ecmwf$_{m}odelsDocumentation$

Release 0.3.post1.dev94+g574a227.d20211115

TU Wien

Nov 15, 2021

CONTENTS

1 Citation **3**

2 Installation **5**

3 Supported Products **7**

4 Contribute **9**

5 Downloading ERA5 Data **11**

6 Downloading ERA Interim Data **13**

7 Reading data **15**

8 Conversion to time series format **17**
8.1 Reading converted time series data 18

9 Contents **19**
9.1 Downloading ERA5 Data 19
9.2 Downloading ERA Interim Data 19
9.3 Reading data 20
9.4 Conversion to time series format 21
9.5 Reading converted time series data 22
9.6 License 22
9.7 Developers 22
9.8 Changelog 23
9.9 ecmwf_models 24

10 Indices and tables **39**

Python Module Index **41**

Index **43**

Readers and converters for data from the ECMWF reanalysis models. Written in Python.

Works great in combination with pytesmo.

CHAPTER

ONE

CITATION

If you use the software in a publication then please cite it using the Zenodo DOI. Be aware that this badge links to the latest package version.

Please select your specific version at https://doi.org/10.5281/zenodo.593533 to get the DOI of that version. You should normally always use the DOI for the specific version of your record in citations. This is to ensure that other researchers can access the exact research artefact you used for reproducibility.

You can find additional information regarding DOI versioning at http://help.zenodo.org/#versioning

CHAPTER

TWO

INSTALLATION

Install required C-libraries via conda. For installation we recommend Miniconda. So please install it according to the official installation instructions. As soon as you have the `conda` command in your shell you can continue:

```
conda install -c conda-forge pandas pygrib netcdf4 pyresample xarray
```

The following command will download and install all the needed pip packages as well as the ecmwf-model package itself.

```
pip install ecmwf_models
```

To create a full development environment with conda, the *yml* files inside the folder *environment/* in this repository can be used. Both environements should work. The file *latest* should install the newest version of most dependencies. The file *pinned* is a fallback option and should always work.

```
git clone --recursive git@github.com:TUW-GEO/ecmwf_models.git ecmwf_models
cd ecmwf_models
conda env create -f environment/latest.yml
source activate ecmwf_models
python setup.py develop
pytest
```

CHAPTER

THREE

SUPPORTED PRODUCTS

At the moment this package supports

- **ERA Interim** (deprecated)
- **ERA5**
- **ERA5-Land**

reanalysis data in **grib** and **netcdf** format (download, reading, time series creation) with a default spatial sampling of 0.75 degrees (ERA Interim), 0.25 degrees (ERA5), resp. 0.1 degrees (ERA5-Land). It should be easy to extend the package to support other ECMWF reanalysis products. This will be done as need arises.

CHAPTER

FOUR

CONTRIBUTE

We are happy if you want to contribute. Please raise an issue explaining what is missing or if you find a bug. Please take a look at the developers guide.

CHAPTER

FIVE

DOWNLOADING ERA5 DATA

ERA5 (and ERA5-Land) data can be downloaded manually from the Copernicus Data Store (CDS) or automatically via the CDS api, as done in the download module (era5_download). Before you can use this, you have to set up an account at the CDS and setup the CDS key.

Then you can use the program `era5_download` to download ERA5 images between a passed start and end date. `era5_download --help` will show additional information on using the command.

For example, the following command in your terminal would download ERA5 images for all available layers of soil moisture in netcdf format, between January 1st and February 1st 2000 in grib format into `/path/to/storage`. The data will be stored in subfolders of the format `YYYY/jjj`. The temporal resolution of the images is 6 hours by default.

```
era5_download /path/to/storage -s 2000-01-01 -e 2000-02-01 --variables swvl1 swvl2 swvl3␣
↪swvl4
```

The names of the variables to download can be its long names, the short names (as in the example) or the parameter IDs. We use the `era5_lut.csv` file to look up the right name for the CDS API. Other flags, that can be activated in `era5_download` are:

- **-h (–help)** : shows the help text for the download function
- **-p (–product)**: specify the ERA5 product to download. Choose either ERA5 or ERA5-Land. Default is ERA5.
- **-keep (–keep_original)** : keeps the originally downloaded files as well. We split the downloaded, monthly stacks into single images and discard the original files by default.
- **-grb (–as_grib)** [download the data in grib format instead of the default nc4] format (grib reading is not supported on Windows OS).
- **–h_steps** : full hours for which images are downloaded (e.g. –h_steps 0 would download only data at 00:00 UTC). By default we use 0, 6, 12 and 18.

CHAPTER

SIX

DOWNLOADING ERA INTERIM DATA

ERA-Interim has been decommissioned. Use ERA5 instead.

ERA-Interim data can be downloaded manually from the ECMWF servers. It can also be done automatically using the ECMWF API. To use the ECMWF API you have to be registered, install the ecmwf-api Python package and setup the ECMWF API Key. A guide for this is provided by ECMWF.

After that you can use the command line program `eraint_download` to download images with a temporal resoltuion of 6 hours between a passed start and end date. `eraint_download --help` will show additional information on using the command.

For example, the following command in your terminal would download ERA Interim soil moisture images of all available layers (see the Variable DB) in netcdf format on the default gaussian grid for ERA-Interim (0.75°x0.75°) into the folder `/path/to/storage` between January 1st and February 1st 2000. The data will be stored in subfolders of the format `YYYY/jjj`, where `YYYY` describes the year and `jjj` the day of the year for the downloaded files.

```
eraint_download /path/to/storage  -s 2000-01-01 -e 2000-02-01 --variables swvl1 swvl2␣
↪swvl3 swvl4
```

Additional optional parameters allow downloading images in netcdf format, and in a different spatial resolution (see the –help function and descriptions for downloading ERA5 data)

CHAPTER

SEVEN

READING DATA

After downloading the data for ERA Interim or ERA5 via `eraint_download` resp. `era5_download`, images can be read with the `ERA5GrbDs` and `ERA5NcDs` (for grib and netcdf image stacks), respectively the `ERA5GrbImg` and `ERA5NcImg` (for single grib and netcdf images) classes. The respective functions for reading images are defined in `ecmwf_models.erainterim.interface` `ecmwf_models.era5.interface`.

The following examples are shown for ERA5 data, but work the same way with the respective ERA Interim functions.

For example, you can read the image for a single variable at a specific date. In this case for a stack of downloaded image files:

```
# Script to load a stack of downloaded netcdf images
# and read a variable for a single date.
from ecmwf_models.era5.interface import ERA5NcDs
root_path = "/path/to/netcdf_storage"
ds = ERA5NcDs(root_path, parameter='swvl1')
data = ds.read(datetime(2010, 1, 1, 0))

# Script to load a stack of downloaded grib images
# and read a variable for a single date.
from ecmwf_models.era5.interface import ERA5GrbDs
root_path = "/path/to/grib_storage"
ds = ERA5GrbDs(root_path, parameter='swvl1')
data = ds.read(datetime(2010, 1, 1, 0))
```

You can also read multiple variables at a specific date by passing a list of parameters. In this case for a set of netcdf files:

```
# Script to load a stack of downloaded netcdf images
# and read two variables for a single date.
from ecmwf_models.era5.interface import ERA5NcDs
root_path = "/path/to/storage"
ds = ERA5NcDs(root_path, parameter=['swvl1', 'swvl2'])
data = ds.read(datetime(2000, 1, 1, 0))
```

All images between two given dates can be read using the `iter_images` methods of all the image stack reader classes.

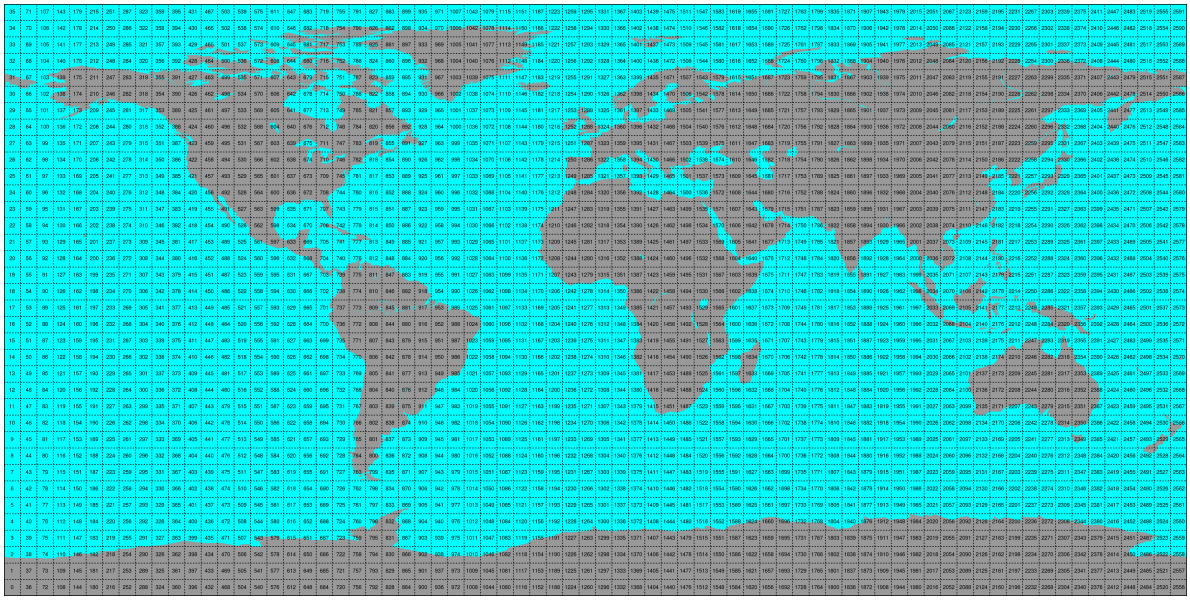
CHAPTER

EIGHT

CONVERSION TO TIME SERIES FORMAT

For a lot of applications it is favorable to convert the image based format into a format which is optimized for fast time series retrieval. This is what we often need for e.g. validation studies. This can be done by stacking the images into a netCDF file and choosing the correct chunk sizes or a lot of other methods. We have chosen to do it in the following way:

- Store only the reduced gaußian grid points (for grib data) since that saves space.
- Store the time series in netCDF4 in the Climate and Forecast convention Orthogonal multidimensional array representation
- Store the time series in 5x5 degree cells. This means there will be 2566 cell files and a file called `grid.nc` which contains the information about which grid point is stored in which file. This allows us to read a whole 5x5 degree area into memory and iterate over the time series quickly.



This conversion can be performed using the `era5_reshuffle` (respectively `eraint_reshuffle`) command line program. An example would be:

```
era5_reshuffle /era_data /timeseries/data 2000-01-01 2001-01-01 swvl1 swvl2
```

Which would take 6-hourly ERA5 images stored in `/era_data` from January 1st 2000 to January 1st 2001 and store the parameters "swvl1" and "swvl2" as time series in the folder `/timeseries/data`. If you time series should have a different resolution than 6H, use the `h_steps` flag here accordingly (images to use for time series generation have to be in the downloaded raw data). The passed names have to correspond with the names in the downloaded file, i.e. use the variable short names here. Other flags, that can be used in `era5_reshuffle` are:

- **-h (–help)** : Shows the help text for the reshuffle function
- **–land_points** : Reshuffle and store only data over land points.
- **-h_steps (–as_grib)** : full hours for which images are reshuffled (e.g. –h_steps 0 would reshuffle only data at 00:00 UTC). By default we use 0, 6, 12 and 18.
- **–imgbuffer** : The number of images that are read into memory before converting them into time series. Bigger numbers make the conversion faster but consume more memory.

Conversion to time series is performed by the repurpose package in the background. For custom settings or other options see the repurpose documentation and the code in `ecmwf_models.reshuffle`.

8.1 Reading converted time series data

For reading time series data, that the `era5_reshuffle` and `eraint_reshuffle` command produces, the class `ERATs` can be used. Optional arguments that are passed to the parent class (`OrthoMultiTs`, as defined in pynetcf.time_series) can be passed as well:

```
from ecmwf_models import ERATs
# read_bulk reads full files into memory
# read_ts takes either lon, lat coordinates to perform a nearest neighbour search
# or a grid point index (from the grid.nc file) and returns a pandas.DataFrame.
ds = ERATs(ts_path, ioclass_kws={'read_bulk': True})
ts = ds.read_ts(45, 15)
```

Bulk reading speeds up reading multiple points from a cell file by storing the file in memory for subsequent calls. Either Longitude and Latitude can be passed to perform a nearest neighbour search on the data grid (`grid.nc` in the time series path) or the grid point index (GPI) can be passed directly.

CHAPTER

NINE

CONTENTS

9.1 Downloading ERA5 Data

ERA5 (and ERA5-Land) data can be downloaded manually from the Copernicus Data Store (CDS) or automatically via the CDS api, as done in the download module (era5_download). Before you can use this, you have to set up an account at the CDS and setup the CDS key.

Then you can use the program `era5_download` to download ERA5 images between a passed start and end date. `era5_download --help` will show additional information on using the command.

For example, the following command in your terminal would download ERA5 images for all available layers of soil moisture in netcdf format, between January 1st and February 1st 2000 in grib format into `/path/to/storage`. The data will be stored in subfolders of the format `YYYY/jjj`. The temporal resolution of the images is 6 hours by default.

```
era5_download /path/to/storage -s 2000-01-01 -e 2000-02-01 --variables swvl1 swvl2 swvl3␣
↪swvl4
```

The names of the variables to download can be its long names, the short names (as in the example) or the parameter IDs. We use the `era5_lut.csv` file to look up the right name for the CDS API. Other flags, that can be activated in `era5_download` are:

- **-h (–help)** : shows the help text for the download function
- **-p (–product)**: specify the ERA5 product to download. Choose either ERA5 or ERA5-Land. Default is ERA5.
- **-keep (–keep_original)** : keeps the originally downloaded files as well. We split the downloaded, monthly stacks into single images and discard the original files by default.
- **-grb (–as_grib)** [download the data in grib format instead of the default nc4] format (grib reading is not supported on Windows OS).
- **–h_steps** : full hours for which images are downloaded (e.g. –h_steps 0 would download only data at 00:00 UTC). By default we use 0, 6, 12 and 18.

9.2 Downloading ERA Interim Data

ERA-Interim has been decommissioned. Use ERA5 instead.

ERA-Interim data can be downloaded manually from the ECMWF servers. It can also be done automatically using the ECMWF API. To use the ECMWF API you have to be registered, install the ecmwf-api Python package and setup the ECMWF API Key. A guide for this is provided by ECMWF.

After that you can use the command line program `eraint_download` to download images with a temporal resoltuion of 6 hours between a passed start and end date. `eraint_download --help` will show additional information on using the command.

For example, the following command in your terminal would download ERA Interim soil moisture images of all available layers (see the Variable DB) in netcdf format on the default gaussian grid for ERA-Interim (0.75°x0.75°) into the folder `/path/to/storage` between January 1st and February 1st 2000. The data will be stored in subfolders of the format `YYYY/jjj`, where `YYYY` describes the year and `jjj` the day of the year for the downloaded files.

```
eraint_download /path/to/storage  -s 2000-01-01 -e 2000-02-01 --variables swvl1 swvl2␣
↪swvl3 swvl4
```

Additional optional parameters allow downloading images in netcdf format, and in a different spatial resolution (see the –help function and descriptions for downloading ERA5 data)

9.3 Reading data

After downloading the data for ERA Interim or ERA5 via `eraint_download` resp. `era5_download`, images can be read with the `ERA5GrbDs` and `ERA5NcDs` (for grib and netcdf image stacks), respectively the `ERA5GrbImg` and `ERA5NcImg` (for single grib and netcdf images) classes. The respective functions for reading images are defined in `ecmwf_models.erainterim.interface` `ecmwf_models.era5.interface`.

The following examples are shown for ERA5 data, but work the same way with the respective ERA Interim functions.

For example, you can read the image for a single variable at a specific date. In this case for a stack of downloaded image files:

```
# Script to load a stack of downloaded netcdf images
# and read a variable for a single date.
from ecmwf_models.era5.interface import ERA5NcDs
root_path = "/path/to/netcdf_storage"
ds = ERA5NcDs(root_path, parameter='swvl1')
data = ds.read(datetime(2010, 1, 1, 0))

# Script to load a stack of downloaded grib images
# and read a variable for a single date.
from ecmwf_models.era5.interface import ERA5GrbDs
root_path = "/path/to/grib_storage"
ds = ERA5GrbDs(root_path, parameter='swvl1')
data = ds.read(datetime(2010, 1, 1, 0))
```

You can also read multiple variables at a specific date by passing a list of parameters. In this case for a set of netcdf files:

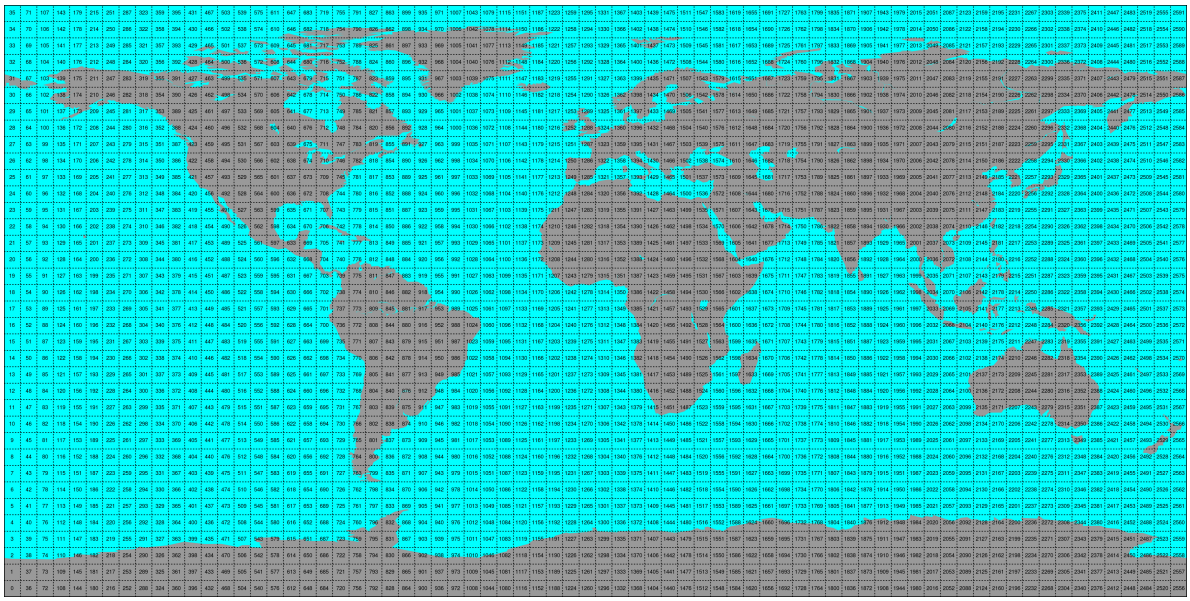
```
# Script to load a stack of downloaded netcdf images
# and read two variables for a single date.
from ecmwf_models.era5.interface import ERA5NcDs
root_path = "/path/to/storage"
ds = ERA5NcDs(root_path, parameter=['swvl1', 'swvl2'])
data = ds.read(datetime(2000, 1, 1, 0))
```

All images between two given dates can be read using the `iter_images` methods of all the image stack reader classes.

9.4 Conversion to time series format

For a lot of applications it is favorable to convert the image based format into a format which is optimized for fast time series retrieval. This is what we often need for e.g. validation studies. This can be done by stacking the images into a netCDF file and choosing the correct chunk sizes or a lot of other methods. We have chosen to do it in the following way:

- Store only the reduced gaußian grid points (for grib data) since that saves space.
- Store the time series in netCDF4 in the Climate and Forecast convention Orthogonal multidimensional array representation
- Store the time series in 5x5 degree cells. This means there will be 2566 cell files and a file called `grid.nc` which contains the information about which grid point is stored in which file. This allows us to read a whole 5x5 degree area into memory and iterate over the time series quickly.



This conversion can be performed using the `era5_reshuffle` (respectively `eraint_reshuffle`) command line program. An example would be:

```
era5_reshuffle /era_data /timeseries/data 2000-01-01 2001-01-01 swvl1 swvl2
```

Which would take 6-hourly ERA5 images stored in `/era_data` from January 1st 2000 to January 1st 2001 and store the parameters "swvl1" and "swvl2" as time series in the folder `/timeseries/data`. If you time series should have a different resolution than 6H, use the `h_steps` flag here accordingly (images to use for time series generation have to be in the downloaded raw data). The passed names have to correspond with the names in the downloaded file, i.e. use the variable short names here. Other flags, that can be used in `era5_reshuffle` are:

- **-h (–help)** : Shows the help text for the reshuffle function
- **–land_points** : Reshuffle and store only data over land points.
- **-h_steps (–as_grib)** : full hours for which images are reshuffled (e.g. –h_steps 0 would reshuffle only data at 00:00 UTC). By default we use 0, 6, 12 and 18.
- **–imgbuffer** : The number of images that are read into memory before converting them into time series. Bigger numbers make the conversion faster but consume more memory.

Conversion to time series is performed by the repurpose package in the background. For custom settings or other options see the repurpose documentation and the code in `ecmwf_models.reshuffle`.

9.5 Reading converted time series data

For reading time series data, that the `era5_reshuffle` and `eraint_reshuffle` command produces, the class `ERATs` can be used. Optional arguments that are passed to the parent class (`OrthoMultiTs`, as defined in pynetcf.time_series) can be passed as well:

```
from ecmwf_models import ERATs
# read_bulk reads full files into memory
# read_ts takes either lon, lat coordinates to perform a nearest neighbour search
# or a grid point index (from the grid.nc file) and returns a pandas.DataFrame.
ds = ERATs(ts_path, ioclass_kws={'read_bulk': True})
ts = ds.read_ts(45, 15)
```

Bulk reading speeds up reading multiple points from a cell file by storing the file in memory for subsequent calls. Either Longitude and Latitude can be passed to perform a nearest neighbour search on the data grid (`grid.nc` in the time series path) or the grid point index (GPI) can be passed directly.

9.6 License

9.7 Developers

- Sebastian Hahn <sebastian.hahn@geo.tuwien.ac.at>
- Christoph Paulik <cpaulik@vandersat.com>
- Wolfgang Preimesberger <wolfgang.preimesberger@geo.tuwien.ac.at>

9.8 Changelog

9.8.1 Unreleased

-

9.8.2 Version 0.8

- Program *era5_download* returns exit code now (PR #27);
- Program *era5_reshuffle* can now take a bounding box to reshuffle spatial subsets;
- TravisCI was replaced by Github Actions;
- Pyscaffold 4 is used; contributing guide added; pre-commit added;
- Code formatting with black (line length 79);

9.8.3 Version 0.7

- Update pyscaffold structure
- Drop support for python2
- Travis deploy to pypi

9.8.4 Version 0.6.1

- Fix bug when creating 0.1 deg grid cells (floating point precision)
- Missing variables in grib files are now replaced by empty images.
- Read variable names from grib files from cfVarNameECMF instead of short_name field

9.8.5 Version 0.6

- Add support for downloading, reading, reshuffling era5-land
- Add support for reading, reshuffling points over land only (era5 and era5-land)
- Add function to create land definition files
- Test with pinned environments

9.8.6 Version 0.5

- Change default time steps to 6 hours.
- Add more tests, also for download functions
- Update documentation, add installation script
- Fix bugs, update command line interfaces, update dependencies
- Separate download programs for ERA5 and ERA Interim
- Change the ERA5 download api to use cdsapi instead of ecmwf api

- Update package structure to better separate between the ERA products
- Add look-up-table file for more flexibility in variable names passed by user
- Update readme

9.8.7 Version 0.4

- Add ERA5 support (download, reading, TS conversion)
- Add netcdf support for ERA5 and ERA-Interim download (regular grid)
- Add new grid defintions: netcdf download in regular grid, grib in gaussian grid
- Add Download with spatial resampling (grib and nc)
- Update GRIB message storing (per day instead of per message)
- Add tests for splitting downloaded files, ERA5 reading, ERA5 reshuffling, generated grids
- Add new test data

9.8.8 Version 0.3

- Fix help text in ecmwf_repurpose command line program.
- Fix reading of metadata for variables that do not have 'levels'
- Fix wrong import when trying to read the reformatted time series data.

9.8.9 Version 0.2

- Add reading of basic metadata fields name, depth and units.
- Fix reading of latitudes and longitudes - where flipped before.
- Fix longitude range to -180, 180.
- Add conversion to time series format.

9.8.10 Version 0.1

- First version
- Add ERA Interim support for downloading and reading.

9.9 ecmwf_models

9.9.1 ecmwf_models package

Subpackages

ecmwf_models.era5 package

Submodules

ecmwf_models.era5.download module

Module to download ERA5 from terminal in netcdf and grib format.

class ecmwf_models.era5.download.**CDSStatusTracker**
Bases: object

Track the status of the CDS download by using the CDS callback functions

handle_error_function(**args*, ***kwargs*)

statuscode_error = -1

statuscode_ok = 0

statuscode_unavailable = 10

ecmwf_models.era5.download.**default_variables**(*product='era5'*)
These variables are being downloaded, when None are passed by the user

Parameters **product** (*str, optional (default: 'era5')*) – Name of the era5 product to read the default variables for. Either 'era5' or 'era5-land'.

ecmwf_models.era5.download.**download_and_move**(*target_path*, *startdate*, *enddate*, *product='era5'*, *variables=None*, *keep_original=False*, *h_steps=[0, 6, 12, 18]*, *grb=False*, *dry_run=False*, *grid=None*, *remap_method='bil'*, *cds_kwds={}*, *stepsize='month'*) → int

Downloads the data from the ECMWF servers and moves them to the target path. This is done in 30 day increments between start and end date.

The files are then extracted into separate grib files per parameter and stored in yearly folders under the target_path.

Parameters

- **target_path** (*str*) – Path where the files are stored to
- **startdate** (*datetime*) – first date to download
- **enddate** (*datetime*) – last date to download
- **product** (*str, optional (default: ERA5)*) – Either ERA5 or ERA5Land
- **variables** (*list, optional (default: None)*) – Name of variables to download
- **keep_original** (*bool*) – keep the original downloaded data
- **h_steps** (*list*) – List of full hours to download data at the selected dates e.g [0, 12]
- **grb** (*bool, optional (default: False)*) – Download data as grib files
- **dry_run** (*bool*) – Do not download anything, this is just used for testing the functions
- **grid** (*dict, optional*) – A grid on which to remap the data using CDO. This must be a dictionary using CDO's grid description format, e.g.:

```
grid = {
    "gridtype": "lonlat",
    "xsize": 720,
    "ysize": 360,
    "xfirst": -179.75,
```

(continues on next page)

(continued from previous page)

```
    "yfirst": 89.75,
    "xinc": 0.5,
    "yinc": -0.5,
}
```

Default is to use no regridding.

- **remap_method** (*str, optional*) – Method to be used for regridding. Available methods are: - "bil": bilinear (default) - "bic": bicubic - "nn": nearest neighbour - "dis": distance weighted - "con": 1st order conservative remapping - "con2": 2nd order conservative remapping - "laf": largest area fraction remapping
- **cds_kwds** (*dict, optional*) – Additional arguments to be passed to the CDS API retrieve request.
- **stepsize** (*str, optional*) – Size of steps for requests, can be "month" or "day".

Returns **status_code** – 0 : Downloaded data ok -1 : Error -10 : No data available for requested time period

Return type int

ecmwf_models.era5.download.**download_era5**(*c*, *years*, *months*, *days*, *h_steps*, *variables*, *target*, *grb=False*, *product='era5'*, *dry_run=False*, *cds_kwds={}*)

Download era5 reanalysis data for single levels of a defined time span

Parameters

- **c** (*cdsapi.Client*) – Client to pass the request to
- **years** (*list*) – Years for which data is downloaded ,e.g. [2017, 2018]
- **months** (*list*) – Months for which data is downloaded, e.g. [4, 8, 12]
- **days** (*list*) – Days for which data is downloaded (range(31)=All days) e.g. [10, 20, 31]
- **h_steps** (*list*) – List of full hours to download data at the selected dates e.g [0, 12]
- **variables** (*list, optional (default: None)*) – List of variables to pass to the client, if None are passed, the default variables will be downloaded.
- **target** (*str*) – File name, where the data is stored.
- **geb** (*bool, optional (default: False)*) – Download data in grib format
- **product** (*str*) – ERA5 data product to download, either era5 or era5-land
- **dry_run** (*bool, optional (default: False)*) – Do not download anything, this is just used for testing the functionality
- **cds_kwds** (*dict, optional*) – Additional arguments to be passed to the CDS API retrieve request.

Returns **success** – Return True after downloading finished

Return type bool

ecmwf_models.era5.download.**main**(*args*)

ecmwf_models.era5.download.**parse_args**(*args*)

Parse command line parameters for recursive download

Parameters **args** (*list*) – Command line parameters as list of strings

Returns **clparams** – Parsed command line parameters

Return type argparse.Namespace

ecmwf_models.era5.download.**run**()

ecmwf_models.era5.interface module

This module contains ERA5/ERA5-Land specific child classes of the netcdf and grib base classes, that are used for reading all ecmwf products.

class ecmwf_models.era5.interface.**ERA5GrbDs**(*root_path: str*, *parameter: Tuple[str, ...] = ('swvl1', 'swvl2')*, *h_steps: Tuple[int, ...] = (0, 6, 12, 18)*, *product: Literal['era5', 'era5-land'] = 'era5'*, *subgrid: Optional[pygeogrids.grids.CellGrid] = None*, *mask_seapoints: Optional[bool] = False*, *array_1D: Optional[bool] = False*)

Bases: *ecmwf_models.interface.ERAGrbDs*

class ecmwf_models.era5.interface.**ERA5GrbImg**(*filename: str*, *parameter: Optional[Tuple[str, ...]] = ('swvl1', 'swvl2')*, *subgrid: Optional[pygeogrids.grids.CellGrid] = None*, *mask_seapoints: Optional[bool] = False*, *array_1D=False*)

Bases: *ecmwf_models.interface.ERAGrbImg*

class ecmwf_models.era5.interface.**ERA5NcDs**(*root_path: str*, *parameter: Tuple[str, ...] = ('swvl1', 'swvl2')*, *product: Literal['era5', 'era5-land'] = 'era5'*, *h_steps: Tuple[int, ...] = (0, 6, 12, 18)*, *subgrid: Optional[pygeogrids.grids.CellGrid] = None*, *mask_seapoints: Optional[bool] = False*, *array_1D: Optional[bool] = False*)

Bases: *ecmwf_models.interface.ERANcDs*

Reader for a stack of ERA5 netcdf image files.

Parameters

- **root_path** (*str*) – Path to the image files to read.
- **parameter** (*list or str, optional (default: ('swvl1', 'swvl2'))*) – Name of parameters to read from the image file.
- **product** (*str, optional (default: 'era5')*) – What era5 product, either era5 or era5-land.
- **h_steps** (*list, optional (default: (0,6,12,18))*) – List of full hours to read images for.
- **subgrid** (*pygeogrids.CellGrid, optional (default: None)*) – Read only data for points of this grid and not global values.
- **mask_seapoints** (*bool, optional (default: False)*) – Read the land-sea mask to mask points over water and set them to nan. This option needs the 'lsm' parameter to be in the file!
- **array_1D** (*bool, optional (default: False)*) – Read data as list, instead of 2D array, used for reshuffling.

class ecmwf_models.era5.interface.**ERA5NcImg**(*filename: str*, *parameter: Optional[Tuple[str, ...]] = ('swvl1', 'swvl2')*, *product: Literal['era5', 'era5-land'] = 'era5'*, *subgrid: Optional[pygeogrids.grids.CellGrid] = None*, *mask_seapoints: Optional[bool] = False*, *array_1D: Optional[bool] = False*)

Bases: *ecmwf_models.interface.ERANcImg*

ecmwf_models.era5.reshuffle module

Module for a command line interface to convert the ERA Interim data into a time series format using the repurpose package

ecmwf_models.era5.reshuffle.**main**(*args*)

ecmwf_models.era5.reshuffle.**parse_args**(*args*)

Parse command line parameters for conversion from image to time series.

Parameters **args** (*list*) – command line parameters as list of strings

Returns **args** – Parsed command line parameters

Return type argparse.Namespace

ecmwf_models.era5.reshuffle.**reshuffle**(*input_root*, *outputpath*, *startdate*, *enddate*, *variables*, *product=None*, *bbox=None*, *h_steps=(0, 6, 12, 18)*, *land_points=False*, *imgbuffer=50*)

Reshuffle method applied to ERA images for conversion into netcdf time series format.

Parameters

- **input_root** (*str*) – Input path where ERA image data was downloaded to.
- **outputpath** (*str*) – Output path, where the reshuffled netcdf time series are stored.
- **startdate** (*datetime*) – Start date, from which images are read and time series are generated.
- **enddate** (*datetime*) – End date, from which images are read and time series are generated.
- **variables** (*tuple or list or str*) – Variables to read from the passed images and convert into time series format.
- **product** (*str, optional (default: None)*) – Either era5 or era5-land, if None is passed we guess the product from the downloaded image files.
- **bbox** (*tuple optional (default: None)*) – (min_lon, min_lat, max_lon, max_lat) - wgs84. To load only a subset of the global grid / file.
- **h_steps** (*list or tuple, optional (default: (0, 6, 12, 18))*) – Hours at which images are read for each day and used for reshuffling, therefore this defines the sub-daily temporal resolution of the time series that are generated.
- **land_points** (*bool, optional (default: False)*) – Reshuffle only land points. Uses the ERA5 land mask to create a land grid. The land grid is fixed to 0.25*0.25 or 0.1*0.1 deg for now.
- **imgbuffer** (*int, optional (default: 200)*) – How many images to read at once before writing time series. This number affects how many images are stored in memory and should be chosen according to the available amount of memory and the size of a single image.

ecmwf_models.era5.reshuffle.**run**()

Module contents

ecmwf_models.erainterim package

Submodules

ecmwf_models.erainterim.download module

Module to download ERA Interim from terminal.

ecmwf_models.erainterim.download.**default_variables**() → list
These variables are being downloaded, when None are passed by the user

ecmwf_models.erainterim.download.**download_and_move**(*target_path*, *startdate*, *enddate*, *variables=None*, *keep_original=False*, *grid_size=None*, *type='an'*, *h_steps=(0, 6, 12, 18)*, *steps=(0,)*, *grb=False*, *dry_run=False*)
Downloads the data from the ECMWF servers and moves them to the target path. This is done in 30 days increments between start and end date to be efficient with the MARS system. See the recommendation for doing it this way in https://software.ecmwf.int/wiki/display/WEBAPI/ERA-Interim+daily+retrieval+efficiency

The files are then extracted into separate grib/nc files and stored in yearly folders under the target_path.

Parameters

- **target_path** (*str*) – Path to which to copy the extracted parameter files
- **startdate** (*datetime*) – First date to download
- **enddate** (*datetime*) – Last date to download
- **variables** (*list, optional (default: None)*) – List of variable ids to pass to the client, if None are passed, the default variable ids will be downloaded.
- **keep_original** (*bool, optional (default: False)*) – Keep the original downloaded data
- **grid_size** (*list, optional (default: None)*) – [lon, lat] extent of the grid (regular for netcdf, at lat=0 for grib) If None is passed, the default grid size for the data product is used.
- **type** (*str, optional (default: 'an')*) – Data stream, model to download data for (fc=forecase)
- **h_steps** (*list, optional (default: [0, 6, 12, 18])*) – List of full hours to download data at the selected dates
- **grb** (*bool, optional (default: False)*) – Download data as grib files instead of netcdf files
- **dry_run** (*bool*) – Do not download anything, this is just used for testing the functions

ecmwf_models.erainterim.download.**download_eraint**(*target_path*, *start*, *end*, *variables*, *grid_size=None*, *type='fc'*, *h_steps=(0, 6, 12, 18)*, *grb=False*, *dry_run=False*, *steps=(0,)*)
Download era interim data

Parameters

- **target_path** (*str*) – path at which to save the downloaded grib file
- **start** (*date*) – start date

- **end** (*date*) – end date
- **variables** (*list*) – parameter ids, see wiki
- **product** (*str, optional*) – Name of the model, "ERA-interim" (default) or "ERA5"
- **grid_size** (*[float,float], optional*) – size of the grid in form (lon, lat), which the data is resampled to If None is passed the minimum grid for the accoring product is chosen
- **h_steps** (*tuple, optional (default: (0, 6, 12, 18))*) – List of full hours to download data at the selected dates
- **grb** (*bool, optional (default: False)*) – Download data as grb files instead of nc files
- **dry_run** (*bool*) – Do not download anything, this is just used for testing the functions

ecmwf_models.erainterim.download.**main**(*args*)

ecmwf_models.erainterim.download.**parse_args**(*args*)
Parse command line parameters for recursive download

Parameters **args** (*list*) – Command line parameters as list of strings

Returns **clparams** – Parsed command line parameters

Return type argparse.Namespace

ecmwf_models.erainterim.download.**run**()

ecmwf_models.erainterim.interface module

This module contains ERA Interim specific child classes of the netcdf and grib base classes, that are used for reading all ecmwf products.

class ecmwf_models.erainterim.interface.**ERAIntGrbDs**(*root_path: str, parameter: Tuple[str, ...] = ('swvl1', 'swvl2'), subgrid: Optional[pygeogrids.grids.CellGrid] = None, mask_seapoints: Optional[bool] = False, h_steps: Tuple[int, ...] = (0, 6, 12, 18), array_1D: Optional[bool] = False*)
Bases: *ecmwf_models.interface.ERAGrbDs*

class ecmwf_models.erainterim.interface.**ERAIntGrbImg**(*filename: str, parameter: Optional[Tuple[str, ...]] = ('swvl1', 'swvl2'), mode: Optional[str] = 'r', subgrid: Optional[pygeogrids.grids.CellGrid] = None, mask_seapoints: Optional[bool] = False, array_1D: Optional[bool] = False*)
Bases: *ecmwf_models.interface.ERAGrbImg*

class ecmwf_models.erainterim.interface.**ERAIntNcDs**(*root_path: str, parameter: Optional[Tuple[str, ...]] = ('swvl1', 'swvl2'), subgrid: Optional[pygeogrids.grids.CellGrid] = None, mask_seapoints: Optional[bool] = False, h_steps: Tuple[int, ...] = (0, 6, 12, 18), array_1D: Optional[bool] = False*)
Bases: *ecmwf_models.interface.ERANcDs*

class ecmwf_models.erainterim.interface.**ERAIntNcImg**(*filename: str*, *parameter: Optional[Tuple[str, ...]] = ('swvl1', 'swvl2')*, *mode: Optional[str] = 'r'*, *subgrid: Optional[pygeogrids.grids.CellGrid] = None*, *mask_seapoints: Optional[bool] = False*, *array_1D: Optional[bool] = False*)

Bases: *ecmwf_models.interface.ERANcImg*

ecmwf_models.erainterim.reshuffle module

Module for a command line interface to convert the ERA Interim data into a time series format using the repurpose package

ecmwf_models.erainterim.reshuffle.**main**(*args*)

ecmwf_models.erainterim.reshuffle.**parse_args**(*args*)

Parse command line parameters for conversion from image to time series.

Parameters **args** (*list*) – command line parameters as list of strings

Returns **args** – Parsed command line parameters

Return type argparse.Namespace

ecmwf_models.erainterim.reshuffle.**reshuffle**(*input_root*, *outputpath*, *startdate*, *enddate*, *variables*, *mask_seapoints=False*, *h_steps=(0, 6, 12, 18)*, *imgbuffer=50*)

Reshuffle method applied to ERA images for conversion into netcdf time series format.

Parameters

- **input_root** (*str*) – Input path where ERA image data was downloaded to.
- **outputpath** (*str*) – Output path, where the reshuffled netcdf time series are stored.
- **startdate** (*datetime*) – Start date, from which images are read and time series are generated.
- **enddate** (*datetime*) – End date, from which images are read and time series are generated.
- **variables** (*list or str or tuple*) – Variables to read from the passed images and convert into time series format.
- **mask_seapoints** (*bool, optional (default: False)*) – Mask points over sea, replace them with nan.
- **h_steps** (*tuple, optional (default: (0,6,12,18))*) – Full hours for which images are available.
- **imgbuffer** (*int, optional (default: 50)*) – How many images to read at once before writing time series. This number affects how many images are stored in memory and should be chosen according to the available amount of memory and the size of a single image.

ecmwf_models.erainterim.reshuffle.**run**()

Module contents

Submodules

ecmwf_models.grid module

Common grid definitions for ECMWF model reanalysis products (regular gridded)

ecmwf_models.grid.**ERA5_RegularImgLandGrid**(*res_lat: float = 0.25*, *res_lon: float = 0.25*, *bbox: Optional[Tuple[float, float, float, float]] = None*) → pygeogrids.grids.CellGrid

Uses the 0.25 DEG ERA5 land mask to create a land grid of the same size, which also excluded Antarctica.

Parameters

- **res_lat** (*float, optional (default: 0.25)*) – Grid resolution (in degrees) in latitude direction.
- **res_lon** (*float, optional (default: 0.25)*) – Grid resolution (in degrees) in longitude direction.
- **bbox** (*tuple, optional (default: None)*) – (min_lon, min_lat, max_lon, max_lat) - wgs84 bbox to cut the global grid to.

ecmwf_models.grid.**ERA_IrregularImgGrid**(*lons: numpy.ndarray*, *lats: numpy.ndarray*, *bbox: Optional[Tuple[float, float, float, float]] = None*) → pygeogrids.grids.CellGrid

Create a irregular grid from the passed coordinates.

ecmwf_models.grid.**ERA_RegularImgGrid**(*res_lat: float = 0.25*, *res_lon: float = 0.25*, *bbox: Optional[Tuple[float, float, float, float]] = None*) → pygeogrids.grids.CellGrid

Create regular cell grid for bounding box with the selected resolution.

Parameters

- **res_lat** (*float, optional (default: 0.25)*) – Grid resolution (in degrees) in latitude direction.
- **res_lon** (*float, optional (default: 0.25)*) – Grid resolution (in degrees) in longitude direction.
- **bbox** (*tuple, optional (default: None)*) – (min_lon, min_lat, max_lon, max_lat) - wgs84 bbox to cut the global grid to.

Returns **CellGrid** – Regular, CellGrid with 5DEG*5DEG cells for the passed bounding box.

Return type CellGrid

ecmwf_models.grid.**get_grid_resolution**(*lats: numpy.ndarray*, *lons: numpy.ndarray*) -> (<class 'float'>, <class 'float'>)

try to derive the grid resolution from given coords.

ecmwf_models.interface module

class ecmwf_models.interface.**ERAGrbDs**(*root_path*, *product*, *parameter=('swvl1', 'swvl2')*, *subgrid=None*, *mask_seapoints=False*, *h_steps=(0, 6, 12, 18)*, *array_1D=True*)

Bases: pygeobase.io_base.MultiTemporalImageBase

Reader for a stack of ERA grib files.

Parameters

- **root_path** (*string*) – Root path where the data is stored
- **product** (*str*) – ERA5 or ERAINT
- **parameter** (*list or str, optional (default: ['swvl1', 'swvl2'])*) – Parameter or list of parameters to read
- **expand_grid** (*bool, optional (default: True)*) – If the reduced gaussian grid should be expanded to a full gaussian grid.

tstamps_for_daterange(*start_date*, *end_date*)

Get datetimes in the correct sub-daily resolution between 2 dates

Parameters

- **start_date** (*datetime*) – Start datetime
- **end_date** (*datetime*) – End datetime

Returns **timestamps** – List of datetimes

Return type list

class ecmwf_models.interface.**ERAGrbImg**(*filename*, *product*, *parameter=('swvl1', 'swvl2')*, *subgrid=None*, *mask_seapoints=False*, *array_1D=True*, *mode='r'*)

Bases: pygeobase.io_base.ImageBase

Base class for reader for a single ERA Grib file.

Parameters

- **filename** (*str*) – Path to the image file to read.
- **product** (*str*) – ERA5 or ERAINT
- **parameter** (*list or str, optional (default: ['swvl1', 'swvl2'])*) – Name of parameters to read from the image file.
- **subgrid** (*pygeogrids.CellGrid, optional (default:None)*) – Read only data for points of this grid and not global values.
- **mask_seapoints** (*bool, optional (default: False)*) – Read the land-sea mask to mask points over water and set them to nan. This option needs the 'lsm' parameter to be in the file!
- **array_1D** (*bool, optional (default: False)*) – Read data as list, instead of 2D array, used for reshuffling.
- **mode** (*str, optional (default: 'r')*) – Mode in which to open the file, changing this can cause data loss. This argument should not be changed!

close()

Close file.

flush()

Flush data.

read(*timestamp=None*)
Read data from the loaded image file.

Parameters **timestamp** (*datetime, optional (default: None)*) – Specific date (time) to read the data for.

write(*data*)
Write data to an image file.

Parameters **image** (*object*) – pygeobase.object_base.Image object

class ecmwf_models.interface.**ERANcDs**(*root_path*, *product*, *parameter=('swvl1', 'swvl2')*, *subgrid=None*, *mask_seapoints=False*, *h_steps=(0, 6, 12, 18)*, *array_1D=False*)
Bases: pygeobase.io_base.MultiTemporalImageBase

Class for reading ERA 5 images in nc format.

Parameters

- **root_path** (*str*) – Root path where image data is stored.
- **parameter** (*list or str, optional (default: ['swvl1', 'swvl2'])*) – Parameter or list of parameters to read from image files.
- **subgrid** (*pygeogrids.CellGrid, optional (default: None)*) – Read only data for points of this grid and not global values.
- **mask_seapoints** (*bool, optional (default: False)*) – Use the land-sea-mask parameter in the file to mask points over water.
- **h_steps** (*list, optional (default: [0,6,12,18])*) – List of full hours for which images exist.
- **array_1D** (*bool, optional (default: False)*) – Read data as list, instead of 2D array, used for reshuffling.

tstamps_for_daterange(*start_date*, *end_date*)
Get datetimes in the correct sub-daily resolution between 2 dates

Parameters

- **start_date** (*datetime*) – Start datetime
- **end_date** (*datetime*) – End datetime

Returns **timestamps** – List of datetimes

Return type list

class ecmwf_models.interface.**ERANcImg**(*filename*, *product*, *parameter=['swvl1', 'swvl2']*, *subgrid=None*, *mask_seapoints=False*, *array_1D=False*, *mode='r'*)
Bases: pygeobase.io_base.ImageBase

Reader for a single ERA netcdf file.

Parameters

- **filename** (*str*) – Path to the image file to read.
- **product** (*str*) – 'era5' or 'era5-land' or 'eraint'
- **parameter** (*list or str, optional (default: ['swvl1', 'swvl2'])*) – Name of parameters to read from the image file.
- **subgrid** (*pygeogrids.CellGrid, optional (default: None)*) – Read only data for points of this grid and not global values.

- **mask_seapoints** (*bool, optional (default: False)*) – Read the land-sea mask to mask points over water and set them to nan. This option needs the 'lsm' parameter to be in the file!
- **array_1D** (*bool, optional (default: False)*) – Read data as list, instead of 2D array, used for reshuffling.
- **mode** (*str, optional (default: 'r')*) – Mode in which to open the file, changing this can cause data loss. This argument should not be changed!

close()
Close file.

flush()
Flush data.

read(*timestamp=None*)
Read data from the loaded image file.

Parameters **timestamp** (*datetime, optional (default: None)*) – Specific date (time) to read the data for.

write(*data*)
Write data to an image file.

Parameters **image** (*object*) – pygeobase.object_base.Image object

class ecmwf_models.interface.**ERATs**(*ts_path*, *grid_path=None*, ***kwargs*)
Bases: pynetcf.time_series.GriddedNcOrthoMultiTs

Time series reader for all reshuffled ERA reanalysis products in time series format. Use the read_ts(lon, lat) resp. read_ts(gpi) function of this class to read data for locations!

Parameters

- **ts_path** (*str*) – Directory where the netcdf time series files are stored
- **grid_path** (*str, optional (default: None)*) – Path to grid file, that is used to organize the location of time series to read. If None is passed, grid.nc is searched for in the ts_path.
- **used** (*Optional keyword arguments that are passed to the Gridded Base when*) –
- ---------- –

  parameters [list, optional (default: None)] Specific variable names to read, if None are selected, all are read.

  offsets [dict, optional (default: None)] Offsets (values) that are added to the parameters (keys)

  scale_factors [dict, optional (default: None)] Offset (value) that the parameters (key) is multiplied with

  ioclass_kws: dict, (optional)

  read_bulk [boolean, optional (default: False)] If set to True, the data of all locations is read into memory, and subsequent calls to read_ts then read from cache and not from disk. This makes reading complete files faster.

read_dates [boolean, optional (default: False)] If false, dates will not be read automatically but only on specific request useable for bulk reading because currently the netCDF num2date routine is very slow for big datasets.

ecmwf_models.utils module

Utility functions for all data products in this package.

exception ecmwf_models.utils.**CdoNotFoundError**(*msg=None*)
Bases: ModuleNotFoundError

ecmwf_models.utils.**get_default_params**(*name='era5'*)
Read only lines that are marked as default variable in the csv file

Parameters **name** (*str*) – Name of the product to get the default parameters for

ecmwf_models.utils.**load_var_table**(*name='era5'*, *lut=False*)
Load the variables table for supported variables to download.

Parameters **lut** (*bool, optional (default: False)*) – If set to true only names are loaded, so that they can be used for a LUT otherwise the full table is loaded

ecmwf_models.utils.**lookup**(*name*, *variables*)
Search the passed elements in the lookup table, if one does not exists, raise a Warning

ecmwf_models.utils.**make_era5_land_definition_file**(*data_file*, *out_file*, *data_file_y_res=0.25*, *ref_var='lsm'*, *threshold=0.5*, *exclude_antarctica=True*)
Create a land grid definition file from a variable within a downloaded, regular (netcdf) era5 file.

Parameters

- **data_file** (*str*) – Path to the downloaded file that cotains the image that is used as the reference for creating the land definition file.
- **out_file** (*str*) – Full output path to the land defintion file to create.
- **data_file_y_res** (*float, optional (default: 0.25)*) – The resolution of the data file in latitude direction.
- **ref_var** (*str, optional (default: 'lsm')*) – A variable in the data_file that is the reference for the land definiton. By default, we use the land-sea-mask variable.
- **threshold** (*float, optional (default: 0.5)*) – Threshold value below which a point is declared water, and above (or equal) which it is declared a land-point. If None is passed, then a point is declared a land point if it is not masked (numpy masked array) in the reference variable.
- **exclude_antarctica** (*bool, optional (default: True)*) – Cut off the definition file at -60° Lat to exclude Land Points in Antarctica.

ecmwf_models.utils.**mkdate**(*datestring*)
Turns a datetime string into a datetime object

Parameters **datestring** (*str*) – Input datetime string

Returns **datetime** – Converted string

Return type datetime

ecmwf_models.utils.**parse_filetype**(*inpath*)
Tries to find out the file type by searching for grib or nc files two subdirectories into the passed input path. If function fails, grib is assumed.

Parameters **inpath** (*str*) – Input path where ERA data was downloaded to

Returns **filetype** – File type string.

Return type str

ecmwf_models.utils.**parse_product**(*inpath: str*) → str

Tries to find out what product is stored in the path. This is done based on the name of the first file in the path that is found.

Parameters **inpath** (*str*) – Input path where ERA data was downloaded to

Returns **product** – Product name

Return type str

ecmwf_models.utils.**save_gribs_from_grib**(*input_grib*, *output_path*, *product_name*, *filename_templ='{product}_AN_%Y%m%d_%H%M.grb'*, *keep_original=True*)

Split the downloaded grib file into daily files and add to folder structure necessary for reshuffling.

Parameters

- **input_grib** (*str*) – Filepath of the downloaded .grb file
- **output_path** (*str*) – Where to save the resulting grib files
- **product_name** (*str*) – Name of the ECMWF model (only for filename generation)
- **filename_templ** (*str, optional (default: product_OPER_0001_AN_date_time)*) – Template for naming each separated grb file

ecmwf_models.utils.**save_ncs_from_nc**(*input_nc*, *output_path*, *product_name*, *filename_templ='{product}_AN_%Y%m%d_%H%M.nc'*, *grid=None*, *keep_original=True*, *remap_method='bil'*)

Split the downloaded netcdf file into daily files and add to folder structure necessary for reshuffling.

Parameters

- **input_nc** (*str*) – Filepath of the downloaded .nc file
- **output_path** (*str*) – Where to save the resulting netcdf files
- **product_name** (*str*) – Name of the ECMWF model (only for filename generation)
- **filename_templ** (*str, optional (default: product_grid_date_time)*) – Template for naming each separated nc file
- **keep_original** (*bool*) – keep the original downloaded data

ecmwf_models.utils.**str2bool**(*v*)

Parse a string to True/False

Parameters **v** (*str*) – String to parse, must be part of the lists below.

Returns **str2bool** – The parsed bool from the passed string

Return type bool

Module contents

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

ecmwf_models, 38
ecmwf_models.era5, 29
ecmwf_models.era5.download, 25
ecmwf_models.era5.interface, 27
ecmwf_models.era5.reshuffle, 28
ecmwf_models.erainterim, 32
ecmwf_models.erainterim.download, 29
ecmwf_models.erainterim.interface, 30
ecmwf_models.erainterim.reshuffle, 31
ecmwf_models.grid, 32
ecmwf_models.interface, 33
ecmwf_models.utils, 36

INDEX

C

CdoNotFoundError, 36
CDSStatusTracker (*class in ecmwf_models.era5.download*), 25
close() (*ecmwf_models.interface.ERAGrbImg method*), 33
close() (*ecmwf_models.interface.ERANcImg method*), 35

D

default_variables() (*in module ecmwf_models.era5.download*), 25
default_variables() (*in module ecmwf_models.erainterim.download*), 29
download_and_move() (*in module ecmwf_models.era5.download*), 25
download_and_move() (*in module ecmwf_models.erainterim.download*), 29
download_era5() (*in module ecmwf_models.era5.download*), 26
download_eraint() (*in module ecmwf_models.erainterim.download*), 29

E

ecmwf_models
    module, 38
ecmwf_models.era5
    module, 29
ecmwf_models.era5.download
    module, 25
ecmwf_models.era5.interface
    module, 27
ecmwf_models.era5.reshuffle
    module, 28
ecmwf_models.erainterim
    module, 32
ecmwf_models.erainterim.download
    module, 29
ecmwf_models.erainterim.interface
    module, 30
ecmwf_models.erainterim.reshuffle
    module, 31
ecmwf_models.grid
    module, 32
ecmwf_models.interface
    module, 33
ecmwf_models.utils
    module, 36
ERA5_RegularImgLandGrid() (*in module ecmwf_models.grid*), 32
ERA5GrbDs (*class in ecmwf_models.era5.interface*), 27
ERA5GrbImg (*class in ecmwf_models.era5.interface*), 27
ERA5NcDs (*class in ecmwf_models.era5.interface*), 27
ERA5NcImg (*class in ecmwf_models.era5.interface*), 27
ERA_IrregularImgGrid() (*in module ecmwf_models.grid*), 32
ERA_RegularImgGrid() (*in module ecmwf_models.grid*), 32
ERAGrbDs (*class in ecmwf_models.interface*), 33
ERAGrbImg (*class in ecmwf_models.interface*), 33
ERAIntGrbDs (*class in ecmwf_models.erainterim.interface*), 30
ERAIntGrbImg (*class in ecmwf_models.erainterim.interface*), 30
ERAIntNcDs (*class in ecmwf_models.erainterim.interface*), 30
ERAIntNcImg (*class in ecmwf_models.erainterim.interface*), 30
ERANcDs (*class in ecmwf_models.interface*), 34
ERANcImg (*class in ecmwf_models.interface*), 34
ERATs (*class in ecmwf_models.interface*), 35

F

flush() (*ecmwf_models.interface.ERAGrbImg method*), 33
flush() (*ecmwf_models.interface.ERANcImg method*), 35

G

get_default_params() (*in module ecmwf_models.utils*), 36
get_grid_resolution() (*in module ecmwf_models.grid*), 32

H

handle_error_function() (*ecmwf_models.era5.download.CDSStatusTracker method*), 25

L

load_var_table() (*in module ecmwf_models.utils*), 36
lookup() (*in module ecmwf_models.utils*), 36

M

main() (*in module ecmwf_models.era5.download*), 26
main() (*in module ecmwf_models.era5.reshuffle*), 28
main() (*in module ecmwf_models.erainterim.download*), 30
main() (*in module ecmwf_models.erainterim.reshuffle*), 31
make_era5_land_definition_file() (*in module ecmwf_models.utils*), 36
mkdate() (*in module ecmwf_models.utils*), 36
module
    ecmwf_models, 38
    ecmwf_models.era5, 29
    ecmwf_models.era5.download, 25
    ecmwf_models.era5.interface, 27
    ecmwf_models.era5.reshuffle, 28
    ecmwf_models.erainterim, 32
    ecmwf_models.erainterim.download, 29
    ecmwf_models.erainterim.interface, 30
    ecmwf_models.erainterim.reshuffle, 31
    ecmwf_models.grid, 32
    ecmwf_models.interface, 33
    ecmwf_models.utils, 36

P

parse_args() (*in module ecmwf_models.era5.download*), 26
parse_args() (*in module ecmwf_models.era5.reshuffle*), 28
parse_args() (*in module ecmwf_models.erainterim.download*), 30
parse_args() (*in module ecmwf_models.erainterim.reshuffle*), 31
parse_filetype() (*in module ecmwf_models.utils*), 36
parse_product() (*in module ecmwf_models.utils*), 37

R

read() (*ecmwf_models.interface.ERAGrbImg method*), 33
read() (*ecmwf_models.interface.ERANcImg method*), 35
reshuffle() (*in module ecmwf_models.era5.reshuffle*), 28
reshuffle() (*in module ecmwf_models.erainterim.reshuffle*), 31
run() (*in module ecmwf_models.era5.download*), 27
run() (*in module ecmwf_models.era5.reshuffle*), 28
run() (*in module ecmwf_models.erainterim.download*), 30
run() (*in module ecmwf_models.erainterim.reshuffle*), 31

S

save_gribs_from_grib() (*in module ecmwf_models.utils*), 37
save_ncs_from_nc() (*in module ecmwf_models.utils*), 37
statuscode_error (*ecmwf_models.era5.download.CDSStatusTracker attribute*), 25
statuscode_ok (*ecmwf_models.era5.download.CDSStatusTracker attribute*), 25
statuscode_unavailable (*ecmwf_models.era5.download.CDSStatusTracker attribute*), 25
str2bool() (*in module ecmwf_models.utils*), 37

T

tstamps_for_daterange() (*ecmwf_models.interface.ERAGrbDs method*), 33
tstamps_for_daterange() (*ecmwf_models.interface.ERANcDs method*), 34

W

write() (*ecmwf_models.interface.ERAGrbImg method*), 34
write() (*ecmwf_models.interface.ERANcImg method*), 35