Using psyplot for visualizing unstructured data and vertical transects

9th Data Science Symposium

Philipp S. Sommer

Helmholtz-Zentrum Hereon, Helmholtz Coastal Data Center

April 4th, 2024, Bremen

Help

https://github.com/Chilipp/psyplot-Data-Science-Symposium-20240404

Technical Note

This presentation is a jupyter notebook presented with rise for interactive execution of the cells. You can run it interactively on mybinder in your browser:

launch binder

The link to the repo on Github: https://github.com/Chilipp/psyplot-Data-Science-Symposium-20240404).

Back to first slide

Technical Note

This presentation is a jupyter notebook presented with rise for interactive execution of the cells. You can run it interactively on mybinder in your browser:

launch binder

The link to the repo on Github: https://github.com/Chilipp/psyplot-Data-Science-Symposium-20240404).

Back to first slide

So let's import some libraries for the execution

```
In [1]: %matplotlib widget

        import psyplot.project as psy

        import numpy as np
        import xarray as xr
        import matplotlib.pyplot as plt
        import cartopy.crs as ccrs
        from IPython.display import display, Video

        from ipympl.backend_nbagg import Canvas
        Canvas.header_visible.default_value = False

        import warnings
```

Outline

Overview on psyplot

Main features of psyplot

Unstructured Grids

Vertical Transects

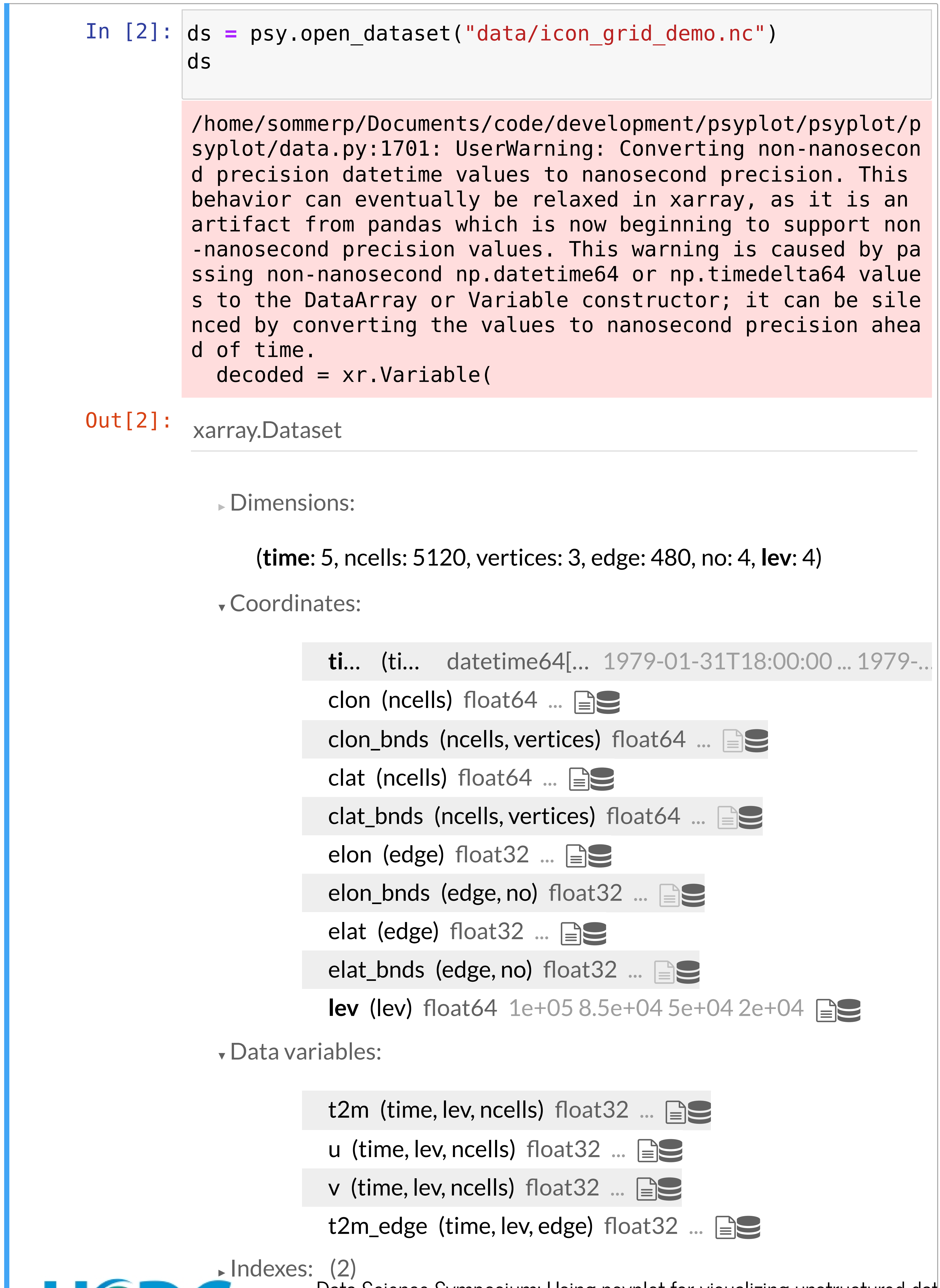
psyplot in one slide

> ***An interactive netCDF data visualization and analysis library, based upon `matplotlib` and `xarray`***

- the `psyplot` package is the core of the framework, visualization methods are implemented in plugins
- `formatoption`s control the appearance of the plot or the data that is shown
- multiple `formatoption` together make up a plotter for a specific type of visualization
    - one plotter for line plots
    - one plotter for scalar 2D plots
    - one plotter for scalar 2D plots on a map
    - one plotter for vector data (i.e. data with u and v component)
    - ...
- object-oriented approach
    - each formatoption and each plotter represents one class
    - formatoption can be combined to handle more complex configuration tasks
- `psyplot` API can be used
    - from the command-line
    - in python scripts (or jupyter notebooks)
    - in a desktop application

Main features of psyplot

Using psyplot from Python



In [2]:
```
ds = psy.open_dataset("data/icon_grid_demo.nc")
ds
```

```
/home/sommerp/Documents/code/development/psyplot/psyplot/p
syplot/data.py:1701: UserWarning: Converting non-nanosecon
d precision datetime values to nanosecond precision. This
behavior can eventually be relaxed in xarray, as it is an
artifact from pandas which is now beginning to support non
-nanosecond precision values. This warning is caused by pa
ssing non-nanosecond np.datetime64 or np.timedelta64 value
s to the DataArray or Variable constructor; it can be sile
nced by converting the values to nanosecond precision ahea
d of time.
  decoded = xr.Variable(
```

Out[2]:
```
xarray.Dataset

▸ Dimensions:
    (time: 5, ncells: 5120, vertices: 3, edge: 480, no: 4, lev: 4)
▾ Coordinates:
        ti...  (ti...  datetime64[...  1979-01-31T18:00:00 ... 1979-...
        clon  (ncells)  float64  ...
        clon_bnds  (ncells, vertices)  float64  ...
        clat  (ncells)  float64  ...
        clat_bnds  (ncells, vertices)  float64  ...
        elon  (edge)  float32  ...
        elon_bnds  (edge, no)  float32  ...
        elat  (edge)  float32  ...
        elat_bnds  (edge, no)  float32  ...
        lev  (lev)  float64  1e+05 8.5e+04 5e+04 2e+04
▾ Data variables:
        t2m  (time, lev, ncells)  float32  ...
        u  (time, lev, ncells)  float32  ...
        v  (time, lev, ncells)  float32  ...
        t2m_edge  (time, lev, edge)  float32  ...
▸ Indexes:  (2)
```

Using psyplot from Python

In [2]:
```
ds = psy.open_dataset("data/icon_grid_demo.nc")
ds
```

```
/home/sommerp/Documents/code/development/psyplot/psyplot/p
syplot/data.py:1701: UserWarning: Converting non-nanosecon
d precision datetime values to nanosecond precision. This
behavior can eventually be relaxed in xarray, as it is an
artifact from pandas which is now beginning to support non
-nanosecond precision values. This warning is caused by pa
ssing non-nanosecond np.datetime64 or np.timedelta64 value
s to the DataArray or Variable constructor; it can be sile
nced by converting the values to nanosecond precision ahea
d of time.
  decoded = xr.Variable(
```

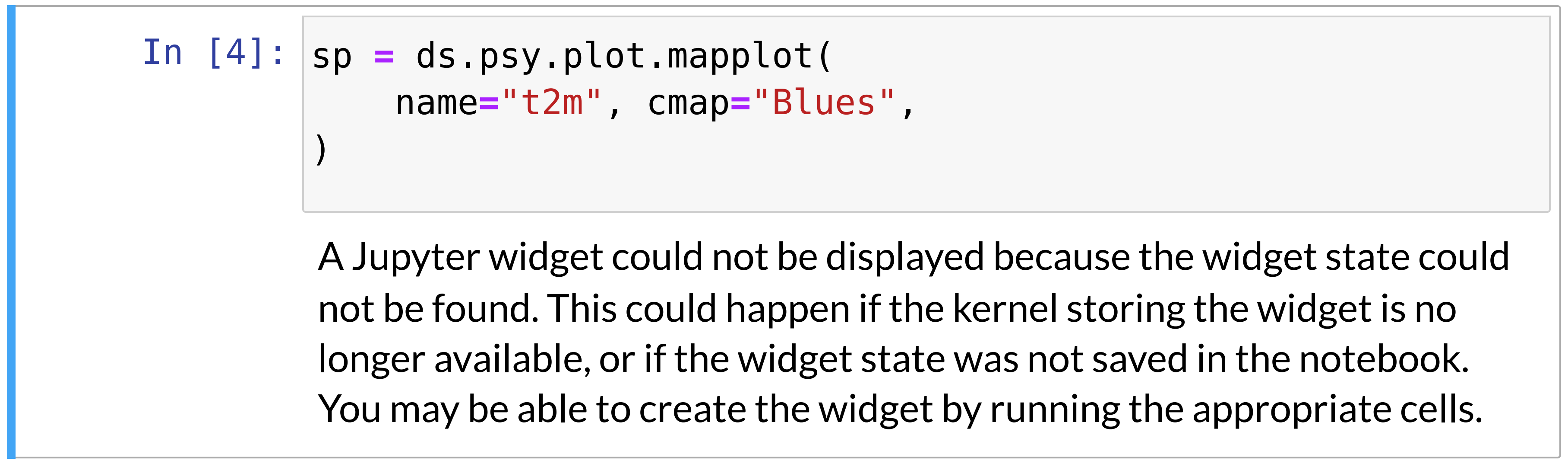
Out[2]:

```
xarray.Dataset

▸ Dimensions:

    (time: 5, ncells: 5120, vertices: 3, edge: 480, no: 4, lev: 4)

▾ Coordinates:

        ti...      (ti...     datetime64[...   1979-01-31T18:00:00 ... 1979-...
        clon       (ncells)               float64 ...
        clon_bnds  (ncells, vertices)     float64 ...
        clat       (ncells)               float64 ...
        clat_bnds  (ncells, vertices)     float64 ...
        elon       (edge)                 float32 ...
        elon_bnds  (edge, no)             float32 ...
        elat       (edge)                 float32 ...
        elat_bnds  (edge, no)             float32 ...
        lev        (lev)                  float64 1e+05 8.5e+04 5e+04 2e+04

▾ Data variables:

        t2m        (time, lev, ncells)    float32 ...
        u          (time, lev, ncells)    float32 ...
        v          (time, lev, ncells)    float32 ...
        t2m_edge   (time, lev, edge)      float32 ...

▸ Indexes: (2)
```

In [3]:
```
sp = ds.psy.plot.mapplot(
    name="t2m",
)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Working interactively from the command line

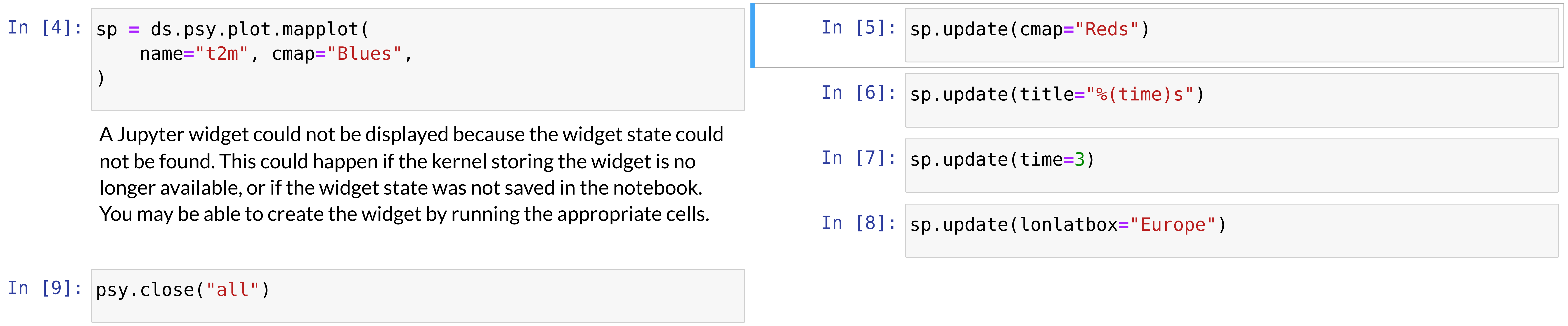


In [4]:

```python
sp = ds.psy.plot.mapplot(
    name="t2m", cmap="Blues",
)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Working interactively from the command line



```
In [4]: sp = ds.psy.plot.mapplot(
            name="t2m", cmap="Blues",
        )
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
In [9]: psy.close("all")
```

```
In [5]: sp.update(cmap="Reds")
```

```
In [6]: sp.update(title="%(time)s")
```

```
In [7]: sp.update(time=3)
```

```
In [8]: sp.update(lonlatbox="Europe")
```

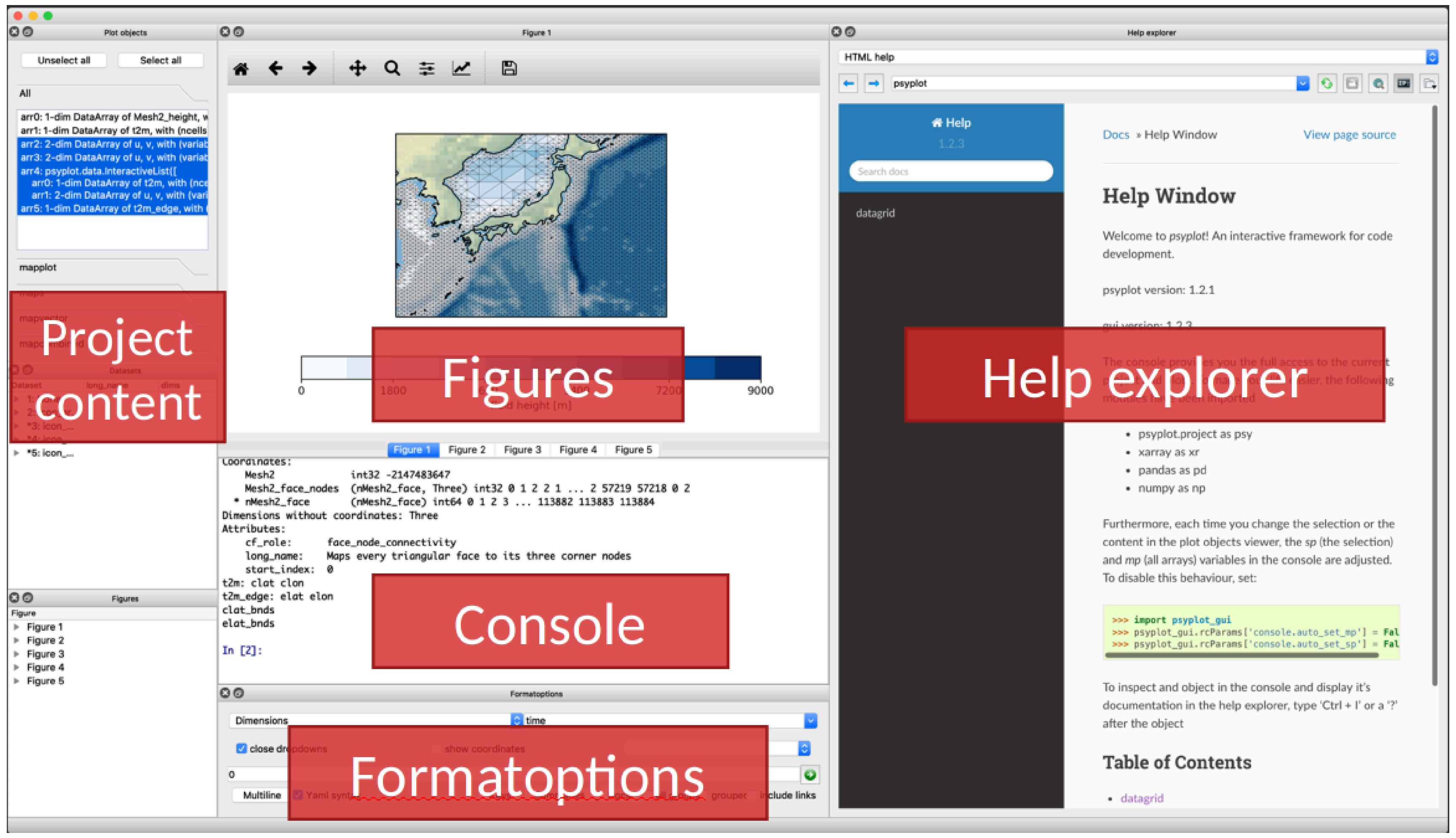
Using the GUI

psyplot comes with a flexible graphical user interface (GUI).

- On mybinder: click here.
- On mistral:
  - either via X11

    ```
    ssh -X mistral
    module load python3
    psyplot
    ```

- On your on own working station: Install it via `conda install -c conda-forge psy-view`



```
Coordinates:
    Mesh2             int32 -2147483647
    Mesh2_face_nodes  (nMesh2_face, Three) int32 0 1 2 2 1 ... 2 57219 57218 0 2
  * nMesh2_face       (nMesh2_face) int64 0 1 2 3 ... 113882 113883 113884
Dimensions without coordinates: Three
Attributes:
    cf_role:      face_node_connectivity
    long_name:    Maps every triangular face to its three corner nodes
    start_index:  0
t2m: clat clon
t2m_edge: elat elon
clat_bnds
elat_bnds

In [2]:
```

Help Window

Welcome to psyplot! An interactive framework for code development.

psyplot version: 1.2.1

- psyplot.project as psy
- xarray as xr
- pandas as pd
- numpy as np

Furthermore, each time you change the selection or the content in the plot objects viewer, the sp (the selection) and mp (all arrays) variables in the console are adjusted. To disable this behaviour, set:

```
>>> import psyplot_gui
>>> psyplot_gui.rcParams['console.auto_set_mp'] = Fal
>>> psyplot_gui.rcParams['console.auto_set_sp'] = Fal
```

To inspect and object in the console and display it's documentation in the help explorer, type 'Ctrl + I' or a '?' after the object

Table of Contents

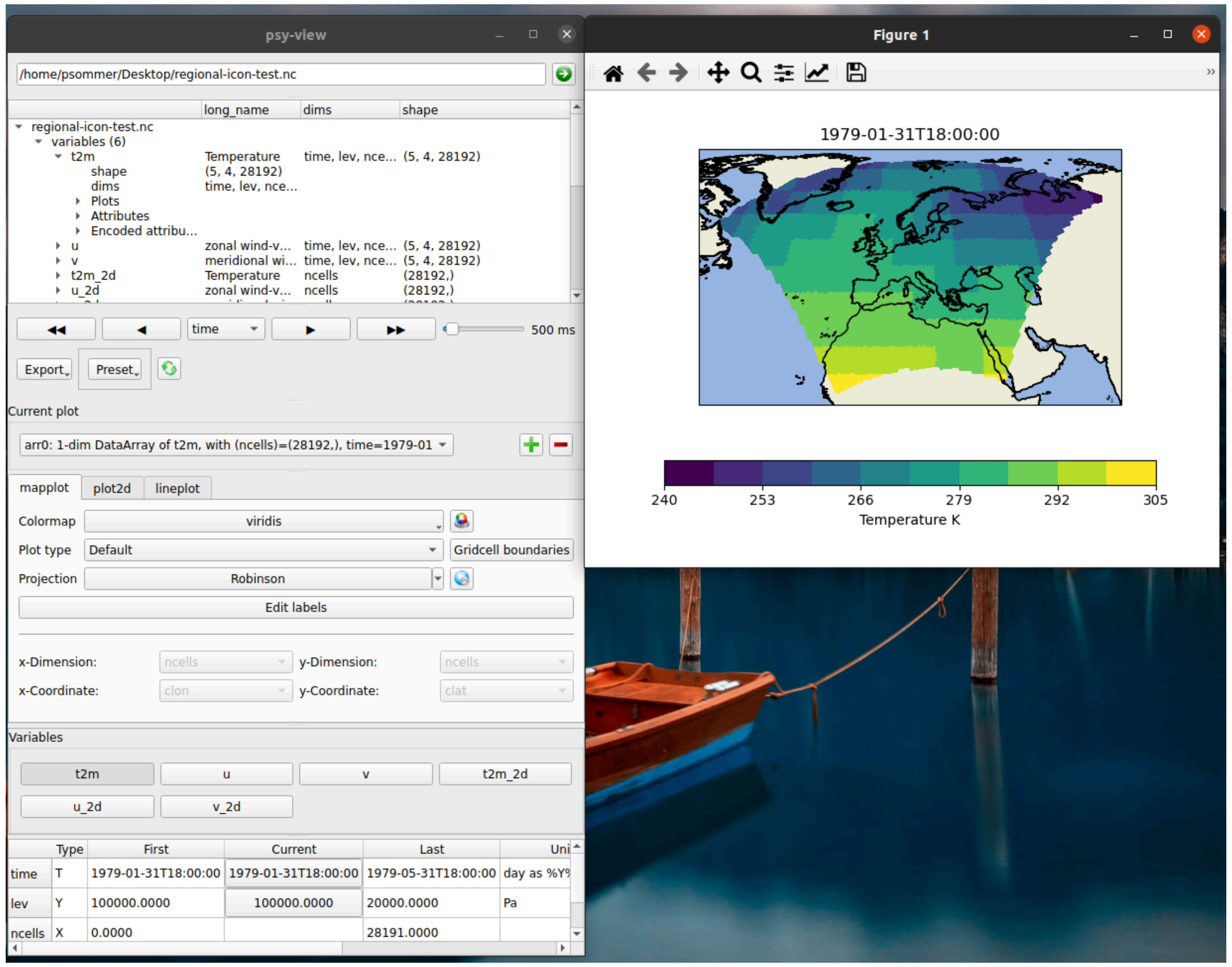
- datagrid

psy-view

An ncview-like interface, but with psyplot

- quick access to netcdf-variables via buttons
- switch between projections
- modify basemap
- change labels, colormaps, etc.
- display time-series when clicking on the map
- load presets for your plots
- animate through time, z, etc.

https://psyplot.github.io/psy-view



/home/psommer/Desktop/regional-icon-test.nc

	long_name	dims	shape
regional-icon-test.nc			
variables (6)			
t2m	Temperature	time, lev, nce...	(5, 4, 28192)
shape	(5, 4, 28192)		
dims	time, lev, nce...		
Plots			
Attributes			
Encoded attribu...			
u	zonal wind-v...	time, lev, nce...	(5, 4, 28192)
v	meridional wi...	time, lev, nce...	(5, 4, 28192)
t2m_2d	Temperature	ncells	(28192,)
u_2d	zonal wind-v...	ncells	(28192,)

Current plot

arr0: 1-dim DataArray of t2m, with (ncells)=(28192,), time=1979-01

Colormap: viridis

Plot type: Default

Projection: Robinson

x-Dimension: ncells
y-Dimension: ncells
x-Coordinate: clon
y-Coordinate: clat

Variables

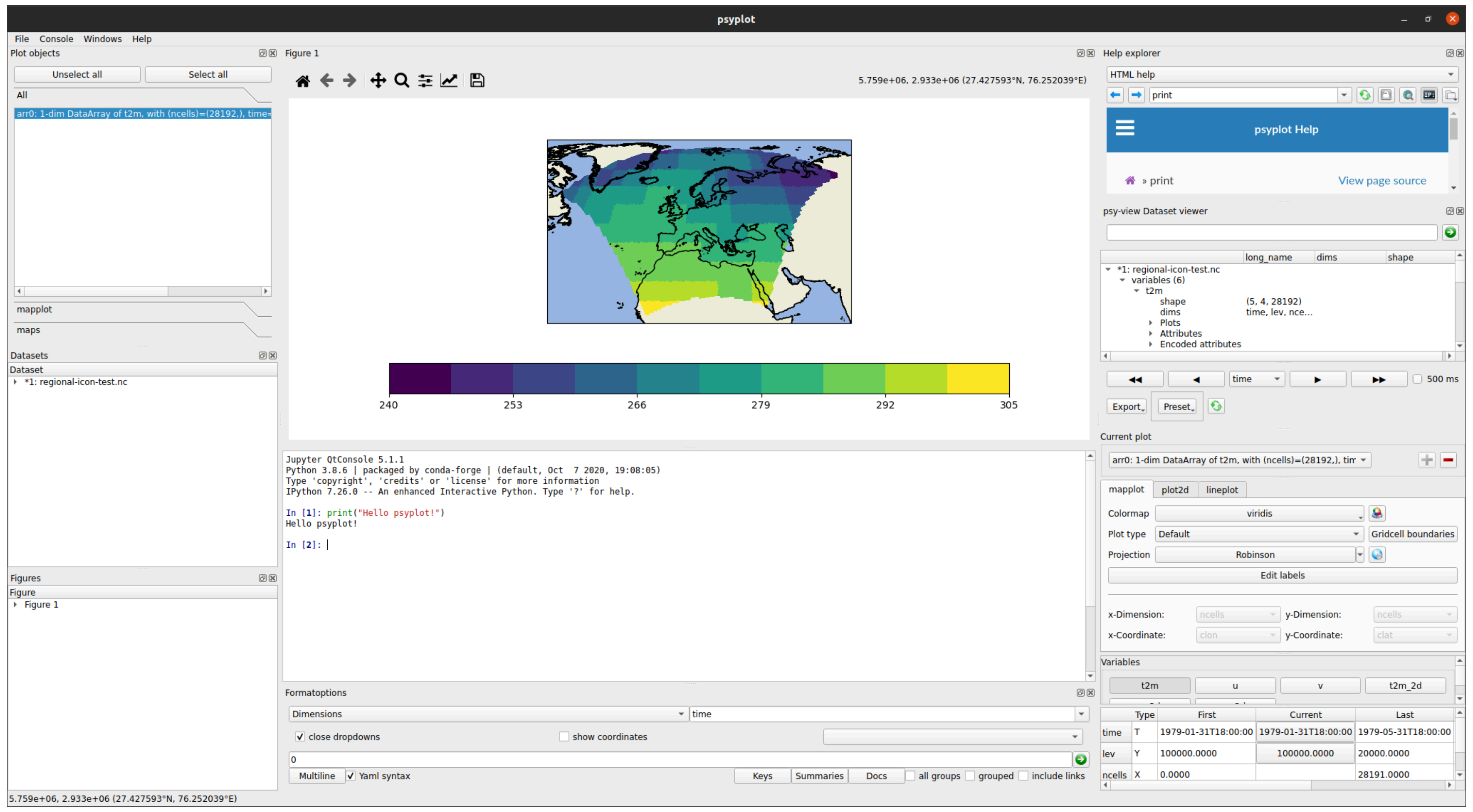
t2m | u | v | t2m_2d | u_2d | v_2d

	Type	First	Current	Last	Uni
time	T	1979-01-31T18:00:00	1979-01-31T18:00:00	1979-05-31T18:00:00	day as %Y%
lev	Y	100000.0000	100000.0000	20000.0000	Pa
ncells	X	0.0000		28191.0000	

The psyplot GUI

Flexible GUI for coding and clicking

- integrated IPython console for interactive use of the command line
- connected help window that renders help and python object documentation
- integrated psy-view window
- shortcut widgets for individual formatoptions



```
Jupyter QtConsole 5.1.1
Python 3.8.6 | packaged by conda-forge | (default, Oct  7 2020, 19:08:05)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.26.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print("Hello psyplot!")
Hello psyplot!

In [2]: 
```

	Type	First	Current	Last
time	T	1979-01-31T18:00:00	1979-01-31T18:00:00	1979-05-31T18:00:00
lev	Y	100000.0000	100000.0000	20000.0000
ncells	X	0.0000		28191.0000

https://psyplot.github.io/psyplot-gui

Plugins for visualization

psyplot is the core that defines the framework (Plotter, Formatoption, Project, CFDecoder), the plot methods are implemented by plugins:

- `psy-simple`: for standard 1D and 2D plot
  - e.g. lineplot, plot2d, vector, barplot
- `psy-maps`: for georeferenced plots (i.e. maps)
  - e.g. mapplot, mapvector, etc.
- `psy-reg`: for regression analysis
  - linreg, densityreg

New plugins

- `psy-ugrid`: for decoding UGRID conventions
- `psy-transect`: for extracting, visualizing and analyzing vertical transects

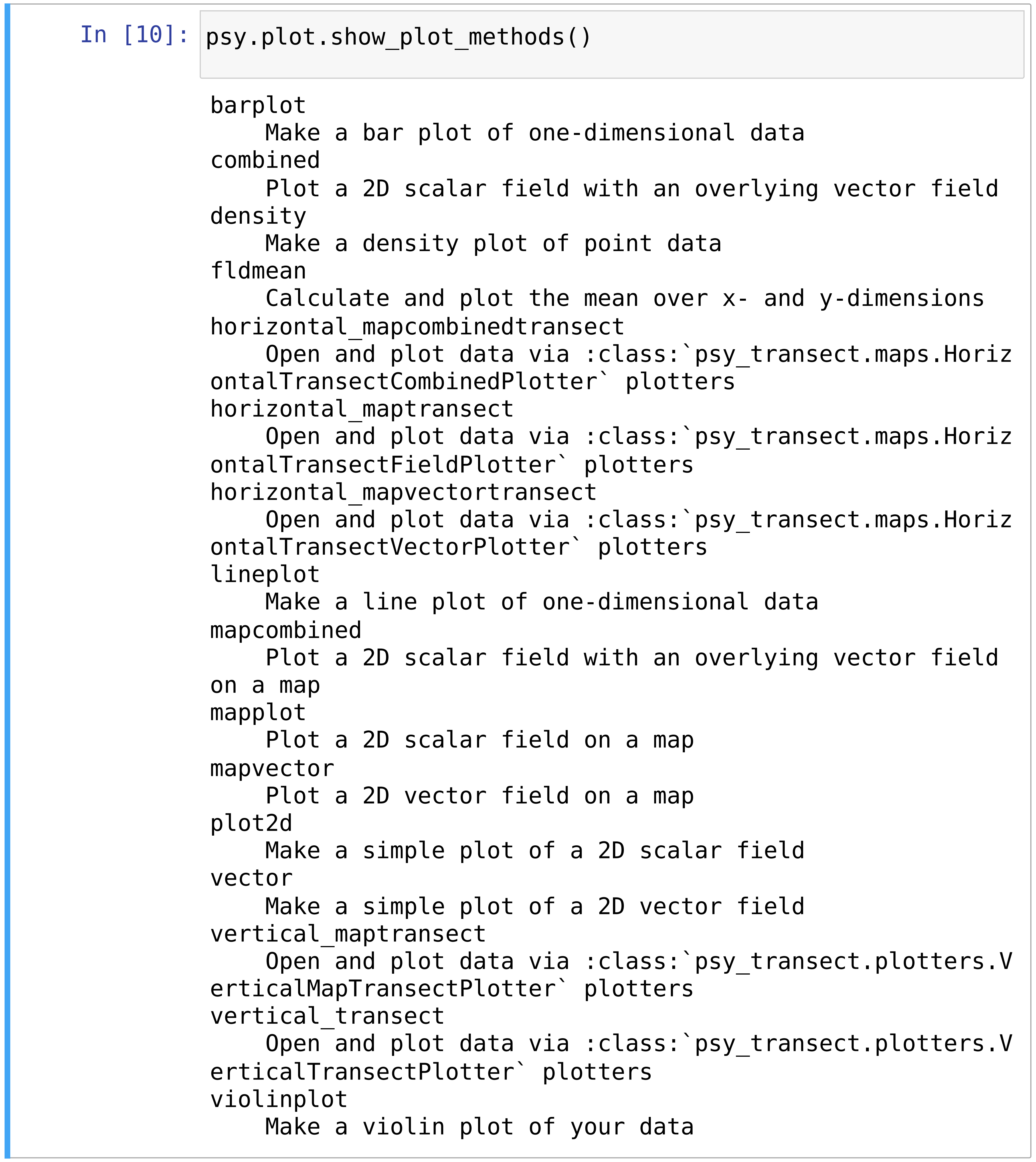
Plugins for visualization

psyplot is the core that defines the framework (Plotter, Formatoption, Project, CFDecoder), the plot methods are implemented by plugins:

- `psy-simple`: for standard 1D and 2D plot
  - e.g. lineplot, plot2d, vector, barplot
- `psy-maps`: for georeferenced plots (i.e. maps)
  - e.g. mapplot, mapvector, etc.
- `psy-reg`: for regression analysis
  - linreg, densityreg

New plugins

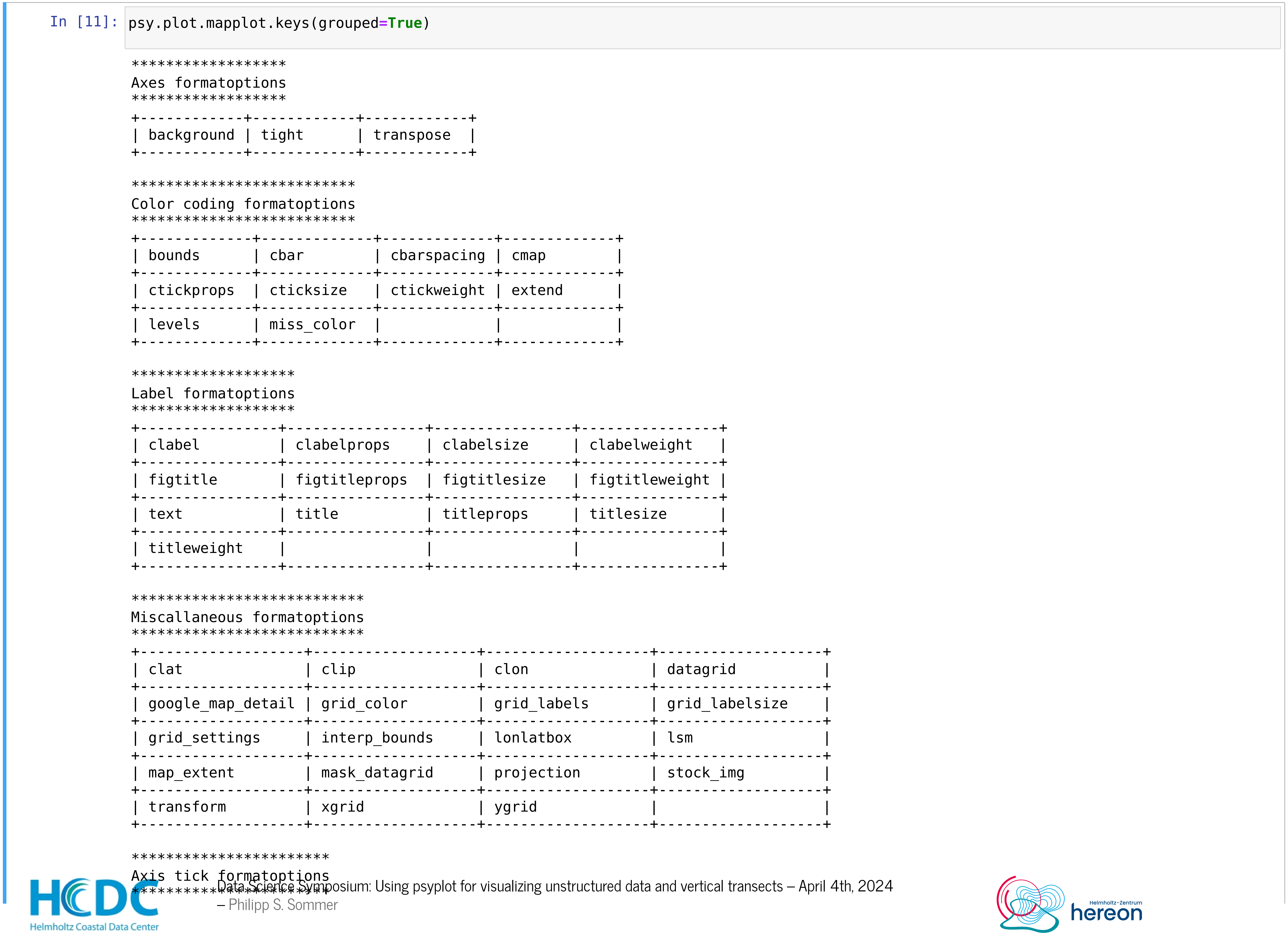
- `psy-ugrid`: for decoding UGRID conventions
- `psy-transect`: for extracting, visualizing and analyzing vertical transects



```
In [10]: psy.plot.show_plot_methods()
```

```
barplot
    Make a bar plot of one-dimensional data
combined
    Plot a 2D scalar field with an overlying vector field
density
    Make a density plot of point data
fldmean
    Calculate and plot the mean over x- and y-dimensions
horizontal_mapcombinedtransect
    Open and plot data via :class:`psy_transect.maps.Horiz
ontalTransectCombinedPlotter` plotters
horizontal_maptransect
    Open and plot data via :class:`psy_transect.maps.Horiz
ontalTransectFieldPlotter` plotters
horizontal_mapvectortransect
    Open and plot data via :class:`psy_transect.maps.Horiz
ontalTransectVectorPlotter` plotters
lineplot
    Make a line plot of one-dimensional data
mapcombined
    Plot a 2D scalar field with an overlying vector field
on a map
mapplot
    Plot a 2D scalar field on a map
mapvector
    Plot a 2D vector field on a map
plot2d
    Make a simple plot of a 2D scalar field
vector
    Make a simple plot of a 2D vector field
vertical_maptransect
    Open and plot data via :class:`psy_transect.plotters.V
erticalMapTransectPlotter` plotters
vertical_transect
    Open and plot data via :class:`psy_transect.plotters.V
erticalTransectPlotter` plotters
violinplot
    Make a violin plot of your data
```

They already have a lot of formatoptions available



```
In [11]: psy.plot.mapplot.keys(grouped=True)
```

```
******************
Axes formatoptions
******************
+------------+------------+------------+
| background | tight      | transpose  |
+------------+------------+------------+

**************************
Color coding formatoptions
**************************
+-------------+-------------+-------------+-------------+
| bounds      | cbar        | cbarspacing | cmap        |
+-------------+-------------+-------------+-------------+
| ctickprops  | cticksize   | ctickweight | extend      |
+-------------+-------------+-------------+-------------+
| levels      | miss_color  |             |             |
+-------------+-------------+-------------+-------------+

*******************
Label formatoptions
*******************
+----------------+----------------+----------------+----------------+
| clabel         | clabelprops    | clabelsize     | clabelweight   |
+----------------+----------------+----------------+----------------+
| figtitle       | figtitleprops  | figtitlesize   | figtitleweight |
+----------------+----------------+----------------+----------------+
| text           | title          | titleprops     | titlesize      |
+----------------+----------------+----------------+----------------+
| titleweight    |                |                |                |
+----------------+----------------+----------------+----------------+

***************************
Miscallaneous formatoptions
***************************
+-------------------+-------------------+-------------------+-------------------+
| clat              | clip              | clon              | datagrid          |
+-------------------+-------------------+-------------------+-------------------+
| google_map_detail | grid_color        | grid_labels       | grid_labelsize    |
+-------------------+-------------------+-------------------+-------------------+
| grid_settings     | interp_bounds     | lonlatbox         | lsm               |
+-------------------+-------------------+-------------------+-------------------+
| map_extent        | mask_datagrid     | projection        | stock_img         |
+-------------------+-------------------+-------------------+-------------------+
| transform         | xgrid             | ygrid             |                   |
+-------------------+-------------------+-------------------+-------------------+

***********************
Axis tick formatoptions
***********************
```

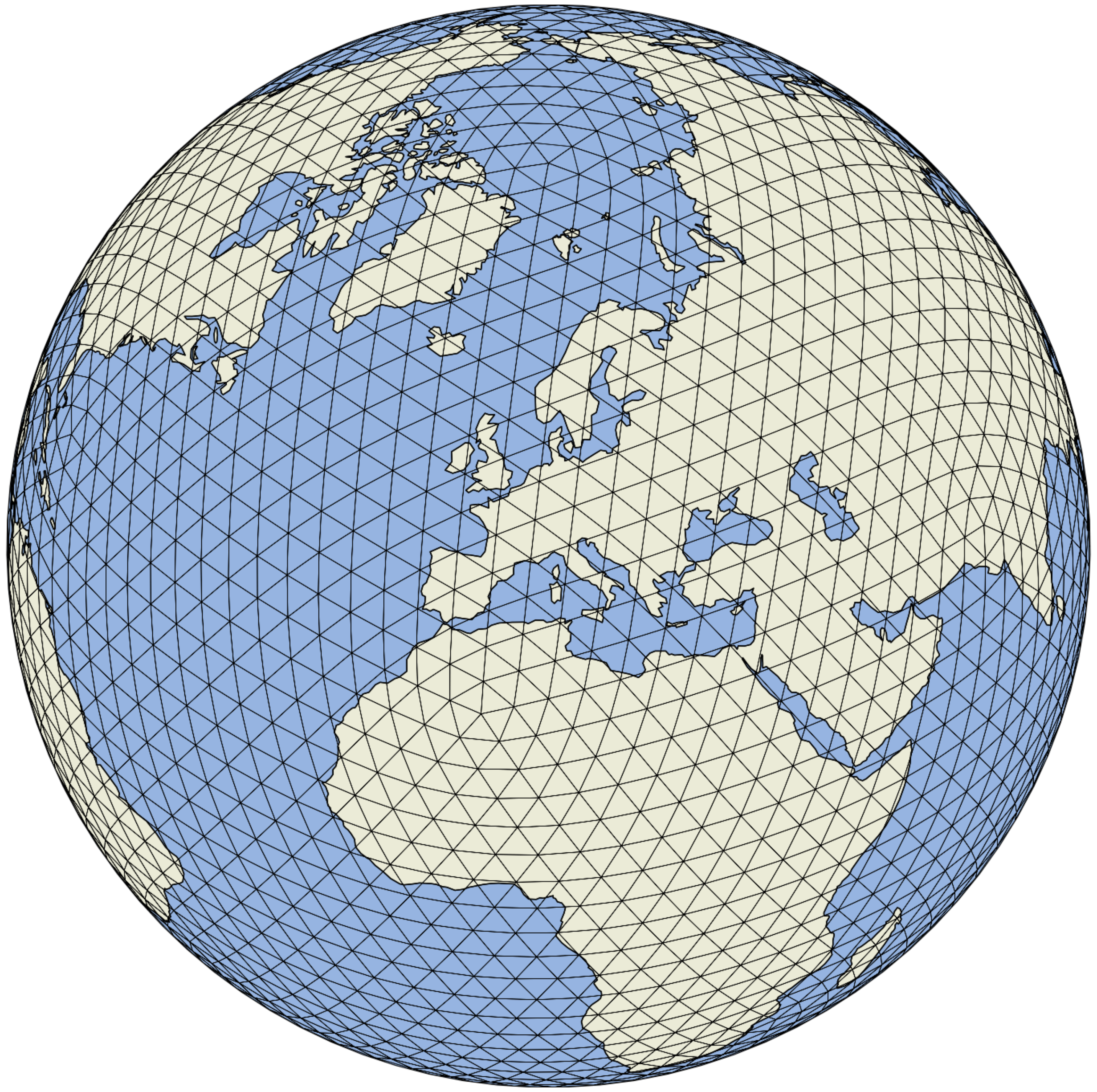
Unstructured Grids

The basics about the ICON Grid

- 1D variables
- 1D coordinates

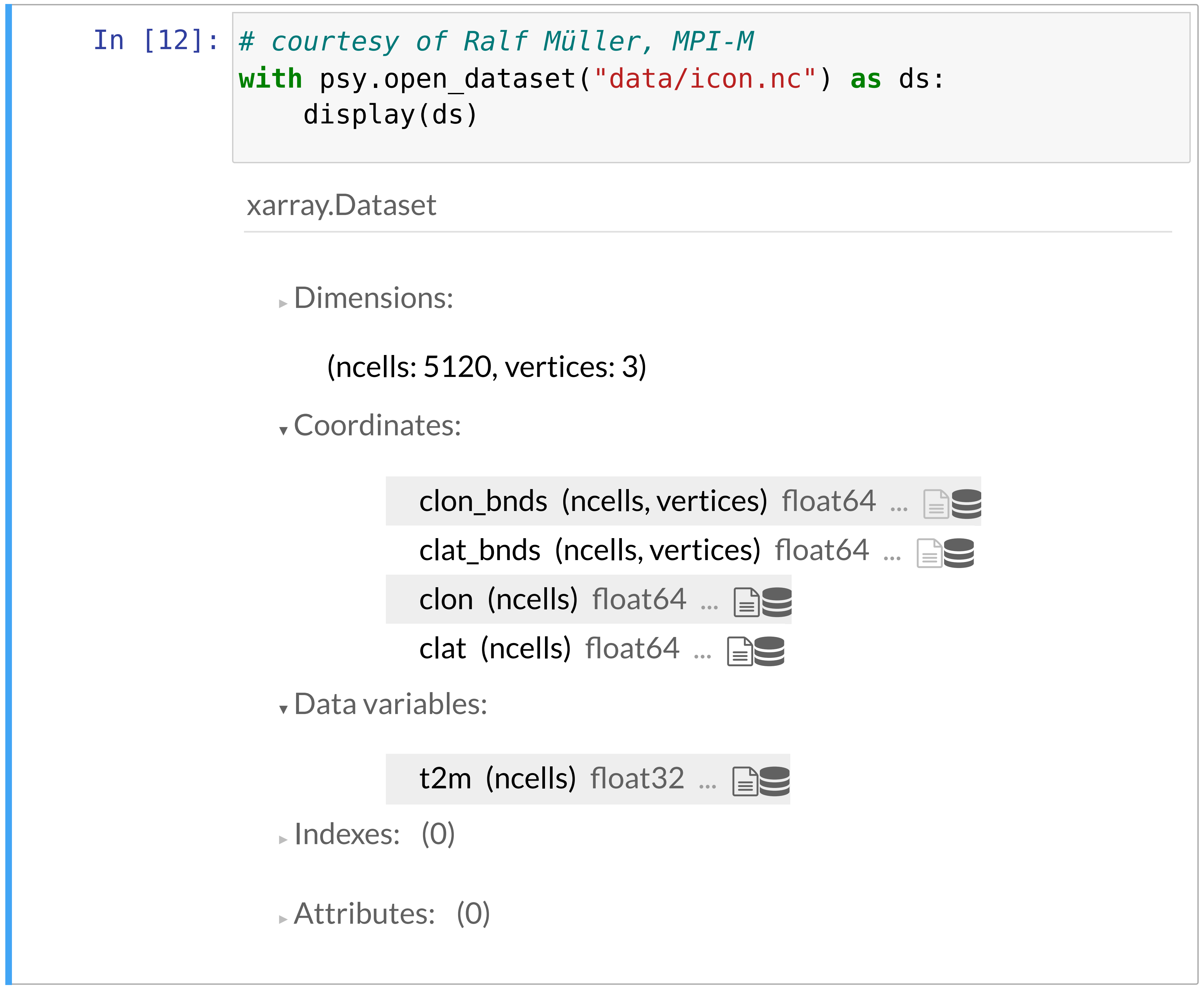
- 2D bounds variable

not supported by standard matplotlib and cartopy approach



Edges and Faces

triangular	edge grid

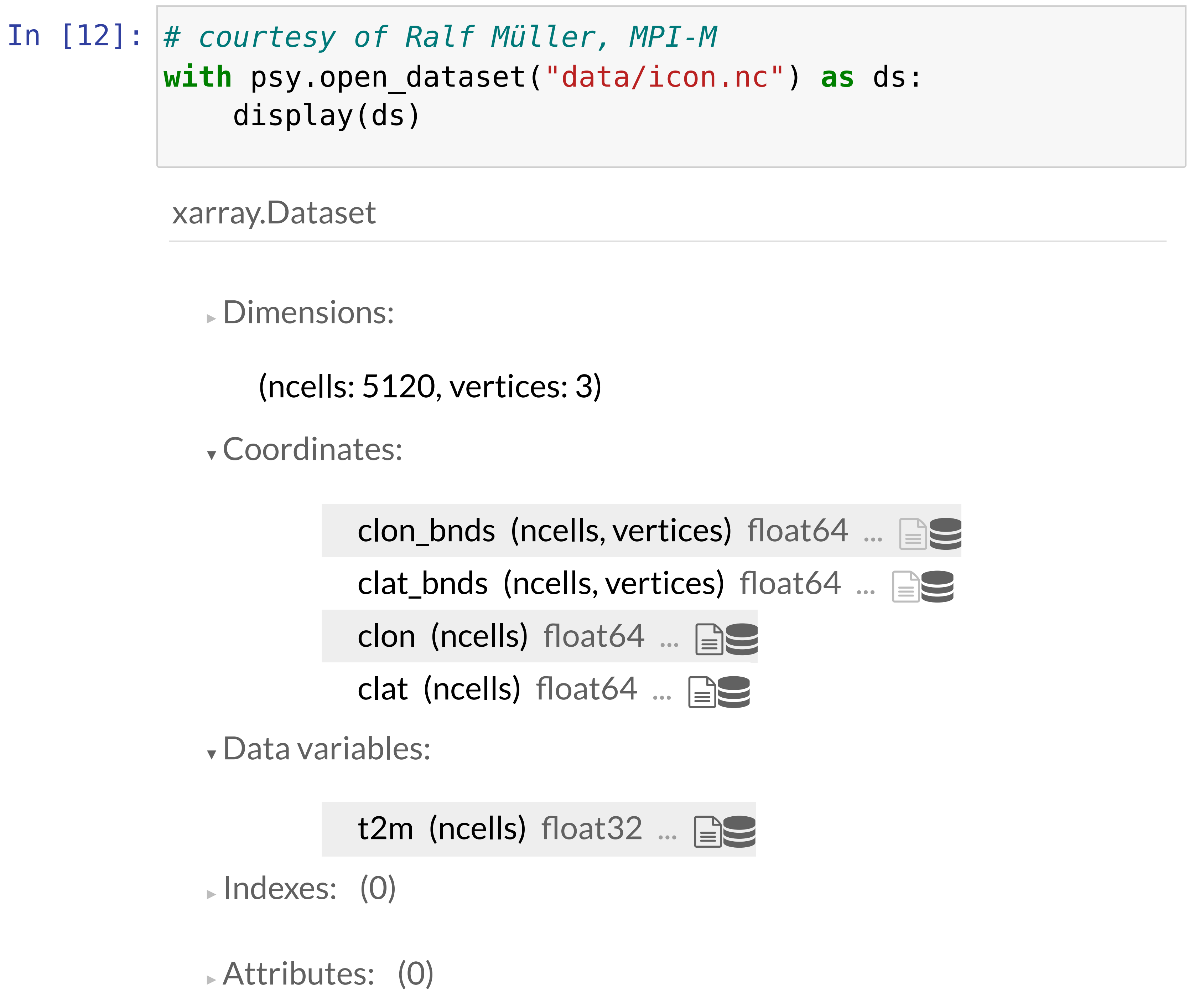


```
In [12]: # courtesy of Ralf Müller, MPI-M
         with psy.open_dataset("data/icon.nc") as ds:
             display(ds)
```

```
xarray.Dataset

▸ Dimensions:
    (ncells: 5120, vertices: 3)
▾ Coordinates:
      clon_bnds  (ncells, vertices)  float64  ...
      clat_bnds  (ncells, vertices)  float64  ...
      clon  (ncells)  float64  ...
      clat  (ncells)  float64  ...
▾ Data variables:
      t2m  (ncells)  float32  ...
▸ Indexes:  (0)
▸ Attributes:  (0)
```

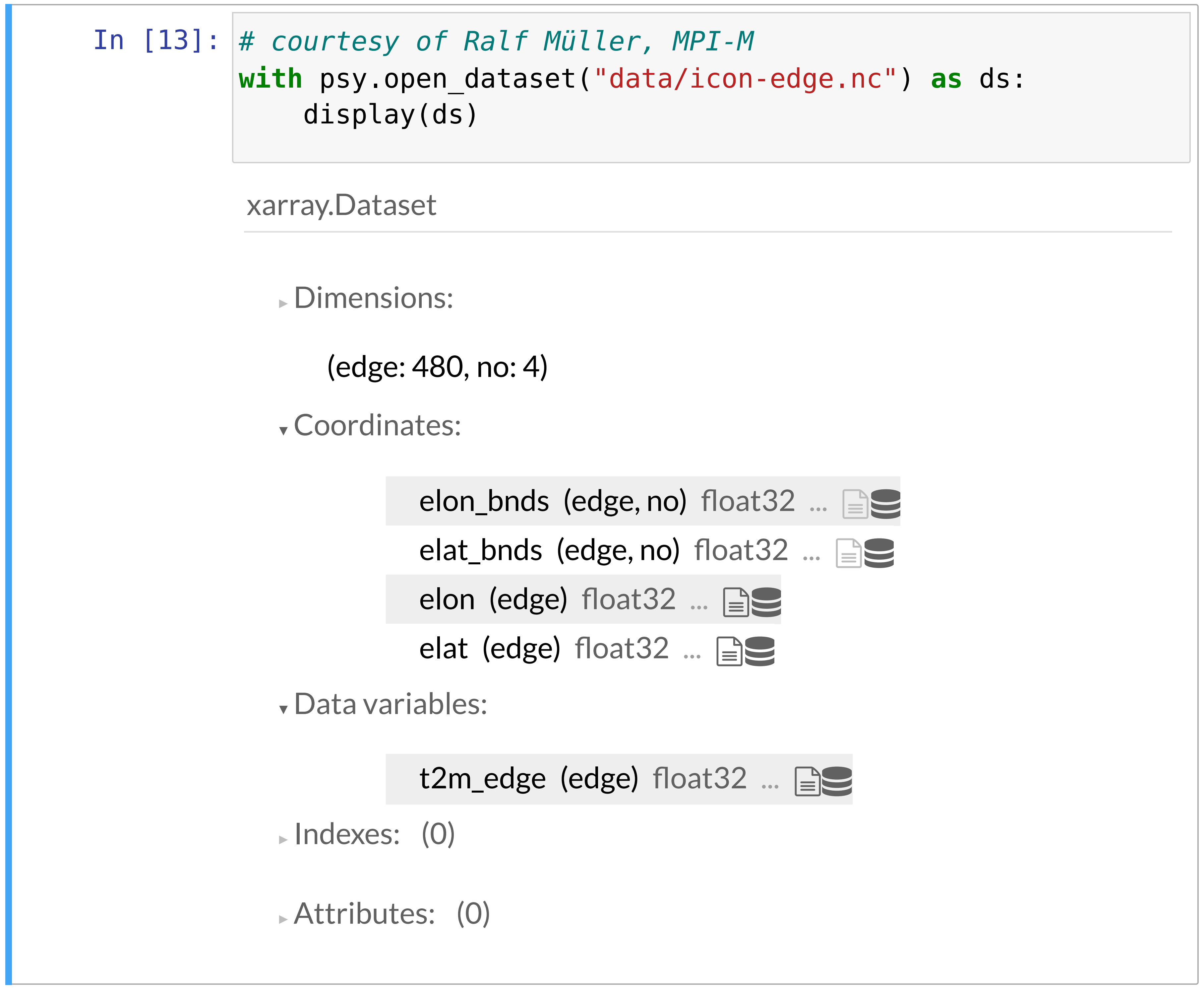
Edges and Faces

triangular	edge grid



```
In [12]: # courtesy of Ralf Müller, MPI-M
         with psy.open_dataset("data/icon.nc") as ds:
             display(ds)
```

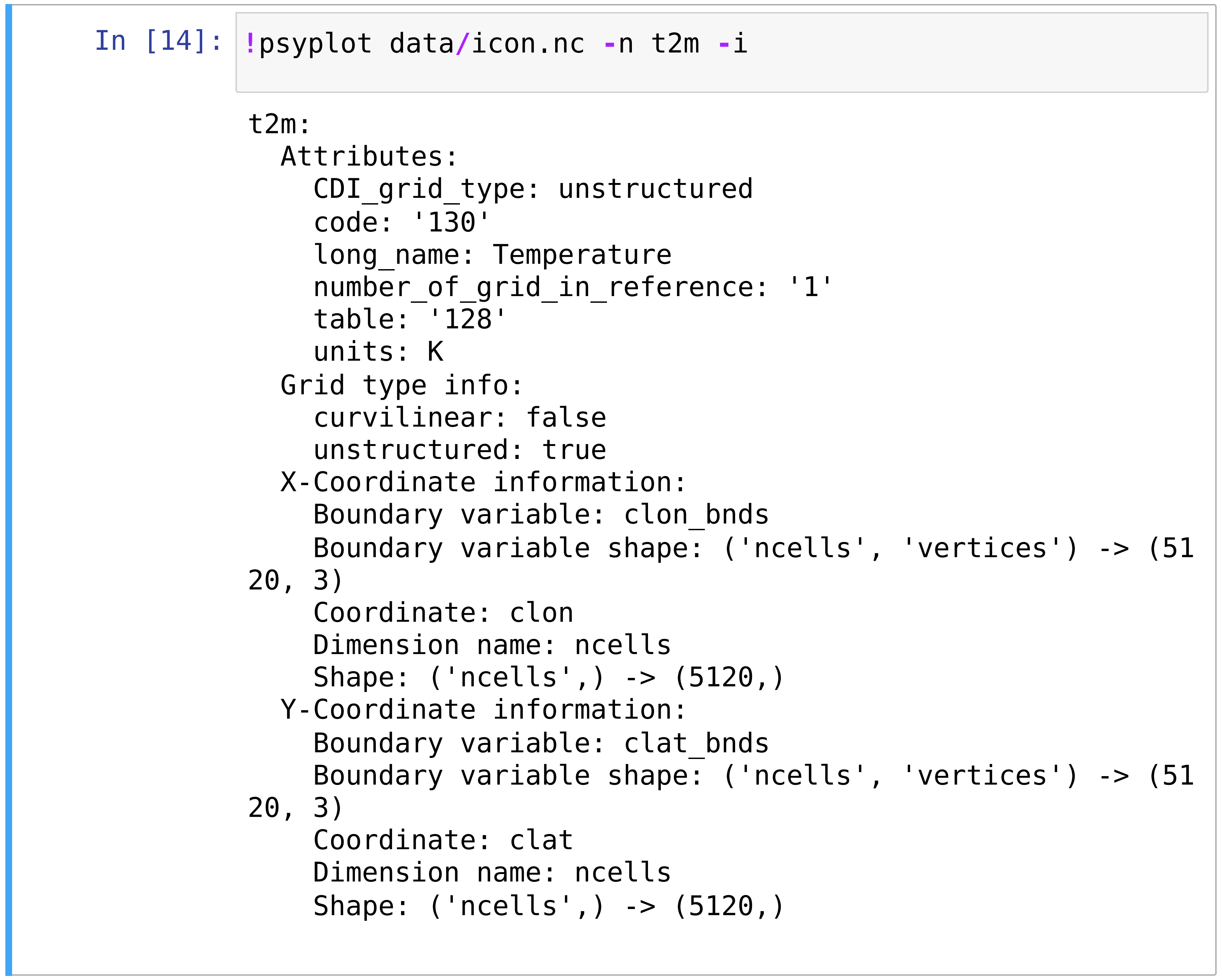
```
xarray.Dataset

▸ Dimensions:
    (ncells: 5120, vertices: 3)
▾ Coordinates:
        clon_bnds  (ncells, vertices)  float64  ...
        clat_bnds  (ncells, vertices)  float64  ...
        clon  (ncells)  float64  ...
        clat  (ncells)  float64  ...
▾ Data variables:
        t2m  (ncells)  float32  ...
▸ Indexes:  (0)
▸ Attributes:  (0)
```



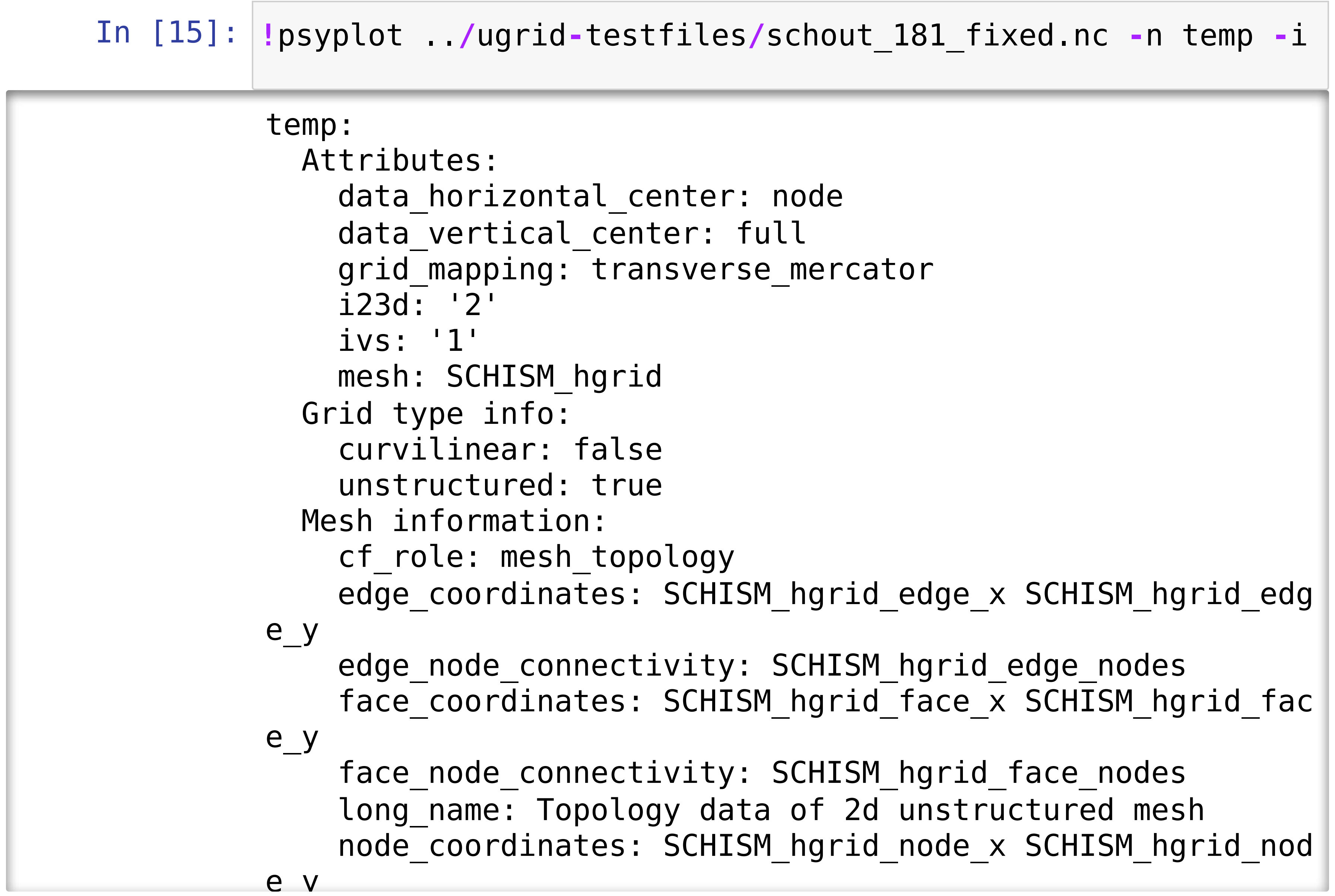
```
In [13]: # courtesy of Ralf Müller, MPI-M
         with psy.open_dataset("data/icon-edge.nc") as ds:
             display(ds)
```

```
xarray.Dataset

▸ Dimensions:
    (edge: 480, no: 4)
▾ Coordinates:
        elon_bnds  (edge, no)  float32  ...
        elat_bnds  (edge, no)  float32  ...
        elon  (edge)  float32  ...
        elat  (edge)  float32  ...
▾ Data variables:
        t2m_edge  (edge)  float32  ...
▸ Indexes:  (0)
▸ Attributes:  (0)
```

Decoding unstructured information

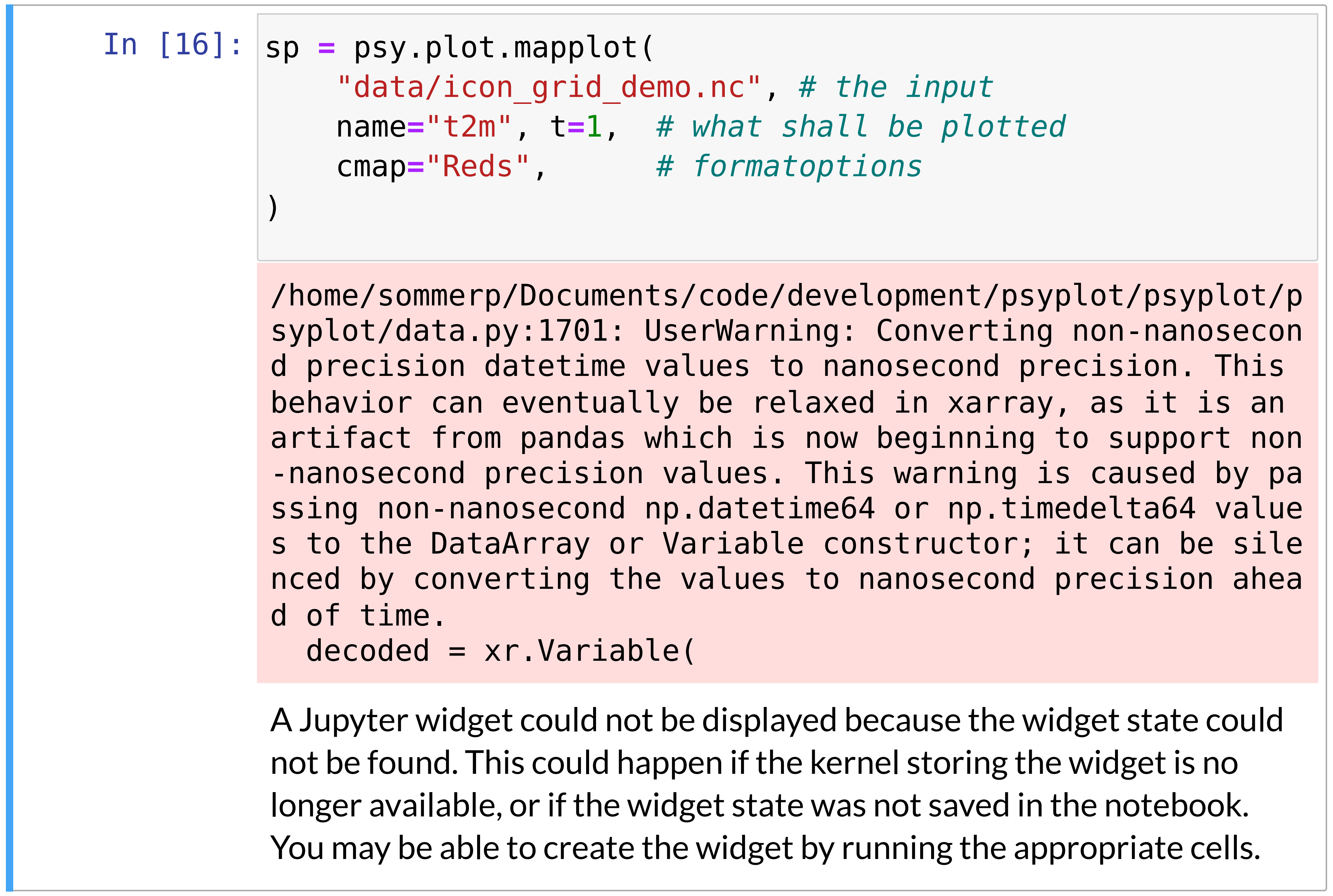


```
In [14]: !psyplot data/icon.nc -n t2m -i

         t2m:
           Attributes:
             CDI_grid_type: unstructured
             code: '130'
             long_name: Temperature
             number_of_grid_in_reference: '1'
             table: '128'
             units: K
           Grid type info:
             curvilinear: false
             unstructured: true
           X-Coordinate information:
             Boundary variable: clon_bnds
             Boundary variable shape: ('ncells', 'vertices') -> (51
         20, 3)
             Coordinate: clon
             Dimension name: ncells
             Shape: ('ncells',) -> (5120,)
           Y-Coordinate information:
             Boundary variable: clat_bnds
             Boundary variable shape: ('ncells', 'vertices') -> (51
         20, 3)
             Coordinate: clat
             Dimension name: ncells
             Shape: ('ncells',) -> (5120,)
```



```
In [15]: !psyplot ../ugrid-testfiles/schout_181_fixed.nc -n temp -i

         temp:
           Attributes:
             data_horizontal_center: node
             data_vertical_center: full
             grid_mapping: transverse_mercator
             i23d: '2'
             ivs: '1'
             mesh: SCHISM_hgrid
           Grid type info:
             curvilinear: false
             unstructured: true
           Mesh information:
             cf_role: mesh_topology
             edge_coordinates: SCHISM_hgrid_edge_x SCHISM_hgrid_edg
         e_y
             edge_node_connectivity: SCHISM_hgrid_edge_nodes
             face_coordinates: SCHISM_hgrid_face_x SCHISM_hgrid_fac
         e_y
             face_node_connectivity: SCHISM_hgrid_face_nodes
             long_name: Topology data of 2d unstructured mesh
             node_coordinates: SCHISM_hgrid_node_x SCHISM_hgrid_nod
         e_y
```

Visualization with psyplot



In [16]:

```
sp = psy.plot.mapplot(
    "data/icon_grid_demo.nc", # the input
    name="t2m", t=1,  # what shall be plotted
    cmap="Reds",      # formatoptions
)
```

```
/home/sommerp/Documents/code/development/psyplot/psyplot/p
syplot/data.py:1701: UserWarning: Converting non-nanosecon
d precision datetime values to nanosecond precision. This
behavior can eventually be relaxed in xarray, as it is an
artifact from pandas which is now beginning to support non
-nanosecond precision values. This warning is caused by pa
ssing non-nanosecond np.datetime64 or np.timedelta64 value
s to the DataArray or Variable constructor; it can be sile
nced by converting the values to nanosecond precision ahea
d of time.
  decoded = xr.Variable(
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Visualization with psyplot

```
In [16]: sp = psy.plot.mapplot(
             "data/icon_grid_demo.nc", # the input
             name="t2m", t=1,  # what shall be plotted
             cmap="Reds",      # formatoptions
         )
```

```
/home/sommerp/Documents/code/development/psyplot/psyplot/p
syplot/data.py:1701: UserWarning: Converting non-nanosecon
d precision datetime values to nanosecond precision. This
behavior can eventually be relaxed in xarray, as it is an
artifact from pandas which is now beginning to support non
-nanosecond precision values. This warning is caused by pa
ssing non-nanosecond np.datetime64 or np.timedelta64 value
s to the DataArray or Variable constructor; it can be sile
nced by converting the values to nanosecond precision ahea
d of time.
  decoded = xr.Variable(
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
In [17]: # update the projection
         # this can take any cartopy projection, but we
         # have a couple of shortcuts
         sp.update(projection="ortho")
```

```
In [18]: # change the lonlatbox
         sp.update(lonlatbox="Europe")
         sp.draw()
```

```
In [19]: # mask certain values
         sp.update(maskleq=280)
```

```
In [20]: # set a title
         sp.update(title="Month: %B %Y")
```

```
In [21]: sp.update(google_map_detail=4)
```

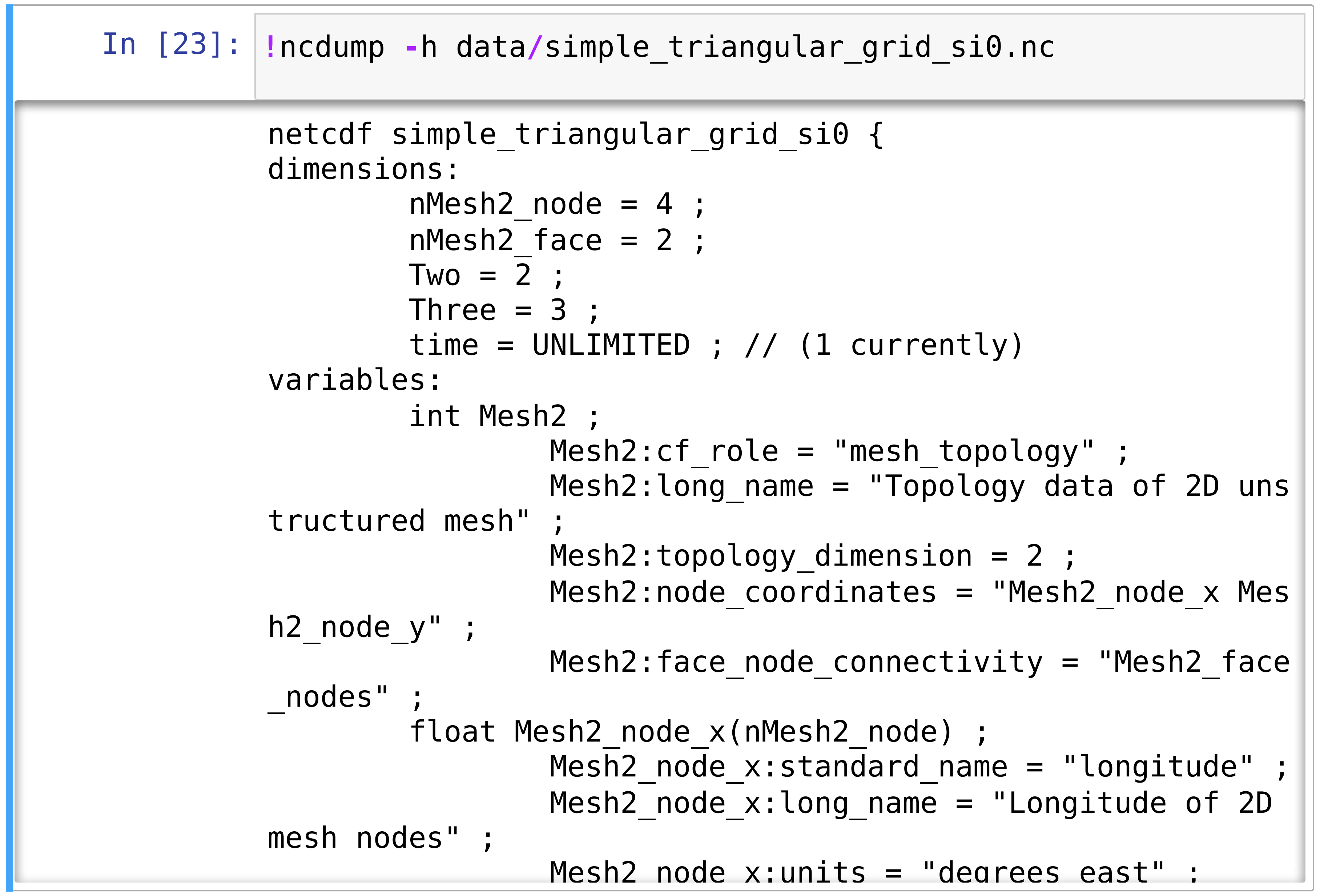
```
In [22]: psy.close('all')
```

UGRID conventions

> *ICON files contain the grid information as grid cell boundaries, UGRID stores the mesh connectivity*

Mesh

```
    int Mesh2 ;
        Mesh2:cf_role = "mesh_topology" ;
        Mesh2:long_name = "Topology data 
2D unstructured mesh" ;
        Mesh2:topology_dimension = 2 ;
        Mesh2:node_coordinates = "Mesh2_n
e_x Mesh2_node_y" ;
        Mesh2:face_node_connectivity = "M
h2_face_nodes" ;
```



In [23]:
```
!ncdump -h data/simple_triangular_grid_si0.nc
```

```
netcdf simple_triangular_grid_si0 {
dimensions:
        nMesh2_node = 4 ;
        nMesh2_face = 2 ;
        Two = 2 ;
        Three = 3 ;
        time = UNLIMITED ; // (1 currently)
variables:
        int Mesh2 ;
                Mesh2:cf_role = "mesh_topology" ;
                Mesh2:long_name = "Topology data of 2D uns
tructured mesh" ;
                Mesh2:topology_dimension = 2 ;
                Mesh2:node_coordinates = "Mesh2_node_x Mes
h2_node_y" ;
                Mesh2:face_node_connectivity = "Mesh2_face
_nodes" ;
        float Mesh2_node_x(nMesh2_node) ;
                Mesh2_node_x:standard_name = "longitude" ;
                Mesh2_node_x:long_name = "Longitude of 2D
mesh nodes" ;
                Mesh2_node_x:units = "degrees_east" ;
```

psy-ugrid: Visualizing triangular and flexible grids

- new repository open-source on GitLab: https://codebase.helmholtz.cloud/psyplot/psy-ugrid/
- efficient computation of the dual node or edge mesh solely
- integrated into psyplots decoding framework

```
In [24]: ds = psy.open_dataset("data/simple_triangular_grid_si0.nc")
         ds.Mesh2_fcvar.psy.decoder

Out[24]: <psy_ugrid.decoder.UGridDecoder at 0x7840c3968ad0>
```

```
In [25]: ds.psy.plot.mapplot(name="Mesh2_fcvar")

Out[25]: psyplot.project.Project([    arr0: 1-dim DataArray of Mesh
         2_fcvar, with (nMesh2_face)=(2,), Mesh2=-2147483647, time=
         1951-01-01])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
In [26]: ds.psy.plot.mapplot(name="Mesh2_ndvar")

Out[26]: psyplot.project.Project([    arr1: 1-dim DataArray of Mesh
         2_ndvar, with (nMesh2_node)=(4,), Mesh2=-2147483647, time=
         1951-01-01])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Visualizing flexible grids

The UGRID conventions support flexible meshes, i.e. mixes of triangles, rectangles, hexagonals, etc.

```
In [27]: ds = psy.open_dataset("data/simple_flexible_grid_si0.nc")
```

```
In [28]: ds.psy.plot.mapplot(name="Mesh2_fcvar")
```

```
Out[28]: psyplot.project.Project([    arr2: 1-dim DataArray of Mesh
2_fcvar, with (nMesh2_face)=(2,), Mesh2=-2147483647, time=
1951-01-01])
```

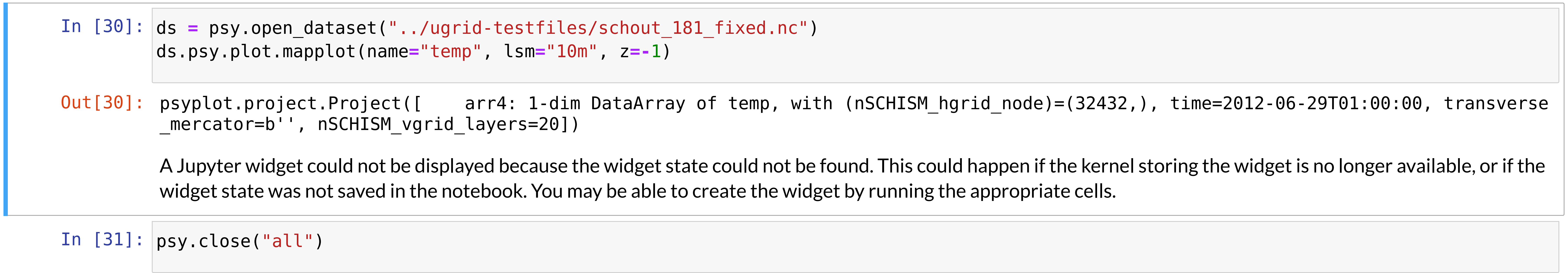
A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
In [29]: ds.psy.plot.mapplot(name="Mesh2_ndvar")
```

```
Out[29]: psyplot.project.Project([    arr3: 1-dim DataArray of Mesh
2_ndvar, with (nMesh2_node)=(5,), Mesh2=-2147483647, time=
1951-01-01])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

This works efficiently for all unstructured grids



In [30]:
```
ds = psy.open_dataset("../ugrid-testfiles/schout_181_fixed.nc")
ds.psy.plot.mapplot(name="temp", lsm="10m", z=-1)
```

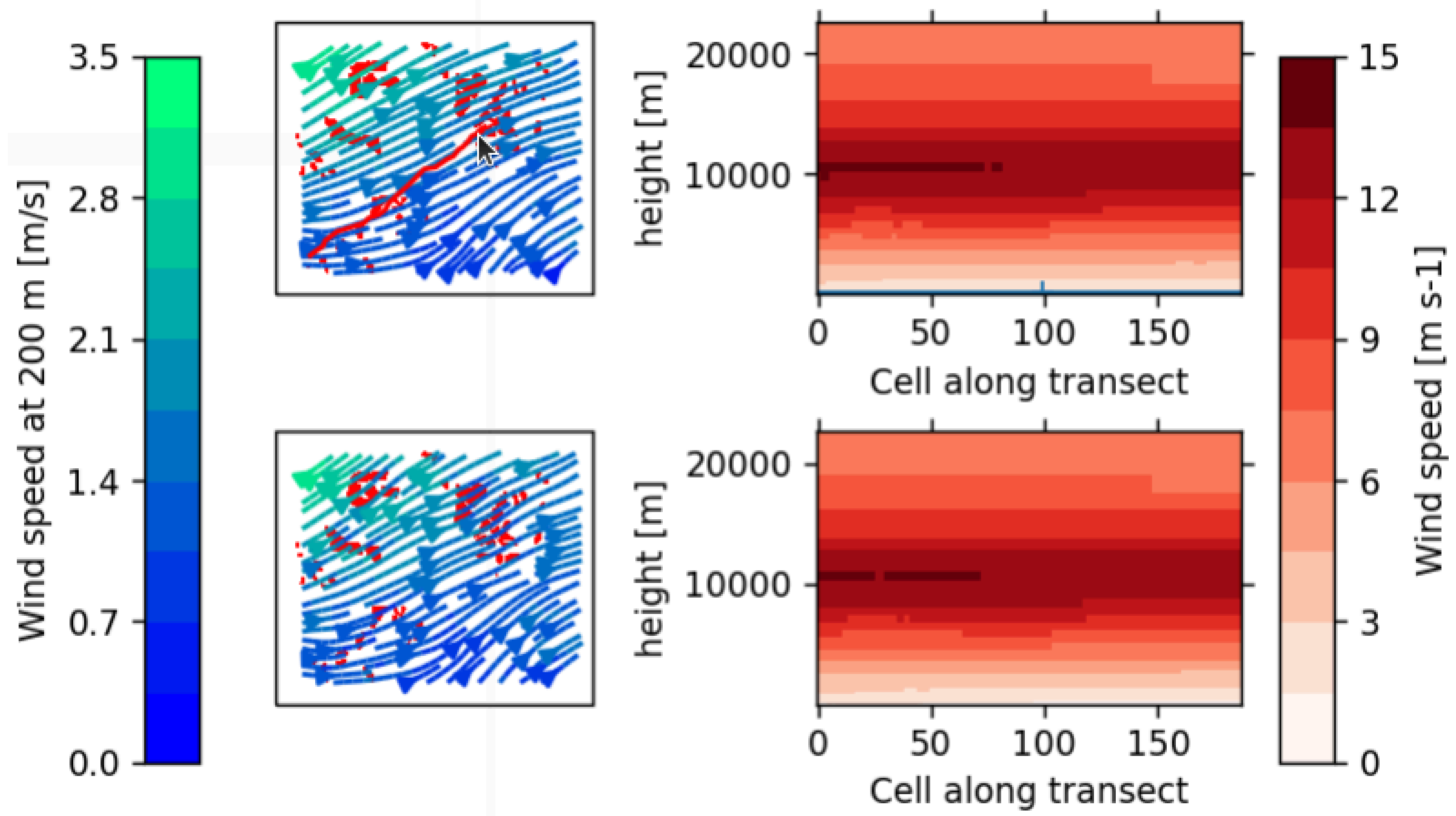
Out[30]:
```
psyplot.project.Project([    arr4: 1-dim DataArray of temp, with (nSCHISM_hgrid_node)=(32432,), time=2012-06-29T01:00:00, transverse
_mercator=b'', nSCHISM_vgrid_layers=20])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [31]:
```
psy.close("all")
```

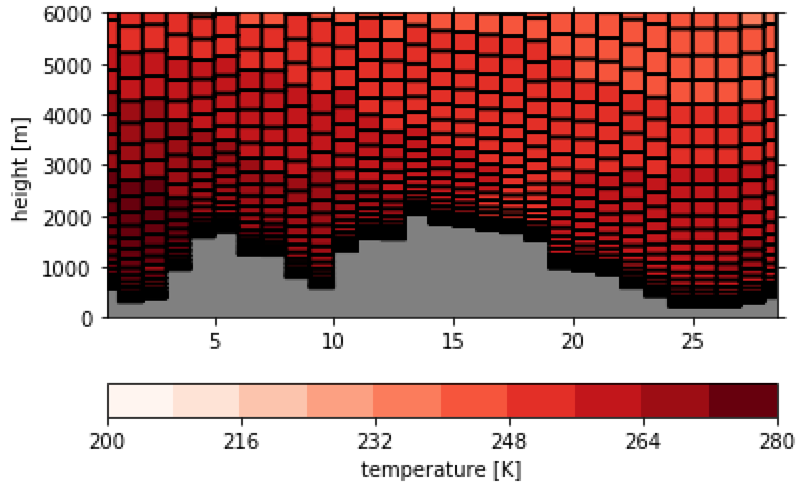
Transects

Analysing horizontal and vertical transects



psy-transect

- interpolate rastered (or unstructured) data on to a path.
- display vertical profiles (with respect to orography, if available)
- supports vector (wind) or scalar fields (temperature)
- made for interactive usage



COSMO-CLM data (standard case)

In [32]:
```
temperature_ds = psy.open_dataset("data/T.nc")
temperature_ds.psy.plot.horizontal_maptransect(
    name="T",
    transect=0,
    cmap="Reds",
    title="Layer at height %(transect)1.2f m",
)
```

Out[32]:
```
psyplot.project.Project([    arr0: 3-dim DataArray of T, w
ith (level, rlat, rlon)=(40, 111, 101), rotated_pole=b'',
time=1983-12-01T21:00:00])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [33]:
```
sp = temperature_ds.psy.plot.vertical_maptransect(
    name="T",
    plot="poly",
    background="0.5",
    datagrid="k-",
    transform="cyl",
)
p1, p2 = psy.gcp(True).plotters
p1.connect_ax(p2)
p2.connect_ax(p1)
```

Out[33]:
```
<matplotlib.widgets.LassoSelector at 0x7841e18c67d0>
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [34]:
```
psy.close("all")
```

COSMO-CLM with vertical level information

```
In [35]: from psy_transect import utils

         temperature_ds = psy.open_dataset("data/T.nc")
         orography = psy.open_dataset("data/lffd1980010100c.nc").psy

         new_ds = utils.mesh_to_cf_bounds(orography, "level1", "leve
         new_ds.psy.plot.horizontal_maptransect(
             name="T",
             transect=0,
             cmap="Reds",
             title="Layer at height %(transect)1.2f m",
         )
```

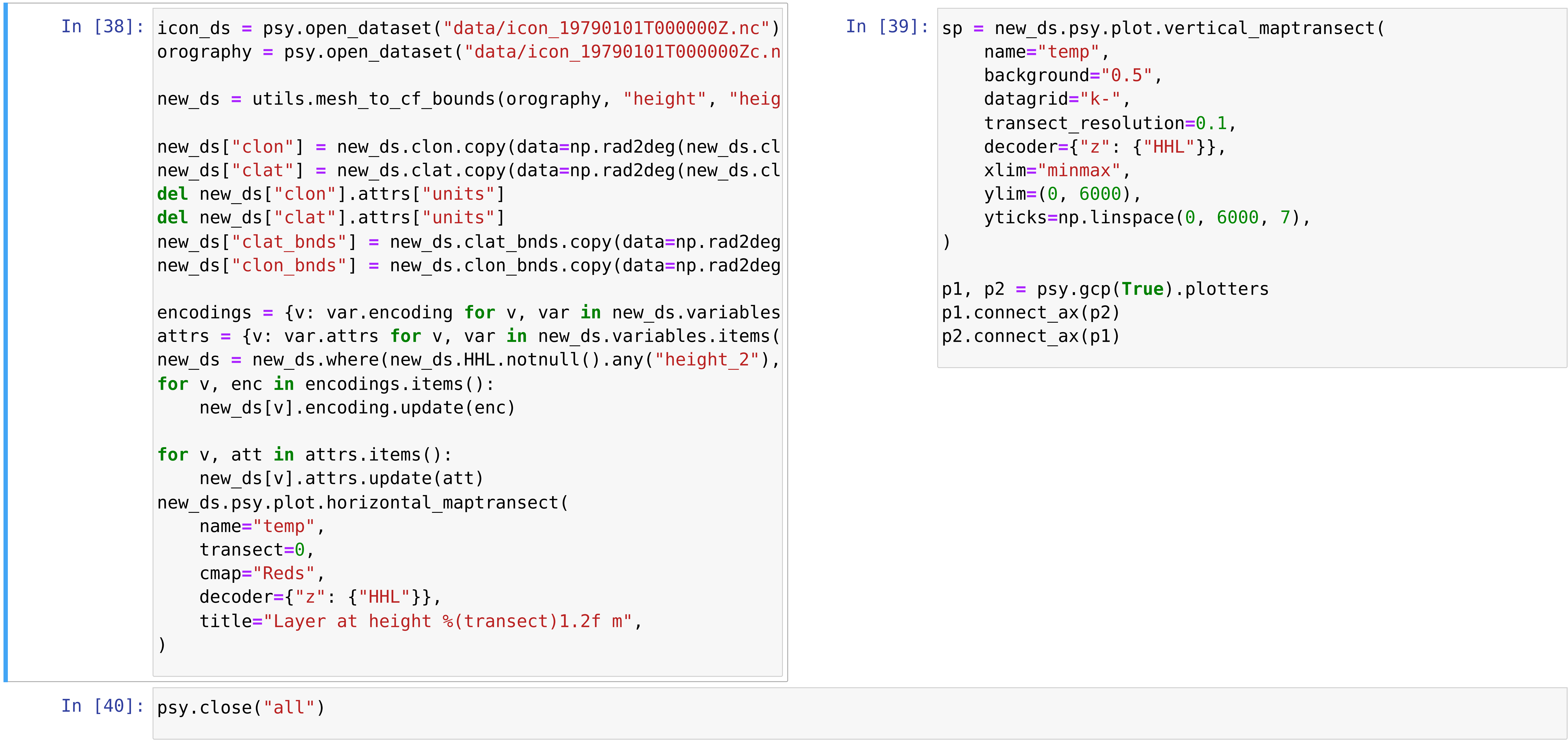
```
Out[35]: psyplot.project.Project([    arr0: 3-dim DataArray of T, w
         ith (level, rlat, rlon)=(40, 111, 101), rotated_pole=b'',
         time=1983-12-01T21:00:00])
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
In [36]: sp = new_ds.psy.plot.vertical_maptransect(
             name="T",
             plot="poly",
             background="0.5",
             datagrid="k-",
             transform="cyl",
             xlim="minmax",
             ylim=(0, 6000),
             yticks=np.linspace(0, 6000, 7),
         )

         p1, p2 = psy.gcp(True).plotters
         slider = p1.connect_ax(p2)
         p2.connect_ax(p1)
```

```
In [37]: psy.close("all")
```

ICON-CLM extraction



In [38]:
```
icon_ds = psy.open_dataset("data/icon_19790101T000000Z.nc")
orography = psy.open_dataset("data/icon_19790101T000000Zc.n

new_ds = utils.mesh_to_cf_bounds(orography, "height", "heig

new_ds["clon"] = new_ds.clon.copy(data=np.rad2deg(new_ds.cl
new_ds["clat"] = new_ds.clat.copy(data=np.rad2deg(new_ds.cl
del new_ds["clon"].attrs["units"]
del new_ds["clat"].attrs["units"]
new_ds["clat_bnds"] = new_ds.clat_bnds.copy(data=np.rad2deg
new_ds["clon_bnds"] = new_ds.clon_bnds.copy(data=np.rad2deg

encodings = {v: var.encoding for v, var in new_ds.variables
attrs = {v: var.attrs for v, var in new_ds.variables.items(
new_ds = new_ds.where(new_ds.HHL.notnull().any("height_2"),
for v, enc in encodings.items():
    new_ds[v].encoding.update(enc)

for v, att in attrs.items():
    new_ds[v].attrs.update(att)
new_ds.psy.plot.horizontal_maptransect(
    name="temp",
    transect=0,
    cmap="Reds",
    decoder={"z": {"HHL"}},
    title="Layer at height %(transect)1.2f m",
)
```

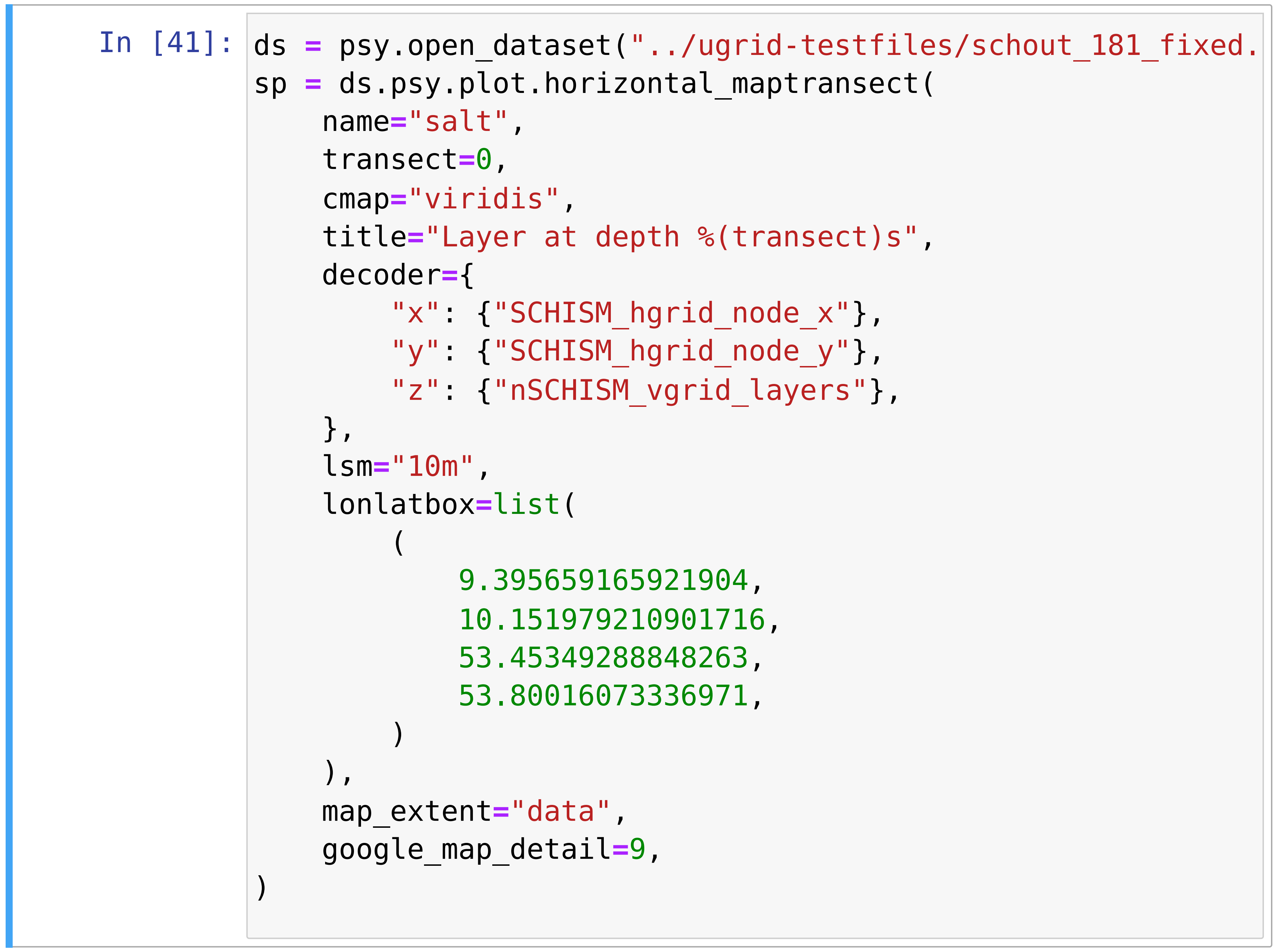
In [39]:
```
sp = new_ds.psy.plot.vertical_maptransect(
    name="temp",
    background="0.5",
    datagrid="k-",
    transect_resolution=0.1,
    decoder={"z": {"HHL"}},
    xlim="minmax",
    ylim=(0, 6000),
    yticks=np.linspace(0, 6000, 7),
)

p1, p2 = psy.gcp(True).plotters
p1.connect_ax(p2)
p2.connect_ax(p1)
```

In [40]:
```
psy.close("all")
```

HCDC Helmholtz Coastal Data Center

Helmholtz-Zentrum hereon

Vertical transect of SCHISM output



In [41]:

```
ds = psy.open_dataset("../ugrid-testfiles/schout_181_fixed.
sp = ds.psy.plot.horizontal_maptransect(
    name="salt",
    transect=0,
    cmap="viridis",
    title="Layer at depth %(transect)s",
    decoder={
        "x": {"SCHISM_hgrid_node_x"},
        "y": {"SCHISM_hgrid_node_y"},
        "z": {"nSCHISM_vgrid_layers"},
    },
    lsm="10m",
    lonlatbox=list(
        (
            9.395659165921904,
            10.151979210901716,
            53.45349288848263,
            53.80016073336971,
        )
    ),
    map_extent="data",
    google_map_detail=9,
)
```

Vertical transect of SCHISM output

In [41]:

```
ds = psy.open_dataset("../ugrid-testfiles/schout_181_fixed.
sp = ds.psy.plot.horizontal_maptransect(
    name="salt",
    transect=0,
    cmap="viridis",
    title="Layer at depth %(transect)s",
    decoder={
        "x": {"SCHISM_hgrid_node_x"},
        "y": {"SCHISM_hgrid_node_y"},
        "z": {"nSCHISM_vgrid_layers"},
    },
    lsm="10m",
    lonlatbox=list(
        (
            9.395659165921904,
            10.151979210901716,
            53.45349288848263,
            53.80016073336971,
        )
    ),
    map_extent="data",
    google_map_detail=9,
)
```

In [42]:

```


ds.psy.plot.vertical_maptransect(
    name="salt",
    plot="poly",
    transect_resolution=1.0,
    decoder={
        "x": {"SCHISM_hgrid_node_x"},
        "y": {"SCHISM_hgrid_node_y"},
        "z": {"nSCHISM_vgrid_layers"},
    },
)
p1, p2 = psy.gcp(True).plotters
p1.connect_ax(p2)
p2.connect_ax(p1)
```

In [43]:

```
psy.close("all")
```

Summary

- the psyplot core for the data model, and plugins for various visualizations
- designed to be flexible and sustainable
- equipped via flexible graphical user interface

Summary

- the psyplot core for the data model, and plugins for various visualizations
- designed to be flexible and sustainable
- equipped via flexible graphical user interface

The data model

- based on a netCDF-like infrastructure and interpretes CF- and UGRID conventions
- support for multiple grids: rectilinear, circumpolar and unstructured

Summary

- the psyplot core for the data model, and plugins for various visualizations
- designed to be flexible and sustainable
- equipped via flexible graphical user interface

The data model

- based on a netCDF-like infrastructure and interpretes CF- and UGRID conventions
- support for multiple grids: rectilinear, circumpolar and unstructured

Flexibility

- convenient python API
- usage via GUI and psy-view
- usage from the command-line

Summary

- the psyplot core for the data model, and plugins for various visualizations
- designed to be flexible and sustainable
- equipped via flexible graphical user interface

The data model

- based on a netCDF-like infrastructure and interpretes CF- and UGRID conventions
- support for multiple grids: rectilinear, circumpolar and unstructured

Flexibility

- convenient python API
- usage via GUI and psy-view
- usage from the command-line

New plugins

- `psy-ugrid` decodes UGRID conventions and (if necessary) computes the dual mesh for visualization on the native grid
- `psy-transect` provides interactive analysis features for horizontal and vertical transects

Summary

- the psyplot core for the data model, and plugins for various visualizations
- designed to be flexible and sustainable
- equipped via flexible graphical user interface

The data model

- based on a netCDF-like infrastructure and interpretes CF- and UGRID conventions
- support for multiple grids: rectilinear, circumpolar and unstructured

Flexibility

- convenient python API
- usage via GUI and psy-view
- usage from the command-line

New plugins

- `psy-ugrid` decodes UGRID conventions and (if necessary) computes the dual mesh for visualization on the native grid
- `psy-transect` provides interactive analysis features for horizontal and vertical transects

Important changes