ecmwf$_{m}odelsDocumentation$

TU Wien

Jun 18, 2020

Contents

1 Citation **3**

2 Installation **5**

3 Supported Products **7**

4 Contribute **9**
- 4.1 Development setup . . . 9
- 4.2 Guidelines . . . 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 . . . 20
- 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 . . . 23
- 9.8 Changelog . . . 23
- 9.9 ecmwf_models . . . 24

10 Indices and tables **27**

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

Works great in combination with pytesmo.

CHAPTER 1

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 2

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 scipy 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 environment.yml file in this repository can be used.

```
git clone git@github.com:TUW-GEO/ecmwf_models.git ecmwf_models
cd ecmwf_models
conda create -n ecmwf-models python=3.6 # or any other supported version
source activate ecmwf-models
conda env update -f environment.yml
python setup.py develop
```

This script should work on Linux or OSX and uses the `environment.yml` file included in this repository.

CHAPTER 3

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 4

Contribute

We are happy if you want to contribute. Please raise an issue explaining what is missing or if you find a bug. We will also gladly accept pull requests against our master branch for new features or bug fixes.

4.1 Development setup

For Development we also recommend the `conda` environment from the installation part.

4.2 Guidelines

If you want to contribute please follow these steps:

- Fork the ecmwf_models repository to your account
- make a new feature branch from the ecmwf_models master branch
- Add your feature
- please include tests for your contributions in one of the test directories We use py.test so a simple function called test_my_feature is enough
- submit a pull request to our master branch

CHAPTER 5

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 6

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 resolution 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 7

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))
```

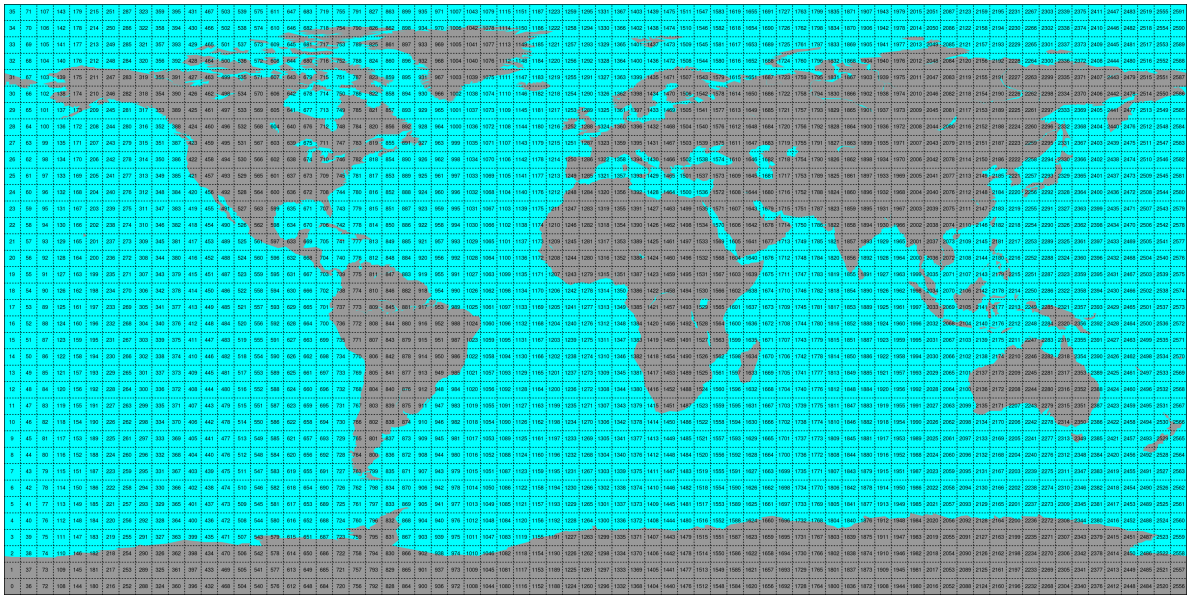
All images between two given dates can be read using the `iter_images` methods of all the image stack reader classes.

CHAPTER 8

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 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`.

```
conda install -c conda-forge libnetcdf==4.3.3.1 --yes
# if this does not work, consider downgrading the netcdf4 library and its␣
↪dependencies:
conda install -c conda-forge netcdf4==1.2.2 --yes
```

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
ds = ERATs(ts_path, ioclass_kws={'read_bulk':True}) # 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.
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 9

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 resolution 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
```

(continues on next page)

(continued from previous page)

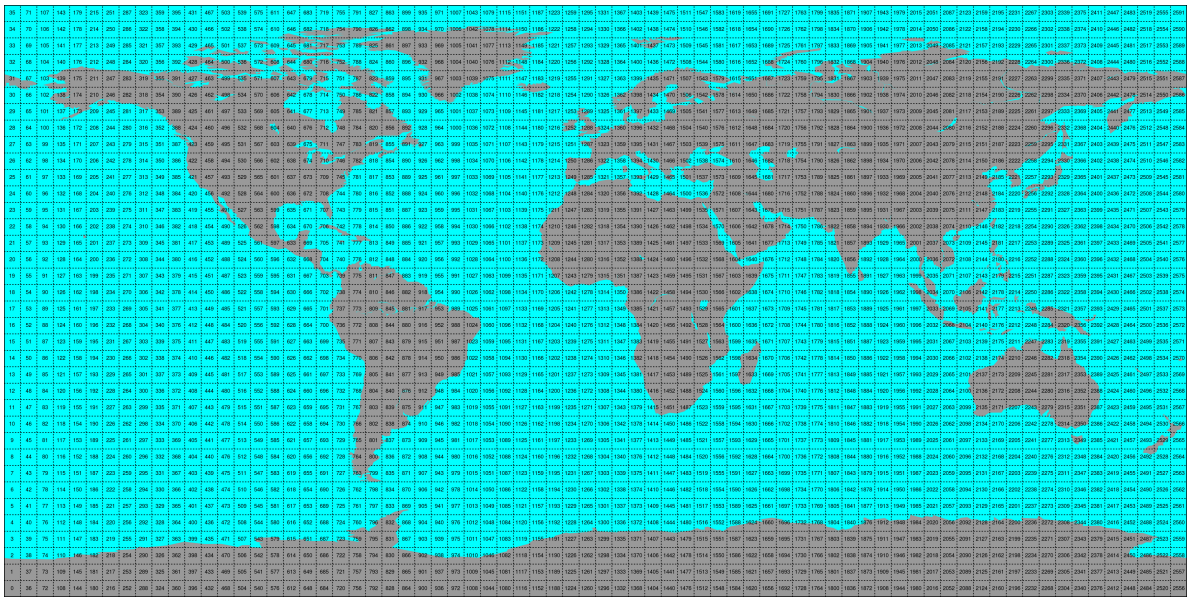
```
root_path = "/path/to/storage"
ds = ERA5NcDs(root_path, parameter=['swvl1', 'swvl2'])
data = ds.read(datetime(2000, 1, 1, 0))
```

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 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`.

```
conda install -c conda-forge libnetcdf==4.3.3.1 --yes
# if this does not work, consider downgrading the netcdf4 library and its␣
↪dependencies:
conda install -c conda-forge netcdf4==1.2.2 --yes
```

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
ds = ERATs(ts_path, ioclass_kws={'read_bulk':True}) # 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.
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

```
The MIT License (MIT)

Copyright (c) 2016 TU Wien

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

(continues on next page)

(continued from previous page)

```
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

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 0.6.1

- 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

9.8.3 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

9.8.4 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

9.8.5 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

9.8.6 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.

9.8.7 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.

9.8.8 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

ecmwf_models.era5.interface module

ecmwf_models.era5.reshuffle module

Module contents

ecmwf_models.erainterim package

Submodules

ecmwf_models.erainterim.download module

ecmwf_models.erainterim.interface module

ecmwf_models.erainterim.reshuffle module

Module contents

Submodules

ecmwf_models.grid module

ecmwf_models.interface module

ecmwf_models.utils module

Module contents

CHAPTER 10

Indices and tables

- genindex
- modindex
- search