gldas Documentation

Release unknown

TU Wien

Jan 07, 2021

Contents

Readers and converters for data from the GLDAS Noah Land Surface Model. 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.596427 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

Setup of a complete environment with conda can be performed using the following commands:

```
conda create -n gldas python=2.7 # or any other supported python version
source activate gldas
```

```
# Either install required conda packages manually
conda install -c conda-forge numpy netCDF4 pyproj pygrib
# Or use the provided environment file to install all dependencies
conda env update -f environment.yml
```

```
# Install the gldas package and pip-dependencies
pip install gldas
```

This will also try to install pygrib for reading the GLDAS grib files. If this does not work then please consult the pygrib manual.

CHAPTER 3

Supported Products

At the moment this package supports GLDAS Noah data version 1 in grib format (reading, time series creation) and GLDAS Noah data version 2.0 and version 2.1 in netCDF format (download, reading, time series creation) with a spatial sampling of 0.25 degrees. It should be easy to extend the package to support other GLDAS based products. This will be done as need arises.

CHAPTER 4

Downloading Products

In order to download GLDAS NOAH products you have to register an account with NASA's Earthdata portal. Instructions can be found here.

After that you can use the command line program `gldas_download`.

```
mkdir ~/workspace/gldas_data
gldas_download ~/workspace/gldas_data
```

would download GLDAS Noah version 2.0 in 0.25 degree sampling into the folder `~/workspace/gldas_data`. For more options run `gldas_download -h`.

CHAPTER 5

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.

5.1 Development setup

For Development we also recommend a `conda` environment. You can create one including test dependencies and debugger by running `conda env create -f environment.yml`. This will create a new `gldas` environment which you can activate by using `source activate gldas`.

5.2 Guidelines

If you want to contribute please follow these steps:

- Fork the gldas repository to your account
- Clone the repository, make sure you use `git clone --recursive` to also get the test data repository.
- make a new feature branch from the gldas 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 6

Note

This project has been set up using PyScaffold 2.5.6. For details and usage information on PyScaffold see http://pyscaffold.readthedocs.org/.

6.1 Reading GLDAS images

Reading of the GLDAS raw grib files can be done in two ways.

6.1.1 Reading by file name

```
import os
from datetime import datetime
from gldas.interface import GLDAS_Noah_v1_025Img

# read several parameters
parameter = ['086_L2', '086_L1', '085_L1', '138', '132', '051']
# the class is initialized with the exact filename.
img = GLDAS_Noah_v1_025Img(os.path.join(os.path.dirname(__file__),
                                        'test-data',
                                        'GLDAS_NOAH_image_data',
                                        '2015',
                                        '001',
                                        'GLDAS_NOAH025SUBP_3H.A2015001.0000.001.
↪2015037193230.grb'),
                           parameter=parameter)

# reading returns an image object which contains a data dictionary
# with one array per parameter. The returned data is a global 0.25 degree
# image/array.
image = img.read()
```

(continues on next page)

(continued from previous page)

```
assert image.data['086_L1'].shape == (720, 1440)
assert image.lon[0, 0] == -179.875
assert image.lon[0, 1439] == 179.875
assert image.lat[0, 0] == 89.875
assert image.lat[719, 0] == -89.875
assert sorted(image.data.keys()) == sorted(parameter)
assert image.data['086_L1'][26, 609] == 30.7344
assert image.data['086_L2'][26, 609] == 93.138
assert image.data['085_L1'][576, 440] == 285.19
assert image.data['138'][26, 609] == 237.27
assert image.data['051'][26, 609] == 0
assert image.lon.shape == (720, 1440)
assert image.lon.shape == image.lat.shape
```

6.1.2 Reading by date

All the gldas data in a directory structure can be accessed by date. The filename is automatically built from the given date.

```
from gldas.interface import GLDAS_Noah_v1_025Ds

parameter = ['086_L2', '086_L1', '085_L1', '138', '132', '051']
img = GLDAS_Noah_v1_025Ds(data_path=os.path.join(os.path.dirname(__file__),
                                                 'test-data',
                                                 'GLDAS_NOAH_image_data'),
                          parameter=parameter)

image = img.read(datetime(2015, 1, 1, 0))
```

For reading all image between two dates the `gldas.interface.GLDAS_Noah_v1_025Ds.iter_images()` iterator can be used.

CHAPTER 7

Variable naming for different versions of GLDAS NOAH

For GLDAS Noah 1.0 parameters are called using their PDS IDs from the table below. A full list of PDS IDs can be found in the GLDAS 1.0 README

For GLDAS Noah 2.0 and GLDAS Noah 2.1 parameters are called using Variable Names from the table below. A full list of variable names can be found in the GLDAS 2.x README

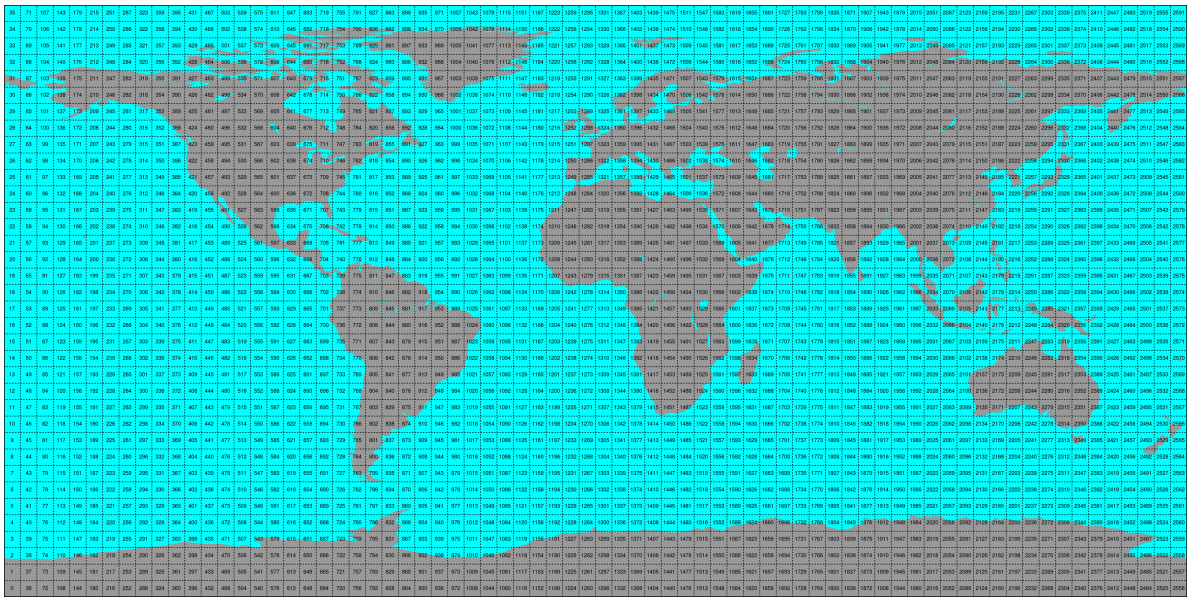
PDS ID (old)	Variable Name (new)	Parameter	Resolution	Depth/Height Interval [m]	Units
086_L1	SoilMoi0_10cm_inst	Soil moisture	0.25°	0.00 - 0.10	[kg/m^2]
086_L2	SoilMoi10_40cm_inst	Soil moisture	0.25°	0.10 - 0.40	[kg/m^2]
086_L3	SoilMoi40_100cm_inst	Soil moisture	0.25°	0.40 - 1.00	[kg/m^2]
086_L4	SoilMoi100_200cm_inst	Soil moisture	0.25°	1.00 - 2.00	[kg/m^2]
085_L1	SoilTMP0_10cm_inst	Soil temperature	0.25°	0.00 - 0.10	[K]
085_L2	SoilTMP10_40cm_inst	Soil temperature	0.25°	0.10 - 0.40	[K]
085_L3	SoilTMP40_100cm_inst	Soil temperature	0.25°	0.40 - 1.00	[K]
085_L4	SoilTMP100_200cm_inst	Soil temperature	0.25°	1.00 - 2.00	[K]
065	SWE_inst	Snow depth water equivalent	0.25°	0	[kg/m^2]
138	AvgSurfT_inst	Average Surface Skin temperature	0.25°	0	[K]
131	Snowf_tavg	Snow precipitation rate	0.25°	0	[kg/m^2/s]
132	Rainf_tavg	Rain precipitation rate	0.25°	0	[kg/m^2/s]
057	Evap_tavg	Total Evapo-transpiration	0.25°	0	[kg/m^2/s]

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 since that saves space.
- Further reduction the amount of stored data by saving only land points if selected.
- 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 (without reduction to land points) 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 `gldas_repurpose` command line program. An example would be:

```
gldas_repurpose /gldas_data /timeseries/data 2000-01-01 2001-01-01 SoilMoi0_10cm_inst␣
↪SoilMoi10_40cm_inst
```

Which would take GLDAS Noah data stored in `/gldas_data` from January 1st 2000 to January 1st 2001 and store the parameters for the top 2 layers of soil moisture as time series in the folder `/timeseries/data`.

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 `gldas.reshuffle`.

Note: If a `RuntimeError: NetCDF: Bad chunk sizes.` appears during reshuffling, consider downgrading the netcdf4 library via:

```
conda install -c conda-forge netcdf4=1.2.2
```

8.1 Reading converted time series data

For reading the data the `gldas_repurpose` command produces the class `GLDASTs` can be used:

```
from gldas.interface import GLDASTs
ds = GLDASTs(ts_path)
# read_ts takes either lon, lat coordinates or a grid point indices.
# and returns a pandas.DataFrame
ts = ds.read_ts(45, 15)
```

CHAPTER 9

Contents

9.1 Reading GLDAS images

Reading of the GLDAS raw grib files can be done in two ways.

9.1.1 Reading by file name

```
import os
from datetime import datetime
from gldas.interface import GLDAS_Noah_v1_025Img

# read several parameters
parameter = ['086_L2', '086_L1', '085_L1', '138', '132', '051']
# the class is initialized with the exact filename.
img = GLDAS_Noah_v1_025Img(os.path.join(os.path.dirname(__file__),
                                        'test-data',
                                        'GLDAS_NOAH_image_data',
                                        '2015',
                                        '001',
                                        'GLDAS_NOAH025SUBP_3H.A2015001.0000.001.
↪2015037193230.grb'),
                           parameter=parameter)

# reading returns an image object which contains a data dictionary
# with one array per parameter. The returned data is a global 0.25 degree
# image/array.
image = img.read()

assert image.data['086_L1'].shape == (720, 1440)
assert image.lon[0, 0] == -179.875
assert image.lon[0, 1439] == 179.875
assert image.lat[0, 0] == 89.875
```

(continues on next page)

(continued from previous page)

```
assert image.lat[719, 0] == -89.875
assert sorted(image.data.keys()) == sorted(parameter)
assert image.data['086_L1'][26, 609] == 30.7344
assert image.data['086_L2'][26, 609] == 93.138
assert image.data['085_L1'][576, 440] == 285.19
assert image.data['138'][26, 609] == 237.27
assert image.data['051'][26, 609] == 0
assert image.lon.shape == (720, 1440)
assert image.lon.shape == image.lat.shape
```

9.1.2 Reading by date

All the gldas data in a directory structure can be accessed by date. The filename is automatically built from the given date.

```
from gldas.interface import GLDAS_Noah_v1_025Ds

parameter = ['086_L2', '086_L1', '085_L1', '138', '132', '051']
img = GLDAS_Noah_v1_025Ds(data_path=os.path.join(os.path.dirname(__file__),
                                                 'test-data',
                                                 'GLDAS_NOAH_image_data'),
                          parameter=parameter)

image = img.read(datetime(2015, 1, 1, 0))
```

For reading all image between two dates the `gldas.interface.GLDAS_Noah_v1_025Ds.iter_images()` iterator can be used.

9.2 Variable naming for different versions of GLDAS NOAH

For GLDAS Noah 1.0 parameters are called using their PDS IDs from the table below. A full list of PDS IDs can be found in the GLDAS 1.0 README

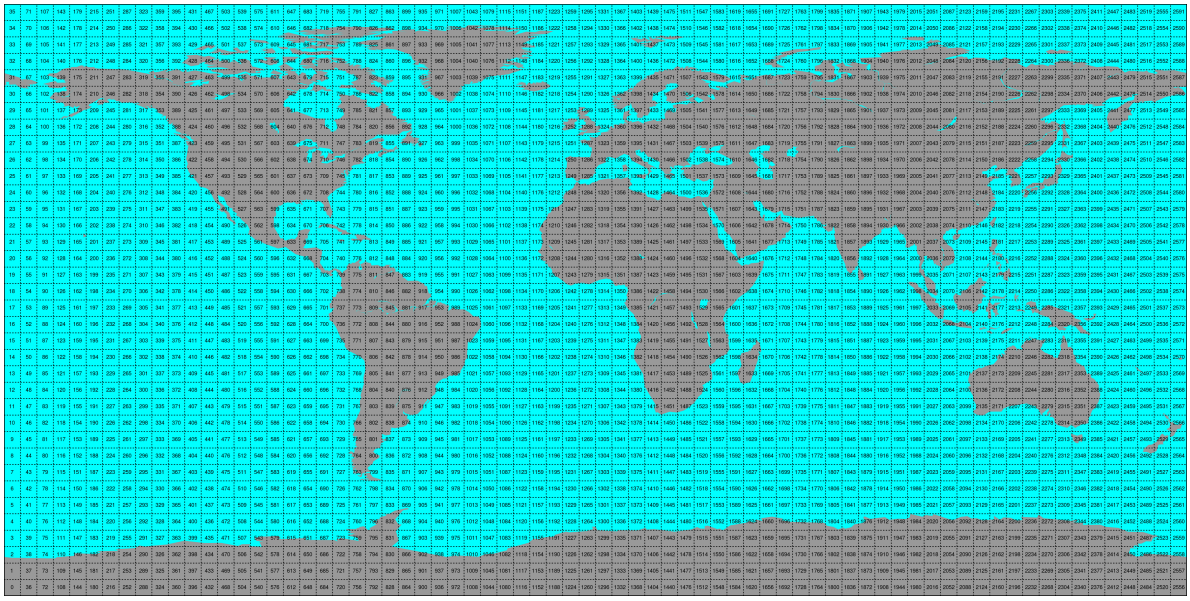
For GLDAS Noah 2.0 and GLDAS Noah 2.1 parameters are called using Variable Names from the table below. A full list of variable names can be found in the GLDAS 2.x README

PDS ID (old)	Variable Name (new)	Parameter	Resolution	Depth/Height Interval [m]	Units
086_L1	SoilMoi0_10cm_inst	Soil moisture	0.25°	0.00 - 0.10	[kg/m^2]
086_L2	SoilMoi10_40cm_inst	Soil moisture	0.25°	0.10 - 0.40	[kg/m^2]
086_L3	SoilMoi40_100cm_inst	Soil moisture	0.25°	0.40 - 1.00	[kg/m^2]
086_L4	SoilMoi100_200cm_inst	Soil moisture	0.25°	1.00 - 2.00	[kg/m^2]
085_L1	SoilTMP0_10cm_inst	Soil temperature	0.25°	0.00 - 0.10	[K]
085_L2	SoilTMP10_40cm_inst	Soil temperature	0.25°	0.10 - 0.40	[K]
085_L3	SoilTMP40_100cm_inst	Soil temperature	0.25°	0.40 - 1.00	[K]
085_L4	SoilTMP100_200cm_inst	Soil temperature	0.25°	1.00 - 2.00	[K]
065	SWE_inst	Snow depth water equivalent	0.25°	0	[kg/m^2]
138	AvgSurfT_inst	Average Surface Skin temperature	0.25°	0	[K]
131	Snowf_tavg	Snow precipitation rate	0.25°	0	[kg/m^2/s]
132	Rainf_tavg	Rain precipitation rate	0.25°	0	[kg/m^2/s]
057	Evap_tavg	Total Evapo-transpiration	0.25°	0	[kg/m^2/s]

9.3 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 since that saves space.
- Further reduction the amount of stored data by saving only land points if selected.
- 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 (without reduction to land points) 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 `gldas_repurpose` command line program. An example would be:

```
gldas_repurpose /gldas_data /timeseries/data 2000-01-01 2001-01-01 SoilMoi0_10cm_inst␣
↪SoilMoi10_40cm_inst
```

Which would take GLDAS Noah data stored in `/gldas_data` from January 1st 2000 to January 1st 2001 and store the parameters for the top 2 layers of soil moisture as time series in the folder `/timeseries/data`.

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 `gldas.reshuffle`.

Note: If a `RuntimeError: NetCDF: Bad chunk sizes.` appears during reshuffling, consider downgrading the netcdf4 library via:

```
conda install -c conda-forge netcdf4=1.2.2
```

9.3.1 Reading converted time series data

For reading the data the `gldas_repurpose` command produces the class `GLDASTs` can be used:

```
from gldas.interface import GLDASTs
ds = GLDASTs(ts_path)
# read_ts takes either lon, lat coordinates or a grid point indices.
# and returns a pandas.DataFrame
ts = ds.read_ts(45, 15)
```

9.4 License

```
Copyright (c) 2016, TU Wien
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
```

(continues on next page)

(continued from previous page)

```
* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of ascat nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

9.5 Developers

- Wolfgang Preimesberger <wolfgang.preimesberger@geo.tuwien.ac.at>
- Christoph Paulik <cpaulik@vandersat.com>
- Andreea Plocon

9.6 Changelog

9.6.1 Unreleased

-

9.6.2 Version 0.6

- New package structure
- Drop python 2 support
- Allow subsetting by bbox
- Add gldas land mask to package (no download necessary)

9.6.3 Version 0.5

- Update trollsift parsing
- Update readme

- Support reshuffling of land points only
- Test with netcdf test data

9.6.4 Version 0.4

- Add support for GLDAS version 2.1
- Compress files during reshuffling.

9.6.5 Version 0.3

- Fix download with new URL
- Only download grb and xml files

9.6.6 Version 0.2

- Fix Python 3 bug when iterating over multiple images.
- Add reshuffling to time series format and reading of time series.

9.6.7 Version 0.1

- First release. Support for Downloading and reading GLDAS Noah v1 0.25 degree data.

9.7 gldas

9.7.1 gldas package

Submodules

gldas.download module

gldas.grid module

gldas.interface module

gldas.reshuffle module

Module contents

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

G