basinex Documentation

Release 0.2.0

mHM Developers

Jul 15, 2022

CONTENTS

CHAPTER 1

BASINEX

The mHM basin extractor. Extract basins for given gauging stations.

1.1 Dependencies

- numpy v1.14.5 or later
- netCDF4
- GDAL
- pyyaml
- C++ compiler (for development version)

1.2 Installation

If you have GDAL already installed, basinex can be installed via pip:

```
pip install basinex
```

GDAL installation

Getting GDAL installed with pip is allways a bit cumbersome. Therefore we compiled instructions for the main target systems.

Ubuntu

To get a recent version of GDAL, you can use the ppa of ubuntugis:

```
sudo add-apt-repository ppa:ubuntugis/ppa
sudo apt-get update
sudo apt install gdal-bin libgdal-dev
pip install wheel numpy
pip install GDAL==$(gdal-config --version)
```

MacOS

GDAL can be installed with homebrew:

```
brew install gdal
pip install wheel numpy
pip install GDAL==$(gdal-config --version)
```

Windows

You can use the unoffical wheels of Christoph Gohlke to install GDAL. The easiest way to do so, is using pipwin:

```
pip install pipwin
pipwin install gdal
```

Development version in conda environment

It is best to use basinex with conda to have gdal and NetCDF installed properly. To use the development version of basinex, download this repository and do the following in your conda environment:

```
conda install -y gdal netcdf4 pyyaml cxx-compiler
pip install .
```

Then you can execute `basinex` in that conda environment.

1.3 Documentation

Here is a short introduction about how to use the basin extractor. Have a look at the example directory or try it out directly with:

```
basinex -c examples
```

Usage

A command line script `basinex` will be installed with this package. You can execute it in your terminal and it will search for an `input.yml` file in your current directory.

To get more information about how to use the command line interface, you can have a look at the help message:

```
$ basinex -h
usage: basinex [-h] [-n LINE] [-i INPUT] [-v] [-c CWD] [--version]

mHM basin extractor

optional arguments:
  -h, --help            show this help message and exit
  -n LINE, --line LINE  the gauge to extract, given as its (0-based) line number in␣
↪the look up table
  -i INPUT, --input INPUT
                        the input yaml file to read (default: 'input.yml')
  -v, --verbose         give some status output
  -c CWD, --cwd CWD     the working directory
  --version             show program's version number and exit
```

The input file

The main input file `input.yml` is documented and should (hopefully) give an overview

The default input file looks like this:

```yaml
outpath: /path/to/output/
flowacc: /path/to/facc.asc
flowdir: /path/to/fdir.asc
gauges: /path/to/lut.txt
matching:
  scaling_factor: 0.001
  max_distance: 800
  max_error: 0.8
mask:
  fname: basin.asc
  outpath: morph
gauge:
  fname: idgauges.asc
  outpath: morph
gridfiles:
  - fname: /path/to/input/facc.asc
    outpath: morph
  - fname: /path/to/input/input1.asc
    outpath: morph
  - fname: /path/to/input/input2.asc
    outpath: luse
ncfiles:
  - fname: /path/to/input/input1.nc
    outpath: meteo
    ydim: northing
    xdim: easting
    y_shift: 0.5
    x_shift: 0.5
  - fname: /path/to/input/input2.nc
    outpath: meteo
    ydim: 'y'
    xdim: 'x'
```

Description

- `outpath:` `outpath/gauge_id/` - **Required**: Output location, all data will be writen to outpath/gauge_id/
- `flowacc:` `/path/to/facc.asc` - **Required**: flowaccumulation
- `flowdir:` `/path/to/fdir.asc` - **Required**: flowdirection
- `gauges:` `/path/to/lut.txt` - **Required**: gauging data lookup table Structure of the table:
  - A simple text table with seperator ';'
  - if the basin should be delineated, the following fields are required:
    - `id`: an unique gauging station identifier
    - `size`: size of the catchment
    - `y`: y coordinate of the gauging station
    - `x`: x coordinate of the gauging station
  - if an pre processed basin mask should be used, the following fields are required:

        * `id`: an unique basin identifier
        * `path`: path to the mask file
        * `varname`: name of the mask variable (optional, only needed if the mask is stored in a netcdf file)

- `latitude-size-correction: False` - **Optional**: perform a latitude correction for the given basin size (default: False)
  - `AREA = N_cells * res_x * ( cos(LAT) * res_y ) * scaling factor^2`
- `matching:` - **Required**: gauge matching parameters
  - **Note**: The gauge matching is based on the flowaccumulation data. The value for any given cell in the flowaccumulation grid is interpreted as the size [in cells] of a river basin drainig into the respective cell. During gauge matching the flowaccumulation grid is searched for a cell with a corresponding basin size close to the given gauge basin size. The search radius will be increased succesively and can be limited to a maximum size. As soon as a matching cell is found (error between catchment sizes is smaller than the given maximum error) the search ends.
  - `scaling_factor: .001` - scaling factor to account for the (possible) unit differences between the flowaccumulation and the gauging data. In order to make the data comparable the effective flowaccumulation will be caclulated as:
    * `flowaccumulation_value * (cellsize * scaling_factor)^2`
  - `max_distance: 800` - maximum distance [in map units] around a given gauging station location to search for a matching cell
  - `max_error: 0.8` - maximum error, as a fraction of the given basin size
- `mask:` - **Optional**: Write the delineated basin
  - `fname: basin.asc` - **Optional**: file name of the mask grid (default: `mask.asc`)
  - `outpath: morph` - output subdirectory
- `gauge:` - **Optional**: Write the gauge basin
  - `fname: idgauges.asc` - **Optional**: file name of the gauge grid (default: `idgauges.asc`)
  - `outpath: morph` - output subdirectory
- `gridfiles:` - **Optional**: Any number of grid files to extract.
  - **Note**: currently only the formats ArcAscii and GeoTIFF are supported
  - `fname: /path/to/input/facc.asc` - flow accumulation and flow direction won't be written unless listed here
  - `outpath: morph` - **Optional**: output subdirectory under outpath/gauge_id
- `ncfiles:` - **Optional**: Any number of netcdf files to extract.
  - **Note**: In order to extract from netcdf, coordinate values must be given.
    * **Example**: If your data variables depend on the three dimensions `time`, `y`, `x` your file should also contain the two one-dimensional (!) variables `y` (depending solely on the dimension `y`) and `x` (depending solely on the dimension `x`). Tools like `cdo` tend to silently remove variables, so double check, that this information is avaialable
  - `fname: /path/to/input/input1.nc`
  - `outpath: meteo` - **Optional**: output subdirectory under outpath/gauge_id
  - `ydim: northing` - **Required**: name of the (1D-) variable holding the y coordinates
  - `xdim: easting` - **Required**: name of the (1D-) variable holding the x coordinates
  - `y_shift: .5` and `x_shift: .5` - **Optional**: Coordinates of spatial data are definied on a certain location of the cell they belong to (e.g. upper or lower left corner). All the supported file formats handle coordinates transparently, with excpetion of netcdf. To account for the flexibility the format offers, it

is possible to specify the fraction of a cell the origin is shifted from the upper left corner in x and y direction. The bounding box of the dataset (an imaginary box, that contains exactly the entire spatial domain) is then caclulated as:

* `ymin = min(y_values) - (cellsize * (1 - y_shift))`
* `ymax = max(y_values) + (cellsize * y_shift)`
* `xmin = min(x_values) - (cellsize * (1 - x_shift))`
* `xmax = max(x_values) + (cellsize * x_shift)`
* **Examples**:
  - Your coordinate values specify the upper left corner of a cell
  - `y_shift: 0`
  - `x_shift: 0`
  - Your coordinate values specify the center of a cell:
  - `y_shift: 0.5`
  - `x_shift: 0.5`
  - Your coordinate values specify the lower left corner of a cell
  - `y_shift: 1`
  - `x_shift: 0`
* Default: lower left corner, i.e:
  - `y_shift: 1`
  - `x_shift: 0`

1.4 Notes

This package was orginally developed by David Schäfer who also provides a standalone version of the geoarray subpackage.

The `netcdf4` and `geoarray` subpackages have been taken from the jams-python package, that was formerly developed at the CHS department at the UFZ and is now released under the MIT license.

1.5 License

LGPLv3

CHAPTER 2

EXAMPLE FOR BASINEX

This is the river Elbe with one gauge at Decin. Needed files are `facc.asc` with the flow accumulation and `fdir.asc` with the flow direction. In addition, we cut out a NetCDF file containing precipitation.

Just execute the basin extractor in this directory:

```
basinex
```

2.1 Data sources

DEM related

The two DEM derivatives (`fdir` and `facc`) are based on upscaled terrain elevation data, that was collected from USGS EROS Archive—Digital Elevation—Global Multi-resolution Terrain Elevation Data 2010 (GMTED2010), available at: https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-global-multi-resolution-terrain-elevation.

Precipitation

Sample ERA5 meteo file, available at: https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=overview

> Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D. and Simmons, A., 2020. The ERA5 global reanalysis. Quarterly Journal of the Royal Meteorological Society, 146(730), pp.1999-2049

Gauge

Gauge IDs are obtained from the GRDC database: https://www.bafg.de/GRDC/EN/Home/homepage_node.html

CHAPTER 3

CHANGELOG

All notable changes to **basinex** will be documented in this file.

3.1 0.2 - 2022-07

Enhancements

- updated `report.out` with catchment area comparison and adjusted x, y of gauges (#3)
- `mask`, `gauge`, `gridfiles` and `ncfiles` are truly optional now (#4)

Bugfixes

- `netcdf4` version needs to be `<1.6`, so we added a restriction to `setup.cfg` (#3)

3.2 0.1 - 2022-02

Enhancements

- modern package structure: https://github.com/mhm-ufz/basinex
- added documentation: https://basinex.readthedocs.io
- made installable: `pip install basinex`
- added entry point for script usage
- added `latitude-size-correction` switch to yaml file for basin area correction with latlon coordinates
- added `--cwd` and `--version` to the basinex CLI

Changes

- added ufz dependecies to src

Bugfixes

- solved yaml warnings