

---

# **roocs-utils Documentation**

***Release 0.1.4***

**Eleanor Smith**

**Oct 22, 2020**



## CONTENTS:

<b>1</b>	<b>Quick Guide</b>	<b>1</b>
1.1	roocs-utils . . . . .	1
1.2	Credits . . . . .	2
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Stable release . . . . .	3
2.2	Install from GitHub . . . . .	3
<b>3</b>	<b>Usage</b>	<b>5</b>
<b>4</b>	<b>API</b>	<b>7</b>
4.1	Parameters . . . . .	7
4.2	Xarray Utils . . . . .	9
4.3	Other utilities . . . . .	10
<b>5</b>	<b>Examples</b>	<b>11</b>
5.1	Parameters . . . . .	12
5.2	Xarray utils . . . . .	13
5.3	Other utilities . . . . .	15
<b>6</b>	<b>Contributing</b>	<b>17</b>
6.1	Types of Contributions . . . . .	17
6.2	Pull Request Guidelines . . . . .	18
6.3	Tips . . . . .	19
6.4	Deploying . . . . .	19
<b>7</b>	<b>Credits</b>	<b>21</b>
7.1	Lead . . . . .	21
7.2	Contributors . . . . .	21
<b>8</b>	<b>Version History</b>	<b>23</b>
8.1	v0.1.4 (2020-10-22) . . . . .	23
8.2	v0.1.3 (2020-10-21) . . . . .	23
8.3	v0.1.2 (2020-10-15) . . . . .	23
8.4	v0.1.1 (2020-10-12) . . . . .	24
8.5	v0.1.0 (2020-07-30) . . . . .	25
<b>9</b>	<b>Indices and tables</b>	<b>27</b>



## QUICK GUIDE

### 1.1 roocs-utils

A package containing common components for the roocs project

- Free software: BSD - see LICENSE file in top-level package directory
- Documentation: <https://roocs-utils.readthedocs.io>.

#### 1.1.1 Features

- 1. Data Inventories

#### 1. Data Inventories

The module `roocs_utils.inventory` provides tools for writing inventories of the known data holdings in a YAML format, e.g.:

```
$ python roocs_utils/inventory/inventory.py -pr c3s-cmip5
[INFO] Reading /group_workspaces/jasmin2/cp4cds1/vol1/data/c3s-cmip5/output1/MOHC/
↪HadGEM2-ES/rcp45/mon/atmos/Amon/r1i1p1/tas/v201111
↪
↪128/tas_Amon_HadGEM2-ES_rcp45_r1i1p1_212412-214911.nc
[INFO] Reading /group_workspaces/jasmin2/cp4cds1/vol1/data/c3s-cmip5/output1/MOHC/
↪HadGEM2-ES/rcp45/mon/atmos/Amon/r1i1p1/ts/v201111
↪
↪28/ts_Amon_HadGEM2-ES_rcp45_r1i1p1_209912-212411.nc
[INFO] Wrote: c3s-cmip5_MOHC_HadGEM2-ES.yml
```

One file is created for each model/institute pairing. These can be merged to one file using `roocs_utils/inventory/merge_yaml.py`

Writes:

```
- base_dir: /group_workspaces/jasmin2/cp4cds1/vol1/data/
  project: c3s-cmip5

- path: c3s-cmip5/output1/MOHC/HadGEM2-ES/rcp45/mon/atmos/Amon/r1i1p1/tas/v20111128
  dsid: c3s-cmip5.output1.MOHC.HadGEM2-ES.rcp45.mon.atmos.Amon.r1i1p1.tas.v20111128
```

(continues on next page)

(continued from previous page)

```
var_id: tas
array_dims: time lat lon
array_shape: 3529 145 192
time: 2005-12-16T00:00:00 2299-12-16T00:00:00
facets:
  activity: c3s-cmip5
  ensemble_member: r1i1p1
  experiment: rcp45
  frequency: mon
  institute: MOHC
  mip_table: Amon
  model: HadGEM2-ES
  product: output1
  realm: atmos
  variable: tas
  version: v20111128
```

## 1.2 Credits

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

- Cookiecutter: <https://github.com/audreyr/cookiecutter>
- cookiecutter-pypackage: <https://github.com/audreyr/cookiecutter-pypackage>

## INSTALLATION

### 2.1 Stable release

To install roocs-utils, run this command in your terminal:

```
$ pip install roocs-utils
```

This is the preferred method to install roocs-utils, 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 Install from GitHub

roocs-utils can be downloaded from the [Github repo](#).

```
$ git clone git://github.com/roocs/roocs-utils
$ cd roocs-utils
```

Get the submodules with ESGF test data:

```
$ git submodule update --init --recursive
```

Create Conda environment named *roocs\_utils*:

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

Install roocs-utils in development mode:

```
$ pip install -r requirements.txt
$ pip install -r requirements_dev.txt
$ pip install -e .
```

Run tests:

```
$ python -m pytest tests/
```





## USAGE

To use roocs-utils in a project:

```
import roocs_utils
```



## 4.1 Parameters

**class** roocs\_utils.parameter.area\_parameter.**AreaParameter** (*input*)

Bases: roocs\_utils.parameter.base\_parameter.\_BaseParameter

Class for area parameter used in subsetting operation.

Area can be input as:

A string of comma separated values: "0.,49.,10.,65"

A sequence of strings: ("0", "-10", "120", "40")

A sequence of numbers: [0, 49.5, 10, 65]

An area must have 4 values.

Validates the area input and parses the values into numbers.

**asdict** ()

Returns a dictionary of the area values

**parse\_method** = '\_parse\_sequence'

**property tuple**

Returns a tuple of the area values

**class** roocs\_utils.parameter.collection\_parameter.**CollectionParameter** (*input*)

Bases: roocs\_utils.parameter.base\_parameter.\_BaseParameter

Class for collection parameter used in operations.

A collection can be input as:

A string of comma separated values:

"cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.r1i1p1.latest.zostoga,cmip5.output1.MPI-M.MPI-ESM-LR.rcp45.mon.ocean.Omon.r1i1p1.latest.zostoga"

A sequence of strings: e.g. ("cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.r1i1p1.latest.zostoga", "cmip5.output1.MPI-M.MPI-ESM-LR.rcp45.mon.ocean.Omon.r1i1p1.latest.zostoga")

Validates the input and parses the ids.

**parse\_method** = '\_parse\_sequence'

**property tuple**

Returns a tuple of the collection ids

**class** roocs\_utils.parameter.level\_parameter.**LevelParameter** (*input*)

Bases: roocs\_utils.parameter.base\_parameter.\_BaseParameter

Class for level parameter used in subsetting operation.

Level can be input as:

A string of slash separated values: “1000/2000”

A sequence of strings: e.g. (“1000.50”, “2000.60”)

A sequence of numbers: e.g. (1000.50, 2000.60)

A level input must be 2 values.

If using a string input a trailing slash indicates you want to use the lowest/highest level of the dataset. e.g. “/2000” will subset from the lowest level in the dataset to 2000.

Validates the level input and parses the values into numbers.

**asdict** ()

Returns a dictionary of the level values

**parse\_method** = ‘\_parse\_range’

**property tuple**

Returns a tuple of the level values

**class** roocs\_utils.parameter.time\_parameter.**TimeParameter** (*input*)

Bases: roocs\_utils.parameter.base\_parameter.\_BaseParameter

Class for time parameter used in subsetting operation.

Time can be input as:

A string of slash separated values: “2085-01-01T12:00:00Z/2120-12-30T12:00:00Z”

A sequence of strings: e.g. (“2085-01-01T12:00:00Z”, “2120-12-30T12:00:00Z”)

A time input must be 2 values.

If using a string input a trailing slash indicates you want to use the earliest/ latest time of the dataset. e.g. “2085-01-01T12:00:00Z/” will subset from 01/01/2085 to the final time in the dataset.

Validates the times input and parses the values into isoformat.

**asdict** ()

Returns a dictionary of the time values

**parse\_method** = ‘\_parse\_range’

**property tuple**

Returns a tuple of the time values

roocs\_utils.parameter.parameterise.**parameterise** (*collection=None, area=None, level=None, time=None*)

Parameterises inputs to instances of parameter classes which allows them to be used throughout roocs. For supported formats for each input please see their individual classes.

**Parameters**

- **collection** – Collection input in any supported format.
- **area** – Area input in any supported format.
- **level** – Level input in any supported format.
- **time** – Time input in any supported format.

**Returns** Parameters as instances of their respective classes.

## 4.2 Xarray Utils

`roocs_utils.xarray_utils.xarray_utils.get_coord_by_attr(ds, attr, value)`

Returns a coordinate based on a known attribute of a coordinate.

**Parameters**

- **ds** – Xarray Dataset or DataArray
- **attr** – (str) Name of attribute to look for.
- **value** – Expected value of attribute you are looking for.

**Returns** Coordinate of xarray dataset if found.

`roocs_utils.xarray_utils.xarray_utils.get_coord_by_type(ds, coord_type, ignore_aux_coords=True)`

Returns the xarray Dataset or DataArray coordinate of the specified type.

**Parameters**

- **ds** – Xarray Dataset or DataArray
- **coord\_type** – (str) Coordinate type to find.
- **ignore\_aux\_coords** – (bool) If True then coordinates that are not dimensions are ignored. Default is True.

**Returns** Xarray Dataset coordinate (ds.coords[coord\_id])

`roocs_utils.xarray_utils.xarray_utils.get_coord_type(coord)`

Gets the coordinate type.

**Parameters** **coord** – coordinate of xarray dataset e.g. coord = ds.coords[coord\_id]

**Returns** The type of coordinate as a string. Either longitude, latitude, time, level or None

`roocs_utils.xarray_utils.xarray_utils.get_main_variable(ds, exclude_common_coords=True)`

Finds the main variable of an xarray Dataset

**Parameters**

- **ds** – xarray Dataset
- **exclude\_common\_coords** – (bool) If True then common coordinates are excluded from the search for the main variable. common coordinates are time, level, latitude, longitude and bounds. Default is True.

**Returns** (str) The main variable of the dataset e.g. 'tas'

`roocs_utils.xarray_utils.xarray_utils.is_latitude(coord)`

Determines if a coordinate is latitude.

**Parameters** **coord** – coordinate of xarray dataset e.g. `coord = ds.coords[coord_id]`

**Returns** (bool) True if the coordinate is latitude.

`roocs_utils.xarray_utils.xarray_utils.is_level(coord)`

Determines if a coordinate is level.

**Parameters** **coord** – coordinate of xarray dataset e.g. `coord = ds.coords[coord_id]`

**Returns** (bool) True if the coordinate is level.

`roocs_utils.xarray_utils.xarray_utils.is_longitude(coord)`

Determines if a coordinate is longitude.

**Parameters** **coord** – coordinate of xarray dataset e.g. `coord = ds.coords[coord_id]`

**Returns** (bool) True if the coordinate is longitude.

`roocs_utils.xarray_utils.xarray_utils.is_time(coord)`

Determines if a coordinate is time.

**Parameters** **coord** – coordinate of xarray dataset e.g. `coord = ds.coords[coord_id]`

**Returns** (bool) True if the coordinate is time.

## 4.3 Other utilities

`roocs_utils.utils.common.parse_size(size)`

Parse size string into number of bytes.

**Parameters** **size** – (str) size to parse in any unit

**Returns** (int) number of bytes

`roocs_utils.utils.time_utils.to_isoformat(tm)`

Returns an ISO 8601 string from a time object (of different types).

**Parameters** **tm** – Time object

**Returns** (str) ISO 8601 time string

## EXAMPLES

```
[27]: import roocs_utils
```

```
[28]: dir(roocs_utils)
```

```
[28]: ['AreaParameter',  
      'CONFIG',  
      'CollectionParameter',  
      'LevelParameter',  
      'TimeParameter',  
      '__author__',  
      '__builtins__',  
      '__cached__',  
      '__contact__',  
      '__copyright__',  
      '__doc__',  
      '__file__',  
      '__license__',  
      '__loader__',  
      '__name__',  
      '__package__',  
      '__path__',  
      '__spec__',  
      '__version__',  
      'area_parameter',  
      'base_parameter',  
      'collection_parameter',  
      'config',  
      'exceptions',  
      'get_config',  
      'level_parameter',  
      'parameter',  
      'parameterise',  
      'roocs_utils',  
      'time_parameter',  
      'utils',  
      'xarray_utils']
```

## 5.1 Parameters

Parameters classes are used to parse inputs of collection, area, time and level used as arguments in the subsetting operation

The area values can be input as: \* A string of comma separated values: “0.,49.,10.,65” \* A sequence of strings: (“0”, “-10”, “120”, “40”) \* A sequence of numbers: [0, 49.5, 10, 65]

```
[29]: area = roocs_utils.AreaParameter("0.,49.,10.,65")

# the lat/lon bounds can be returned in a dictionary
print(area.asdict())

# the values can be returned as a tuple
print(area.tuple)

{'lon_bnds': (0.0, 10.0), 'lat_bnds': (49.0, 65.0)}
(0.0, 49.0, 10.0, 65.0)
```

A collection can be input as \* A string of comma separated values: “cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.r1ilp1.latest.zostoga,cmip5.output1.MPI-M.MPI-ESM-LR.rcp45.mon.ocean.Omon.r1ilp1.latest.zostoga” \* A sequence of strings: e.g. (“cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.r1ilp1.latest.zostoga”, “cmip5.output1.MPI-M.MPI-ESM-LR.rcp45.mon.ocean.Omon.r1ilp1.latest.zostoga”)

```
[30]: collection = roocs_utils.CollectionParameter("cmip5.output1.INM.inmcm4.rcp45.mon.
↪ocean.Omon.r1ilp1.latest.zostoga,cmip5.output1.MPI-M.MPI-ESM-LR.rcp45.mon.ocean.
↪Omon.r1ilp1.latest.zostoga")

# the collection ids can be returned as a tuple
print(collection.tuple)

('cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.r1ilp1.latest.zostoga', 'cmip5.
↪output1.MPI-M.MPI-ESM-LR.rcp45.mon.ocean.Omon.r1ilp1.latest.zostoga')
```

Level can be input as: \* A string of slash separated values: “1000/2000” \* A sequence of strings: e.g. (“1000.50”, “2000.60”) A sequence of numbers: e.g. (1000.50, 2000.60)

Level inputs should be a range of the levels you want to subset over

```
[31]: level = roocs_utils.LevelParameter((1000.50, 2000.60))

# the first and last level in the range provided can be returned in a dictionary
print(level.asdict())

# the values can be returned as a tuple
print(level.tuple)

{'first_level': 1000.5, 'last_level': 2000.6}
(1000.5, 2000.6)
```

Time can be input as: \* A string of slash separated values: “2085-01-01T12:00:00Z/2120-12-30T12:00:00Z” \* A sequence of strings: e.g. (“2085-01-01T12:00:00Z”, “2120-12-30T12:00:00Z”)

Time inputs should be the start and end of the time range you want to subset over

```
[32]: time = roocs_utils.TimeParameter("2085-01-01T12:00:00Z/2120-12-30T12:00:00Z")

# the first and last time in the range provided can be returned in a dictionary
```

(continues on next page)



(continued from previous page)

```
print(time.asdict())

# the values can be returned as a tuple
print(time.tuple)

{'start_time': '2085-01-01T12:00:00+00:00', 'end_time': '2120-12-30T12:00:00+00:00'}
('2085-01-01T12:00:00+00:00', '2120-12-30T12:00:00+00:00')
```

Parameterise parameterises inputs to instances of parameter classes which allows them to be used throughout roocs.

```
[33]: roocs_utils.parameter.parameterise("cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.
↪rlilpl.latest.zostoga", "0.,49.,10.,65", (1000.50, 2000.60), "2085-01-01T12:00:00Z/
↪2120-12-30T12:00:00Z")

[33]: {'collection': Datasets to analyse:
cmip5.output1.INM.inmcm4.rcp45.mon.ocean.Omon.rlilpl.latest.zostoga,
'area': Area to subset over:
(0.0, 49.0, 10.0, 65.0),
'level': Level range to subset over
first_level: 1000.5
last_level: 2000.6,
'time': Time period to subset over
start time: 2085-01-01T12:00:00+00:00
end time: 2120-12-30T12:00:00+00:00}
```

## 5.2 Xarray utils

Xarray utils can be used to identify the main variable in a dataset as well as identifying the type of a coordinate or returning a coordinate based on an attribute or a type

```
[34]: from roocs_utils.xarray_utils import xarray_utils as xu
import xarray as xr

[35]: ds = xr.open_mfdataset("../tests/mini-esgf-data/test_data/badc/cmip5/data/cmip5/
↪output1/MOHC/HadGEM2-ES/rcp85/mon/atmos/Amon/rlilpl/latest/tas/*.nc", use_
↪cftime=True, combine="by_coords")

[36]: # find the main variable of the dataset
main_var = xu.get_main_variable(ds)

print("main var =", main_var)

ds[main_var]

main var = tas

[36]: <xarray.DataArray 'tas' (time: 3530, lat: 2, lon: 2)>
dask.array<concatenate, shape=(3530, 2, 2), dtype=float32, chunksize=(300, 2, 2),
↪chunktype=numpy.ndarray>
Coordinates:
  height    float64 1.5
  * lat     (lat) float64 -90.0 35.0
  * lon     (lon) float64 0.0 187.5
  * time    (time) object 2005-12-16 00:00:00 ... 2299-12-16 00:00:00
Attributes:
```

(continues on next page)

(continued from previous page)

```

standard_name:      air_temperature
long_name:          Near-Surface Air Temperature
comment:            near-surface (usually, 2 meter) air temperature.
units:              K
original_name:      mo: m01s03i236
cell_methods:       time: mean
cell_measures:      area: areacella
history:            2010-12-04T13:50:30Z altered by CMOR: Treated scalar d...
associated_files:    baseUrl: http://cmip-pcmdi.llnl.gov/CMIP5/dataLocation...

```

```
[37]: # to get the coord types
```

```

for coord in ds.coords:
    print("\ncoord name =", coord, "\ncoord type =", xu.get_coord_type(ds[coord]))

print("\n There is a level, time, latitude and longitude coordinate in this dataset")

```

```

coord name = height
coord type = level

```

```

coord name = lat
coord type = latitude

```

```

coord name = lon
coord type = longitude

```

```

coord name = time
coord type = time

```

```

There is a level, time, latitude and longitude coordinate in this dataset

```

```
[38]: # to check the type of a coord
```

```

print(xu.is_level(ds["height"]))
print(xu.is_latitude(ds["lon"]))

```

```

True
None

```

```
[39]: # to find a coordinate of a specific type
```

```
print("time =", xu.get_coord_by_type(ds, "time"))
```

```
# to find the level coordinate, set ignore_aux_coords to False
```

```
print("\nlevel =", xu.get_coord_by_type(ds, "level", ignore_aux_coords=False))
```

```

time = <xarray.DataArray 'time' (time: 3530)>
array([cftime.Datetime360Day(2005, 12, 16, 0, 0, 0, 0),
       cftime.Datetime360Day(2006, 1, 16, 0, 0, 0, 0),
       cftime.Datetime360Day(2006, 2, 16, 0, 0, 0, 0), ...,
       cftime.Datetime360Day(2299, 10, 16, 0, 0, 0, 0),
       cftime.Datetime360Day(2299, 11, 16, 0, 0, 0, 0),
       cftime.Datetime360Day(2299, 12, 16, 0, 0, 0, 0)], dtype=object)
Coordinates:
  height    float64 1.5

```

(continues on next page)

(continued from previous page)

```

* time      (time) object 2005-12-16 00:00:00 ... 2299-12-16 00:00:00
Attributes:
  bounds:      time_bnds
  axis:        T
  long_name:    time
  standard_name: time

level = <xarray.DataArray 'height' ()>
array(1.5)
Coordinates:
  height      float64 1.5
Attributes:
  units:      m
  axis:      Z
  positive:    up
  long_name:    height
  standard_name: height

```

```
[40]: # to find a coordinate based on an attribute you expect it to have
```

```
xu.get_coord_by_attr(ds, "standard_name", "latitude")
```

```

[40]: <xarray.DataArray 'lat' (lat: 2)>
array([-90.,  35.])
Coordinates:
  height      float64 1.5
  * lat        (lat) float64 -90.0 35.0
Attributes:
  bounds:      lat_bnds
  units:      degrees_north
  axis:      Y
  long_name:    latitude
  standard_name: latitude

```

## 5.3 Other utilities

Other utilities allow parsing a memory size of any unit into bytes and converting a time object into an ISO 8601 string

```

[41]: from roocs_utils.utils.common import parse_size
      from roocs_utils.utils.time_utils import to_isoformat
      from datetime import datetime

```

```

[42]: # to parse a size into bytes
      size = '50MiB'
      size_in_b = parse_size(size)
      size_in_b

```

```
[42]: 52428800.0
```

```

[43]: # to convert a time object into a time string
      time = datetime(2005, 7, 14, 12, 30)
      time_str = to_isoformat(time)
      time_str

```

```
[43]: '2005-07-14T12:30:00'
```

## 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/roocs-utils/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

roocs-utils could always use more documentation, whether as part of the official roocs-utils 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/roocs-utils/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 :)

### 6.1.6 Get Started!

Ready to contribute? Here's how to set up `roocs-utils` for local development.

#. Fork the `roocs-utils` repo on GitHub. #.

Clone your fork locally:

```
$ git clone git@github.com:your_name_here/roocs-utils.git
```

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

```
$ mkvirtualenv roocs-utils $ cd roocs-utils/ $ python setup.py develop
```

2. Create a branch for local development:

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

Now you can make your changes locally.

3. When you are done making changes, check that your changes pass `flake8` and the tests, including testing other Python versions with `tox`:

```
$ flake8 roocs-utils tests $ python setup.py test or py.test $ tox
```

To get `flake8` and `tox`, just `pip` install them into your virtualenv.

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

```
$ git add . $ git commit -m "Your detailed description of your changes." $ git push origin name-of-your-bugfix-or-feature
```

5. Submit a pull request through the GitHub website.

## 6.2 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in `README.md`.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check [https://travis-ci.com/github/roocs/roocs-utils/pull\\_requests](https://travis-ci.com/github/roocs/roocs-utils/pull_requests) and make sure that the tests pass for all supported Python versions.

## 6.3 Tips

To run a subset of tests:

```
$ py.test tests.test_roocs_utils
```

## 6.4 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.md). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.





## CREDITS

### 7.1 Lead

- Ag Stephens [ag.stephens@stfc.ac.uk](mailto:ag.stephens@stfc.ac.uk)

### 7.2 Contributors

- Eleanor Smith [eleanor.smith@stfc.ac.uk](mailto:eleanor.smith@stfc.ac.uk)
- Carsten Ehbrecht [ehbrecht@dkrz.de](mailto:ehbrecht@dkrz.de)



## VERSION HISTORY

### 8.1 v0.1.4 (2020-10-22)

Fixing pip install

#### 8.1.1 Bug Fixes

- Importing and using roocs-utils when pip installing now works

### 8.2 v0.1.3 (2020-10-21)

Fixing formatting of doc strings

#### 8.2.1 New Features

- Added a notebook to show examples

#### 8.2.2 Other Changes

- Updated formatting of doc strings

### 8.3 v0.1.2 (2020-10-15)

Updating the documentation and improving the changelog.

### 8.3.1 Other Changes

- Updated doc strings to improve documentation.
- Updated documentation.

## 8.4 v0.1.1 (2020-10-12)

Fixing mostly existing functionality to work more efficiently with the other packages in roocs.

### 8.4.1 Breaking Changes

- `environment.yml` has been updated to bring it in line with `requirements.txt`.
- `level` coordinates would previously have been identified as `None`. They are now identified as `level`.

### 8.4.2 New Features

- `parameterise` function added in `roocs_utils.parameter` to use in all roocs packages.
- `ROOCS_CONFIG` environment variable can be used to override default config in `etc/roocs.ini`. To use a local config file set `ROOCS_CONFIG` as the file path to this file. Several file paths can be provided separated by a `:`
- Inventory functionality added - this can be used to create an inventory of datasets. See `README` for more info.
- `project_utils` added with the following functions to get the project name of a dataset and the base directory for that project.
- `utils.common` and `utils.time_utils` added.
- `is_level` implemented in `xarray_utils` to identify whether a coordinate is a level or not.

### 8.4.3 Bug Fixes

- `xarray_utils.xarray_utils.get_main_variable` updated to exclude common coordinates from the search for the main variable. This fixes a bug where coordinates such as `lon_bounds` would be returned as the main variable.

### 8.4.4 Other Changes

- `README` update to explain inventory functionality.
- `Black` and `flake8` formatting applied.
- Fixed import warning with `collections.abc`.

## 8.5 v0.1.0 (2020-07-30)

- First release.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`