to allow file paths and URLs to be collected.
- `clisops`>=0.6.1 and `roocs-utils`>=0.2.1 used.
- New dev dependency: GitPython==3.1.12
- `consolidate_dset` has been removed in `daops.utils.consolidate` as `dset_to_filepaths` from `roocs_utils.project_utils` is now used to find the file paths.

8.6.2 New Features

- `daops.utils.core.is_characterised` implemented - datasets are looked up in the character store.
- `apply_fixes` option now added to `daops.ops.subset.subset`, `daops.utils.normalise.normalise` and `daops.utils.core.open_dataset`. The default in all cases is to apply fixes (True).

8.6.3 Other Changes

- Changes to allow datasets without a time dimension to be processed.
- Swapped from travis CI to GitHub actions
- Test data no longer a submodule - the data is retrieved from GitHub when the tests are run.
- Use of `DatasetMapper` functions in `daops.consolidate` and `daops.core` to ensure all datasets are mapped to ids/file paths correctly.

8.7 v0.3.0 (2020-11-19)

Updating doc strings and documentation.

8.7.1 Breaking Changes

- `clisops`>=0.4.0 and `roocs-utils`>=0.1.4 used.
- `data_refs` parameter of `daops.ops.subset.subset` renamed to `collection`.
- `space` parameter of `daops.ops.subset.subset` renamed to `area`.
- `chunk_rules` parameter of `daops.ops.subset.subset` renamed to `split_method`.
- `filenamer` parameter of `daops.ops.subset.subset` renamed to `file_namer`.
- `output_type` parameter option added to `daops.ops.subset.subset`.
- `data_root_dir` parameter in no longer needed `daops.ops.subset.subset`.
- `data_root_dir` no longer a parameter of `daops.utils.consolidate.consolidate`.

8.7.2 New Features

- Added notebook with example usage.
- Config file now exists at `daops.etc.roocs.ini`. This can be overwritten by setting the environment variable `ROOCS_CONFIG` to the file path of a config file.
- `split_method` implemented to split output files by if they exceed the memory limit provided in `clisops.etc.roocs.ini` named `file_size_limit`. Currently only the `time:auto` exists which splits evenly on time ranges.
- `file_namer` implemented in subset operation. This has `simple` and `standard` options. `simple` numbers output files whereas `standard` names them according to the input dataset.
- Directories, file paths and dataset ids can now be used as inputs to the subset operation.
- Fixer class now looks up fixes on our elasticsearch index.

8.7.3 Other Changes

- Updated documentation.
- Functions that take the `data_refs` parameter have been changed to use `collection` parameter instead.
- Functions that take the `data_ref` parameter have been changed to use `dset` parameter instead.

8.8 v0.2.0 (2020-06-22)

- Updated to use clisops v0.2.0 (#17)
- Added xarray aggregation tests (#16)

8.9 v0.1.0 (2020-04-27)

- First release with clisops v0.1.0.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX