Deltares

Delft Software Days
imod-python

Huite Bootsma

Joeri van Engelen

Martijn Visser

Julian Hofer

November 2021

Table of contents

Introduction

Working with unstructured grids

Unstructured grids in imod-python

Documentation improvements

Developments for 2022

Deltares

Summary: imod-python

A collection of Python tools for groundwater modeling and MODFLOW input & output

- Preparing: e.g. rasterizing river shapefiles
- Formatting: produce MODFLOW input
- Extracting and post-processing: compare heads with piezometers, compute water balance
- Visualization: time series, map, cross-sections, 3D

Part of a much broader set of tools for reproducible groundwater modeling

Set of tools

Scripting language

Scripts to a workflow

Data representation

Script version control

Data version control

File format

Collaboration

Link to MODFLOW

Example: NHI fresh-salt

(National Hydrological Instruments)

iMOD-Water Quality model (structured):

SEAWAT + PH3TD + bells & whistles

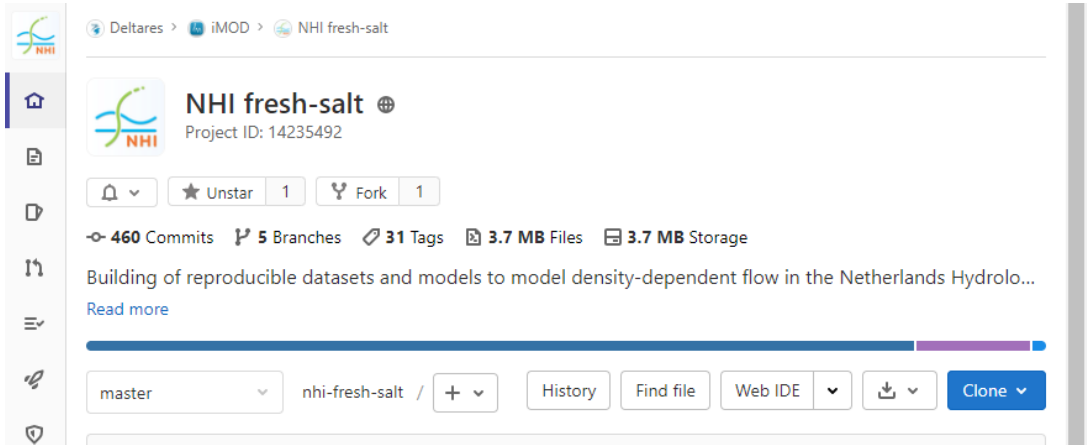
50 layers, 1300 rows, 1200 columns

- Fully scripted
- In version control
- One workflow from external data to figures

Openly available at:

https://gitlab.com/deltares/imod/nhi-fresh-salt

Deltares



Deltares › iMOD › NHI fresh-salt

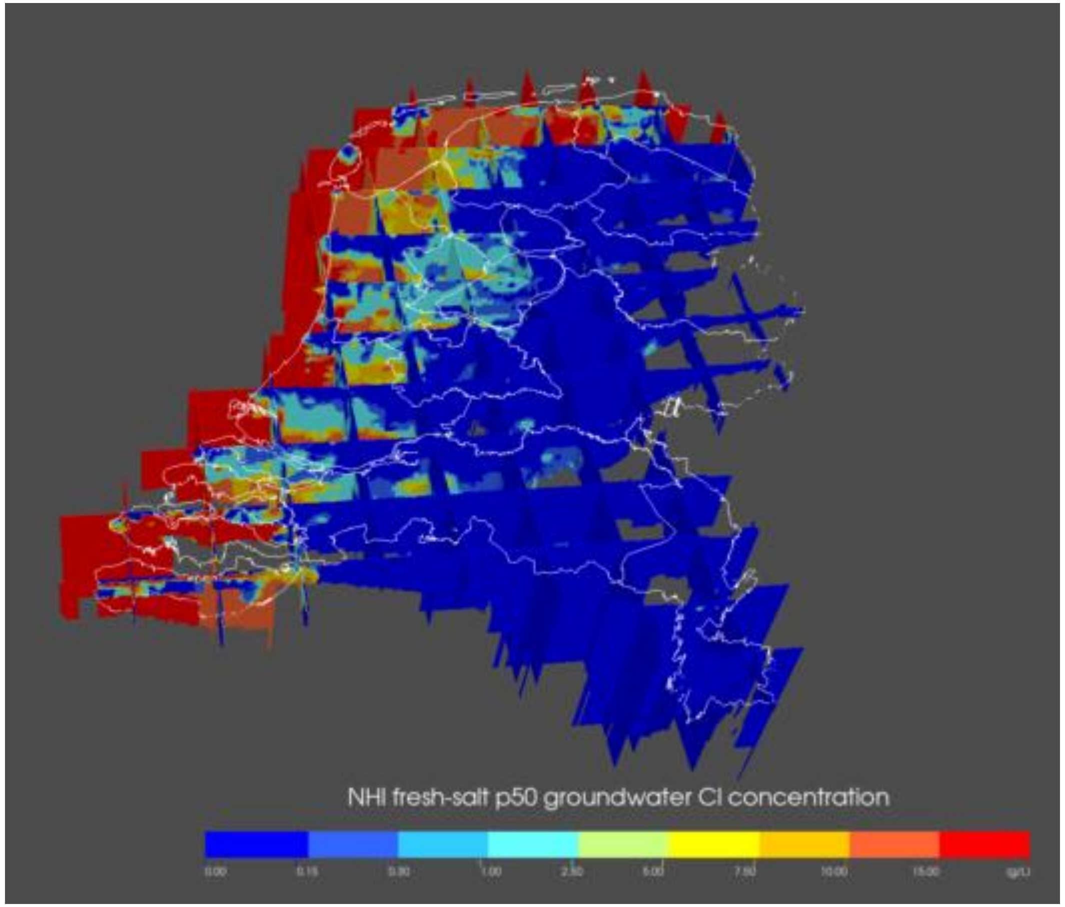
NHI fresh-salt

Project ID: 14235492

460 Commits · 5 Branches · 31 Tags · 3.7 MB Files · 3.7 MB Storage

Building of reproducible datasets and models to model density-dependent flow in the Netherlands Hydrolo...

Read more



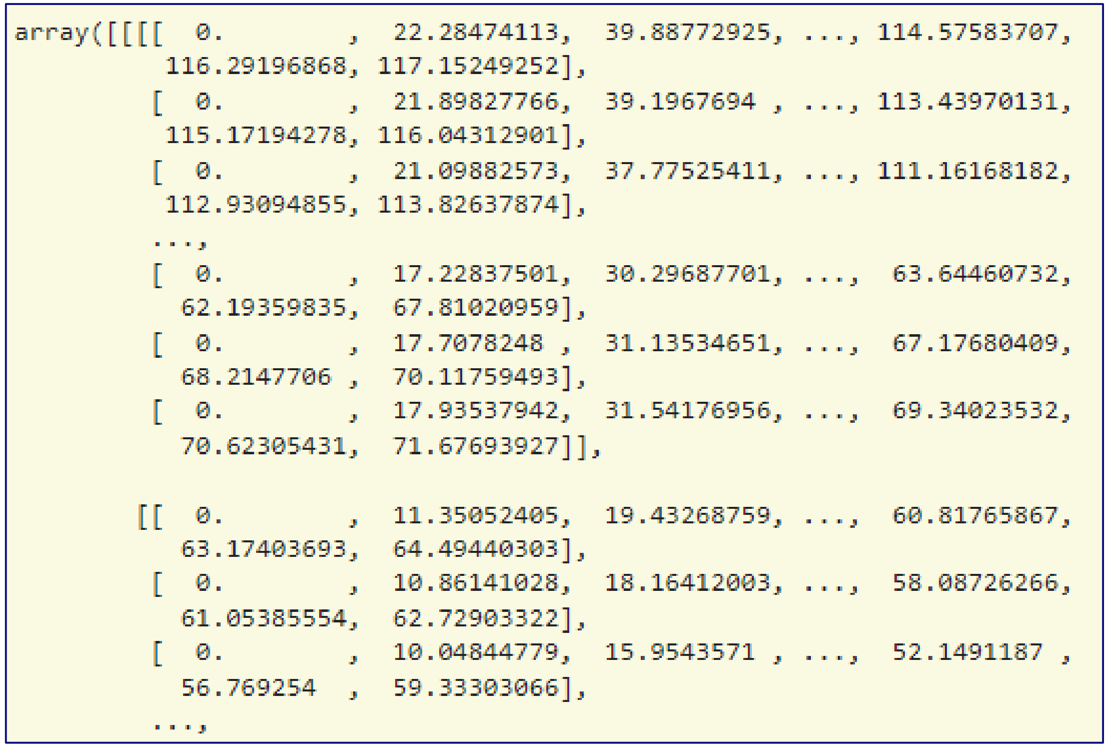
NHI fresh-salt p50 groundwater Cl concentration

In comparison with FloPy

flopy: fundamental data structure is numpy	imod: fundamental data structure is xarray
is easy to install: pip install flopy	is a large install: mamba install imod
supports every member of the USGS MODFLOW family	supports iMODFLOW, iMOD-WQ, MODFLOW6
supports nearly every option of MODFLOW6	supports a selection of MODFLOW6 options
defaults to text formats	defaults to (faster) binary formats, aiming at large models
provides a complete, but “low-level” interface	provides an incomplete, but “high-level” interface

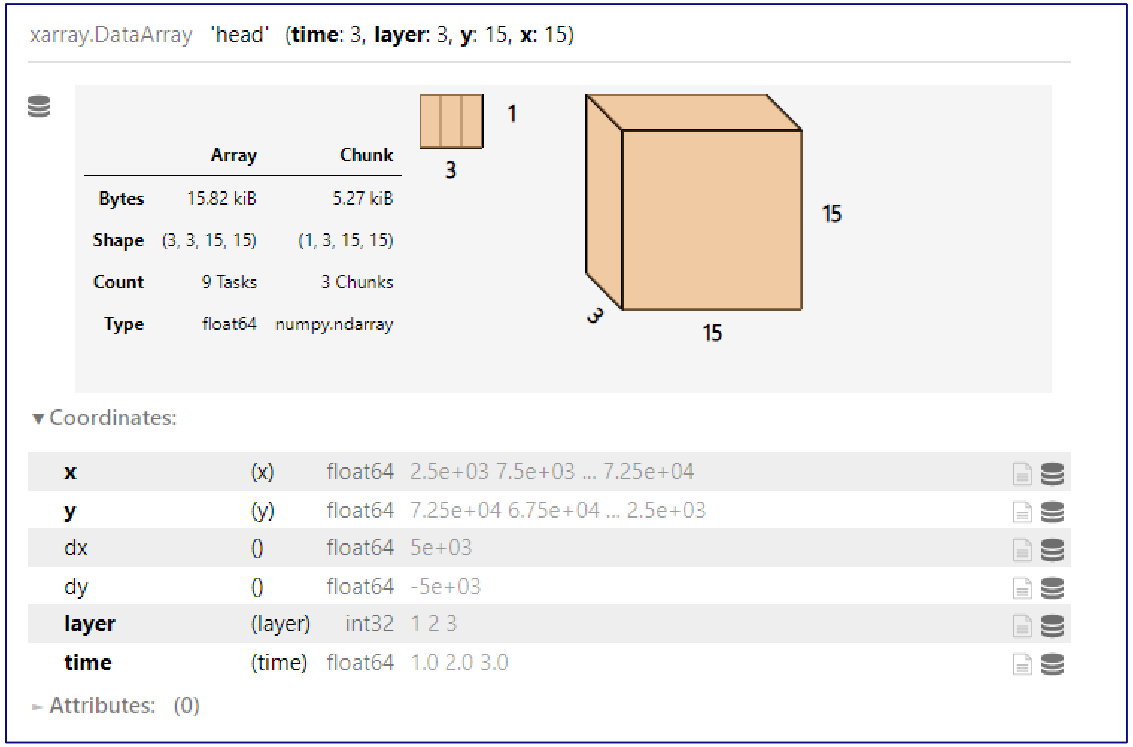
What does “high-level” mean?

```
hds = flopy.utils.binaryfile.HeadFile(
  "GWF_1/GWF_1.hds“
)
head = hds.get_data()
head
```



```
array([[[[  0.        ,  22.28474113,  39.88772925, ..., 114.57583707,
          116.29196868, 117.15249252],
         [  0.        ,  21.89827766,  39.1967694 , ..., 113.43970131,
          115.17194278, 116.04312901],
         [  0.        ,  21.09882573,  37.77525411, ..., 111.16168182,
          112.93094855, 113.82637874],
         ...,
         [  0.        ,  17.22837501,  30.29687701, ...,  63.64460732,
           62.19359835,  67.81020959],
         [  0.        ,  17.7078248 ,  31.13534651, ...,  67.17680409,
           68.2147706 ,  70.11759493],
         [  0.        ,  17.93537942,  31.54176956, ...,  69.34023532,
           70.62305431,  71.67693927]],

        [[  0.        ,  11.35052405,  19.43268759, ...,  60.81765867,
           63.17403693,  64.49440303],
         [  0.        ,  10.86141028,  18.16412003, ...,  58.08726266,
           61.05385554,  62.72903322],
         [  0.        ,  10.04844779,  15.9543571 , ...,  52.1491187 ,
           56.769254  ,  59.33303066],
         ...,
```

```
head = imod.mf6.open_hds(
  "GWF_1/GWF_1.hds",
  "GWF_1/dis.dis.grb",
)
head
```



xarray.DataArray 'head' (**time**: 3, **layer**: 3, **y**: 15, **x**: 15)

	Array	Chunk
Bytes	15.82 kiB	5.27 kiB
Shape	(3, 3, 15, 15)	(1, 3, 15, 15)
Count	9 Tasks	3 Chunks
Type	float64	numpy.ndarray

▾Coordinates:

x	(x)	float64	2.5e+03 7.5e+03 ... 7.25e+04
y	(y)	float64	7.25e+04 6.75e+04 ... 2.5e+03
dx	()	float64	5e+03
dy	()	float64	-5e+03
layer	(layer)	int32	1 2 3
time	(time)	float64	1.0 2.0 3.0

▸Attributes: (0)

What does “high-level” mean?

Now let’s assume the heads are (much) too big to fit in memory (e.g. 100 GB).
How to compute the mean of the simulated head over time?

```
hds = flopy.utils.binaryfile.HeadFile(
   "GWF_1/GWF_1.hds“
)
times = hds.get_times()
accumulator = hds.get_data(totim=times[0])

for time in times[1:]:
   accumulator += hds.get_data(totim=time)

result = accumulator / len(times)
```

FloPy

```
head = imod.mf6.open_hds(
   "GWF_1/GWF_1.hds",
   "GWF_1/dis.dis.grb",
)

result = head.mean("time")
```

A high level interface is **convenient**, **scalable**, and **extensible**.

Deltares

Unstructured: 3 problems

Scripting language

Script version control

Collaboration

Scripts to a workflow

Data version control

Data representation

File format

Link to MODFLOW

Unstructured: problem & solution

File format: no widely used convention for unstructured grid data

File format:
UGRID Conventions

Data representation: xarray does not fully represent unstructure data

Data representation:
create new data structure

Imod-python: relies on xarray to represent data

imod-python:
Use new data structure

https://ugrid-conventions.github.io/ugrid-conventions/

Working with unstructured grids

- Xarray + UGRID: Xugrid
- Extension of Xarray, specifically for unstructured grids
- Xugrid automatically reads the grid specification from a UGRID netCDF and returns an “xarray-like” data structure
- When possible, join forces with NSF-funded Project Raijjin
- Behaves like Xarray behavior where possible

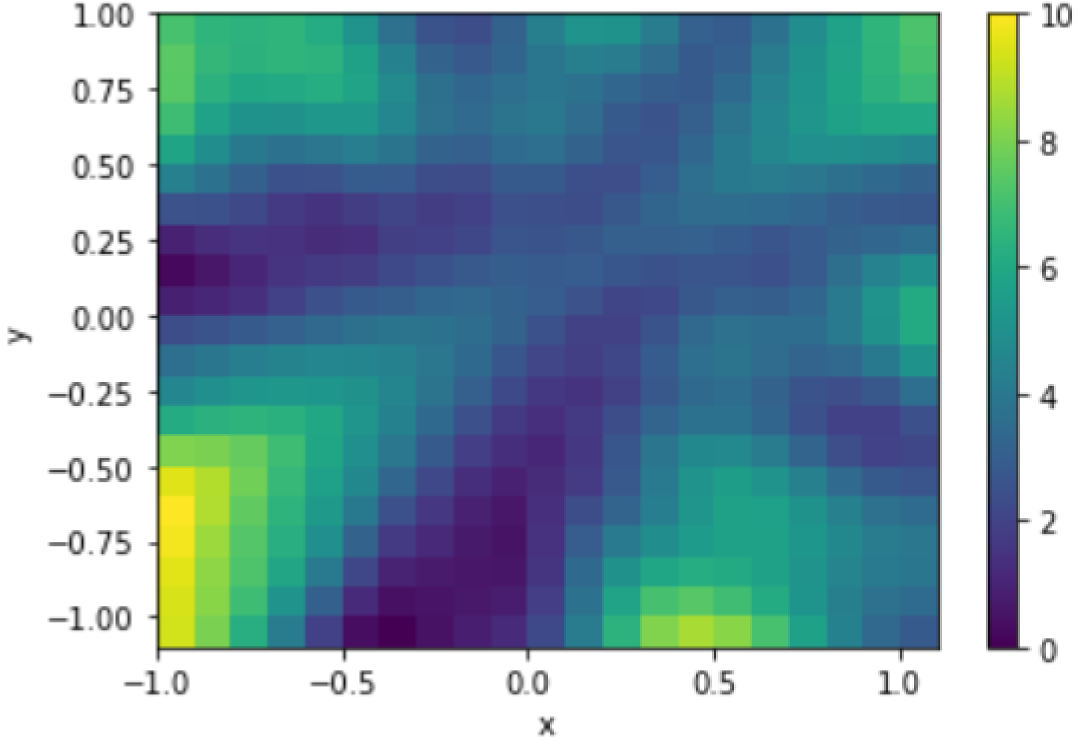
https://github.com/Deltares/xugrid

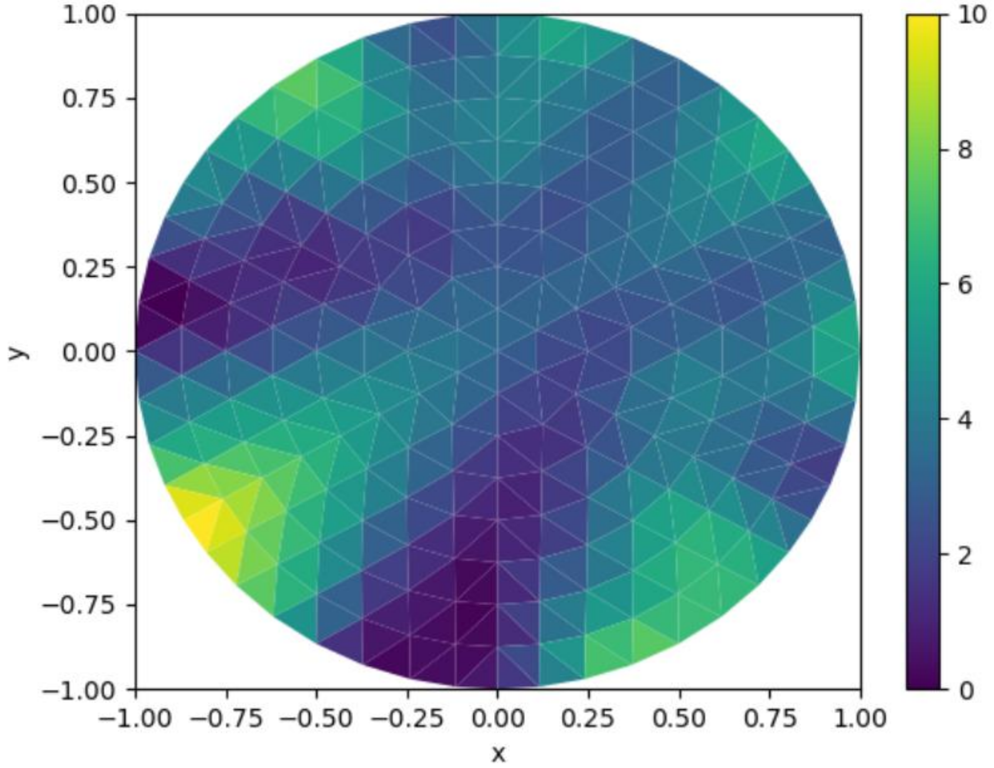
https://raijin.ucar.edu/
https://github.com/UXARRAY/uxarray

Behaves like Xarray where possible

structured.plot()

unstructured.ugrid.plot()





Deltares

Unstructured grids in imod-python

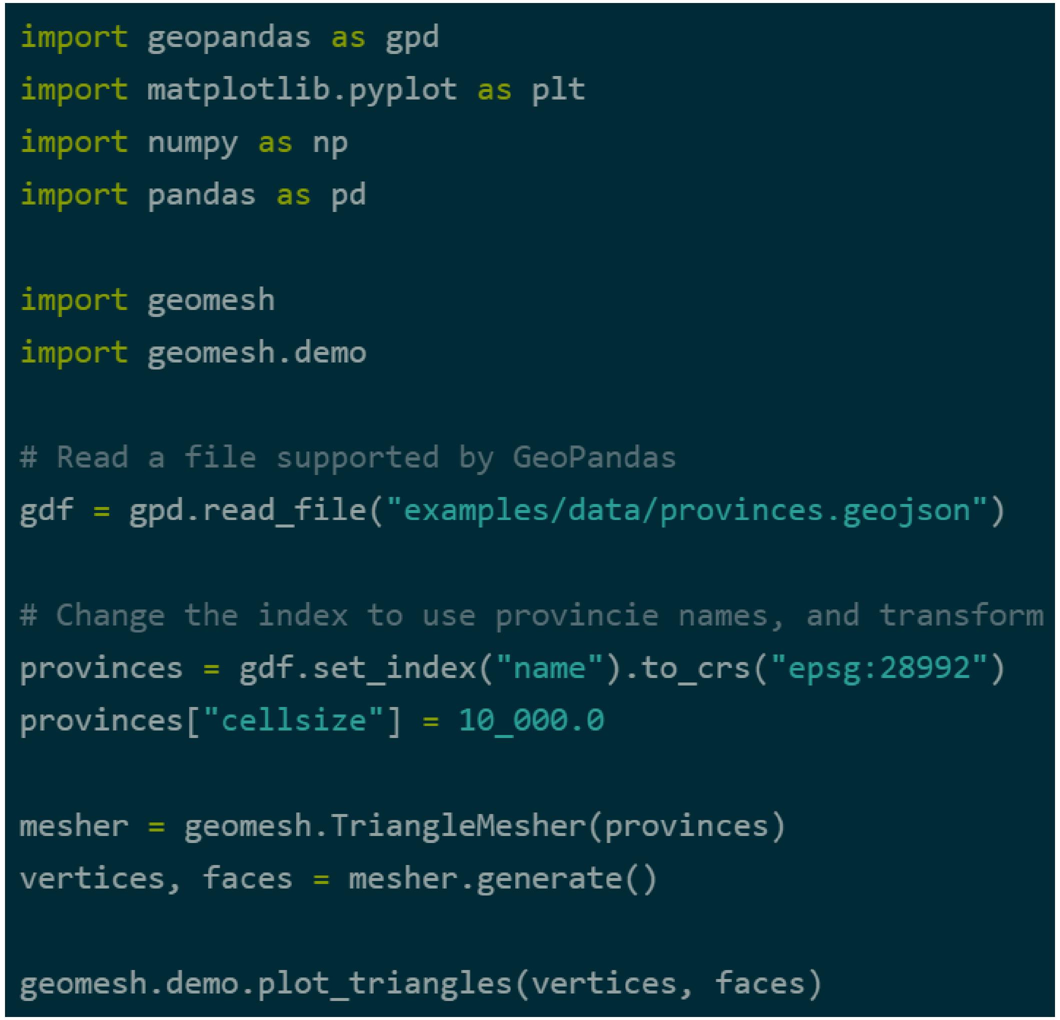
The classes in imod-python have been expanded to take xugrid.UgridDataArray objects next to xarray.DataArray objects, and the regridder changed to understand unstructured grids.

To build an unstructured model instead of a structured MODFLOW6 model:

- Create an unstructured mesh
- Use the imod-python Regridder to create unstructured UgridDataArray data
- Use the VerticesDiscretization instead of StructuredDiscretization

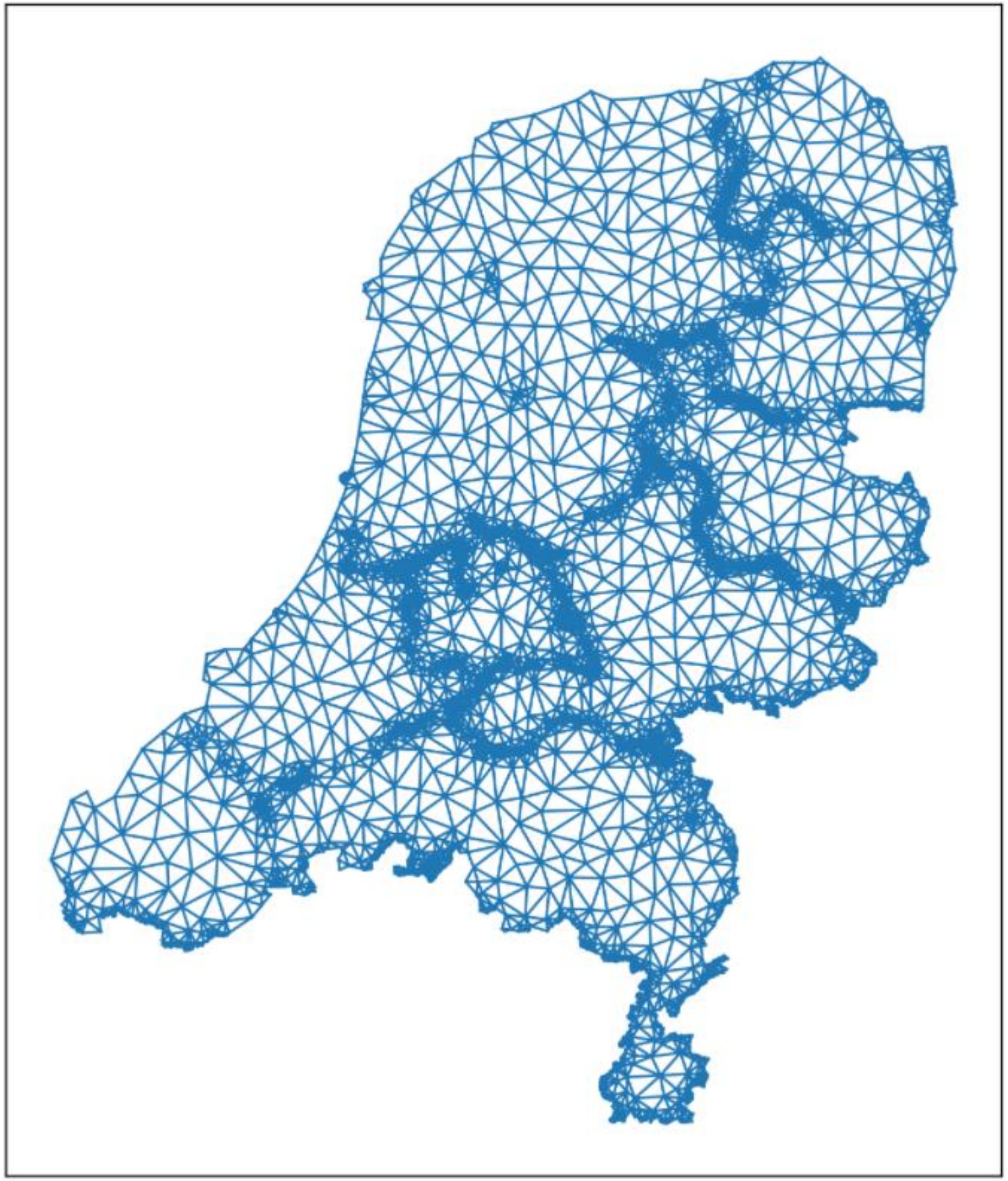
The examples in the documentation demonstrate.

Mesh generation

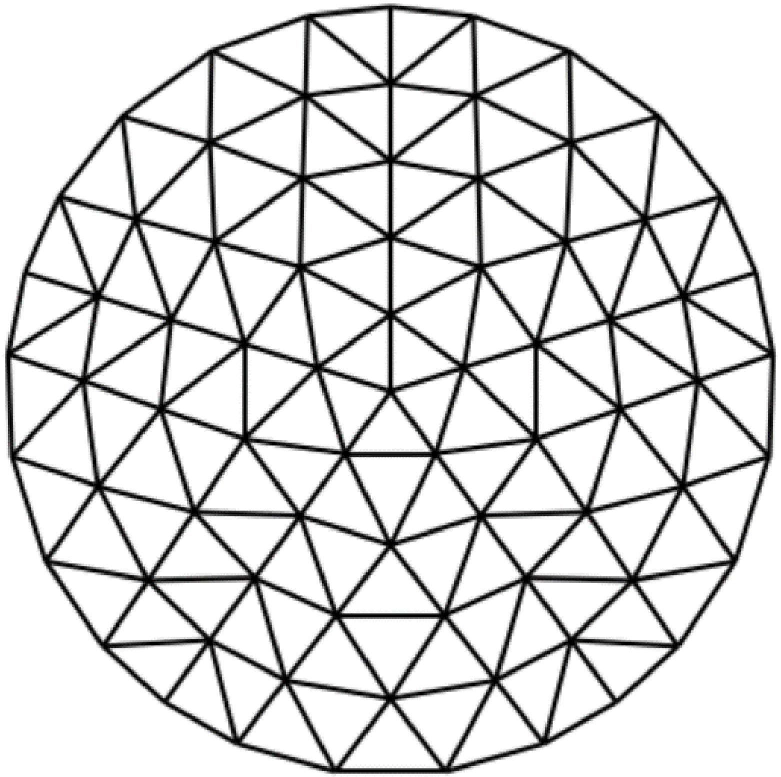


```
import geopandas as gpd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

import geomesh
import geomesh.demo

# Read a file supported by GeoPandas
gdf = gpd.read_file("examples/data/provinces.geojson")

# Change the index to use provincie names, and transform
provinces = gdf.set_index("name").to_crs("epsg:28992")
provinces["cellsize"] = 10_000.0

mesher = geomesh.TriangleMesher(provinces)
vertices, faces = mesher.generate()

geomesh.demo.plot_triangles(vertices, faces)
```

https://gitlab.com/deltares/imod/geomesh



Or just a simple mesh example



Documentation improvements

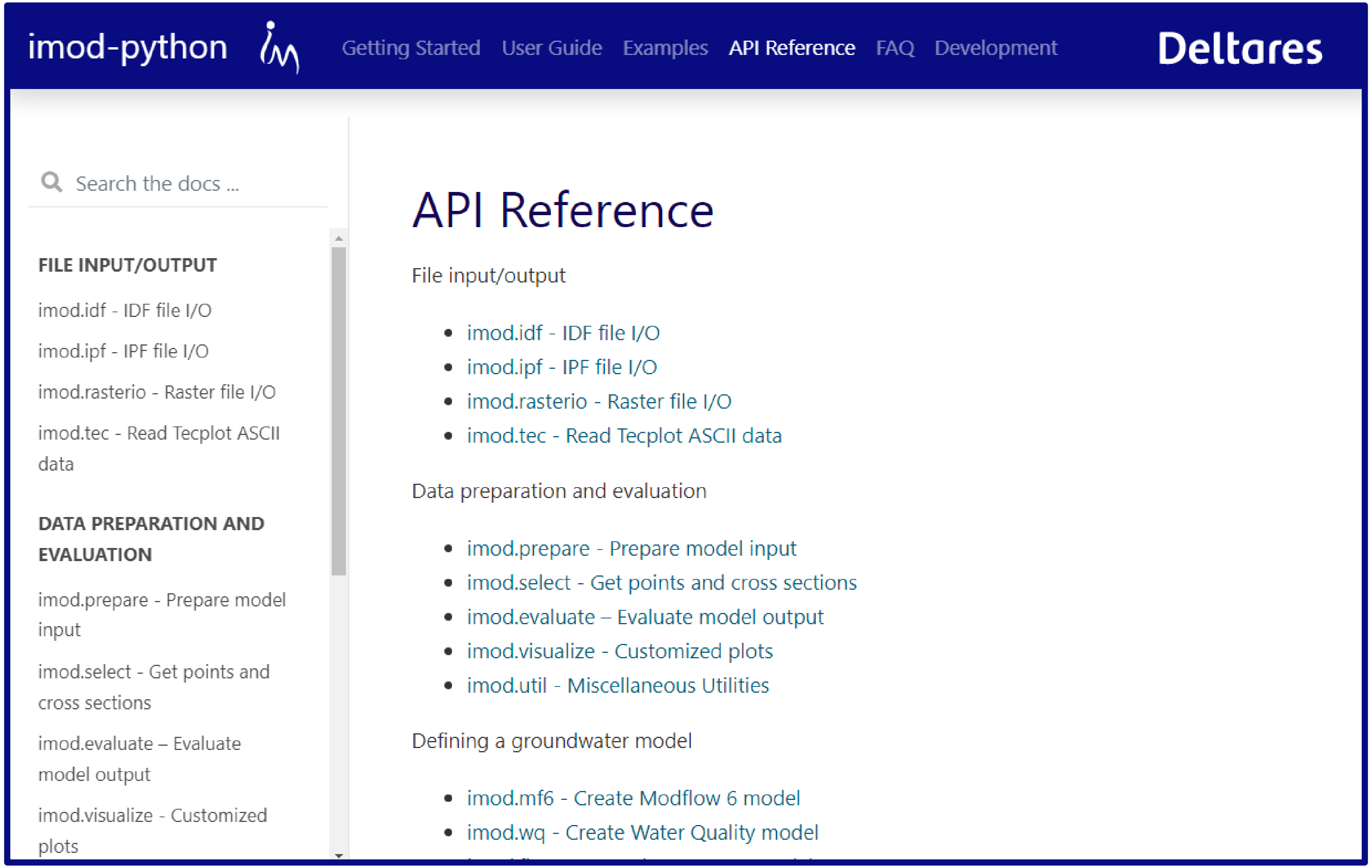
New theme

User Guide

Examples

Frequently Asked Questions (FAQ) /

“How do I …”



FILE INPUT/OUTPUT

imod.idf - IDF file I/O

imod.ipf - IPF file I/O

imod.rasterio - Raster file I/O

imod.tec - Read Tecplot ASCII data

DATA PREPARATION AND EVALUATION

imod.prepare - Prepare model input

imod.select - Get points and cross sections

imod.evaluate – Evaluate model output

imod.visualize - Customized plots

API Reference

File input/output

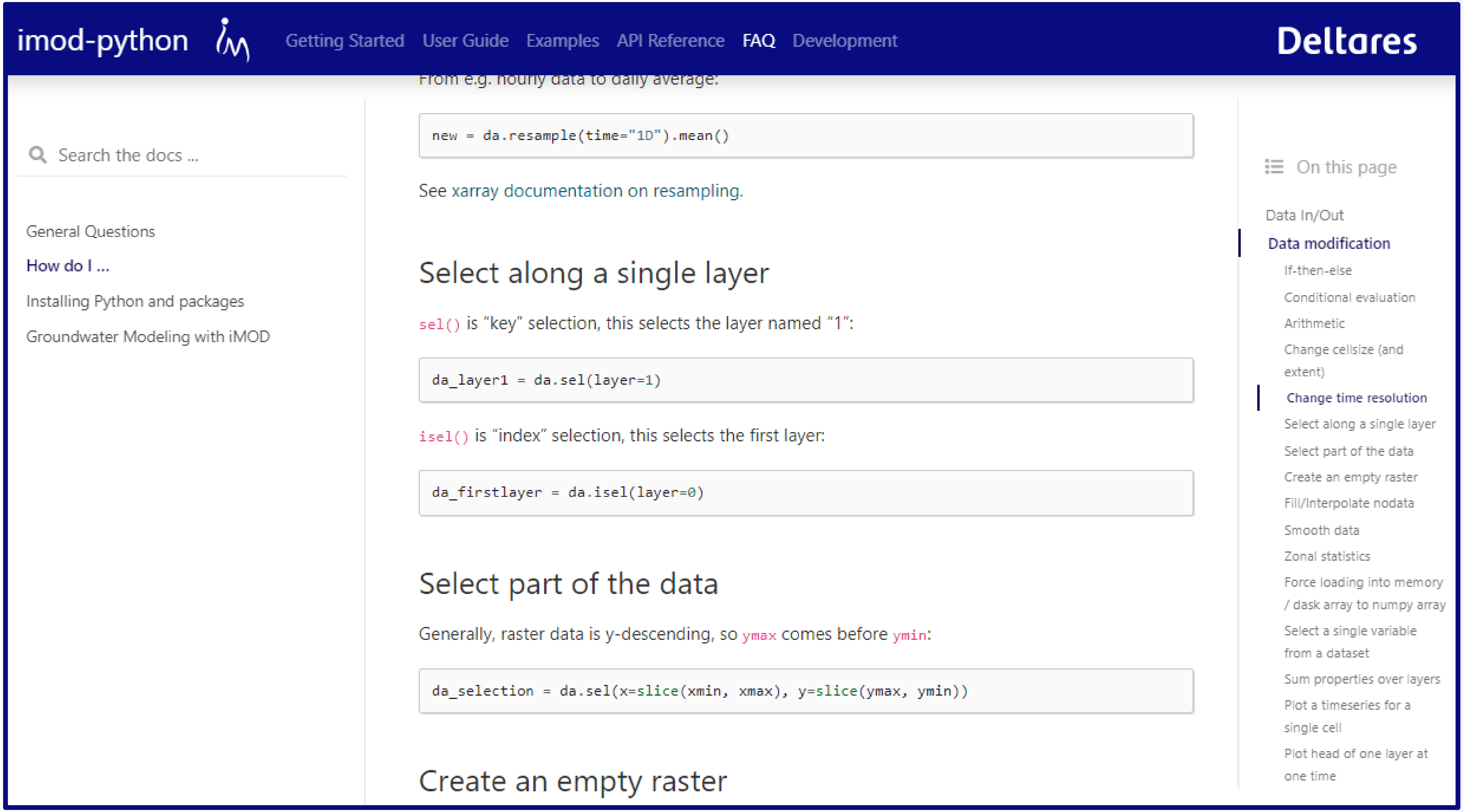
- imod.idf - IDF file I/O
- imod.ipf - IPF file I/O
- imod.rasterio - Raster file I/O
- imod.tec - Read Tecplot ASCII data

Data preparation and evaluation

- imod.prepare - Prepare model input
- imod.select - Get points and cross sections
- imod.evaluate – Evaluate model output
- imod.visualize - Customized plots
- imod.util - Miscellaneous Utilities

Defining a groundwater model

- imod.mf6 - Create Modflow 6 model
- imod.wq - Create Water Quality model



From e.g. hourly data to daily average:

```
new = da.resample(time="1D").mean()
```

See xarray documentation on resampling.

Select along a single layer

`sel()` is "key" selection, this selects the layer named "1":

```
da_layer1 = da.sel(layer=1)
```

`isel()` is "index" selection, this selects the first layer:

```
da_firstlayer = da.isel(layer=0)
```

Select part of the data

Generally, raster data is y-descending, so `ymax` comes before `ymin`:

```
da_selection = da.sel(x=slice(xmin, xmax), y=slice(ymax, ymin))
```

Create an empty raster

Developments for 2022

- Explore options for “simplified” use:
  - Develop a project file equivalent (automatic resampling/regridding in time, space)
  - Run functions through a command line interface (compare iMOD Batch functionality)
- Unsaturated zone: pre- and post-processing support for MetaSWAP
- Add support for MODFLOW6 Transport and Buoyancy for solute transport and variable density modeling
- Add more utilities for unstructured grids, explore unstructured grids without layers (DISU)
- Add MODPATH support for particle tracking
- Connection to surface water modules